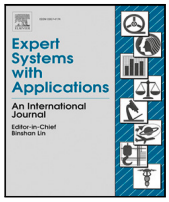
Expert Systems With Applications 255 (2024) 124476

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Stealthy attack on graph recommendation system

Hao Ma [a,b], Min Gao [a,b,*], Feng Wei [c], Zongwei Wang [b], Feng Jiang [b,d], Zehua Zhao [e], Zhengyi Yang [b]

[a] Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 401331, China
[b] School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China
[c] Big Data and AI Center, Chongqing Branch, China Telecom, Chongqing 401121, China
[d] School of Finance and Management, Chongqing Business Vocational College, Chongqing 400036, China
[e] School of Civil Engineering, The University of Sydney, Sydney 2006, Australia

ARTICLE INFO

Keywords:
Recommendation systems
Robustness
Poisoning attack
Stealthiness

ABSTRACT

The graph-based recommendation systems achieve significant success, yet they are accompanied by malicious attacks. In most scenes, attackers will inject crafted fake profiles into the recommendation system to boost the ranking of their target items in the recommendation lists. For an exploration of potential attacks hidden in real life, researchers have proposed various attacks with severe threats. However, current efforts in exploring attack strategies neglect the stealthiness aspect of the attack, i.e., the perturbation of recommendation performance after an attack can be quite noticeable, potentially alerting the defenders. To fill this research gap, this paper introduces a novel attack framework named InfoAtk, designed to conduct attacks while ensuring stealthiness. Specifically, given the dependency of precise recommendation predominantly on representations, the framework employs contrastive learning techniques to align representations before and after the attack, thereby augmenting stealthiness. Additionally, we optimize the representation of target items to outrank the last items in users' recommendation lists, thereby promoting the visibility of the target item to increase the attack's effectiveness. Extensive experiments on four public datasets validate the stealthiness and effectiveness of our proposed attack framework.

1. Introduction

Currently, recommendation systems are widely applied in various domains, such as gaming websites (Lee, Hwang, Kim, Lee, & Choo, 2022), music platforms (Dinnissen, 2022), e-commerce platforms (Liu et al., 2023), and other web services (Fan, Hu, Zheng, Wang, Brézillon et al., 2021). Their primary objective is to enhance users' online experiences and satisfaction by unearthing latent interests at a profound level. Due to the vast amount of information on the Internet, users often rely on recommendation systems to filter information, consequently, these systems have a significant impact on user behavior. However, the collaborative filtering mechanism of recommendation systems is susceptible to manipulation, and the performance of the recommendation systems can be undermined by the crafted fake data introduced by malicious attackers (Gunes, Kaleli, Bilge, & Polat, 2014; Si & Li, 2020). For example, a malicious merchant selling a particular product can inject some well-designed fake user profiles into the recommendation system to promote their own product, making such products more likely to be recommended (Rong, He & Chen, 2022). A more perilous scenario is that some politicians inject malicious users into news recommendation systems to increase the exposure of their own news, thereby garnering more votes (Gunes et al., 2014).

In the early stages of recommendation systems, heuristic methods (Gunes et al., 2014; Si & Li, 2020) are proposed to explore the robustness of the recommendation system. For example, random attack involves interacting with random items to launch attacks, while bandwagon attack entails interacting with popular items. With the advancement of recommendation systems, their complexity and robustness have grown significantly (Xia, Wu, Yu, Kim, Rossi et al., 2023; Zhu, Ge, Gu, Zhao, & Lee, 2023). As a result, heuristic methods have proven inadequate in terms of effectiveness against the current situation (Rani, Kaur, Kumar, Ravi, Ghosh et al., 2023; You et al., 2023). An increasing number of researchers are turning their focus towards exploring a more diverse range of attack methods (Koren, Bell, & Volinsky, 2009; Li, Jin, Xu, & Tang, 2021; Liu et al., 2022; Zhang, Li, Ding, & Gao, 2020; Zügner, Akbarnejad, & Günnemann, 2018). For example, the approach proposed by Lin Lin, Chen, Li, Xiao, Li et al. (2020) utilizes Generative Adversarial Networks to generate fake user profiles, while Wu, Lian

* Corresponding author at: School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China.
E-mail addresses: ma_hao@cqu.edu.cn (H. Ma), gaomin@cqu.edu.cn (M. Gao), fengwei@cqu.edu.cn (F. Wei), zongwei@cqu.edu.cn (Z. Wang), jiangfeng@cqu.edu.cn (F. Jiang), zehua.zhao@sydney.edu.au (Z. Zhao), zyyang@cqu.edu.cn (Z. Yang).

https://doi.org/10.1016/j.eswa.2024.124476
Received 27 December 2023; Received in revised form 2 March 2024; Accepted 7 June 2024
Available online 10 June 2024

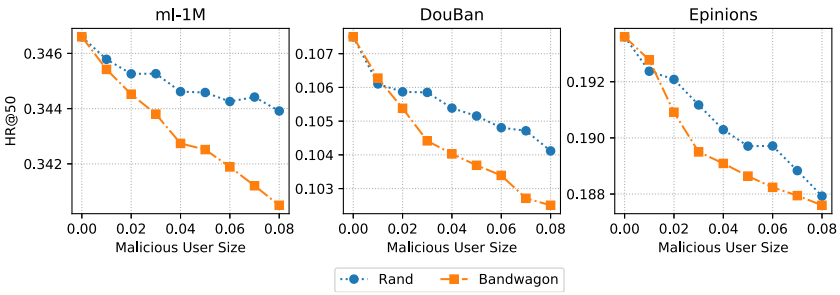


Fig. 1. Hit ratio under random attack and Bandwagon-Attack with different malicious user sizes on three datasets. HR@50 denotes the hit ratio of target items in the top 50 recommendation list. With the increase in the proportion of injected fake users, the hit rate of the recommendation system declines significantly, indicating that the attack does lead to a deterioration in the overall performance of the recommendation system.

et al. (2021) employ influence functions to guide the generation of fake users.

The majority of attack methods primarily emphasize the effectiveness of attacks which pertains to their ability to enhance the visibility of target items. However, they often overlook the stealthiness of attacks, referring to the attack's impact on the overall performance of the recommendation system. Crafting intentional malicious user interactions is designed to boost the recommendation rate of target items while perturbing the embeddings of these items. This perturbation can lead to an overall reduction in recommendation system performance and potentially raise the defender's awareness. To assess the impact of attack methods on the overall performance of recommendation systems, we conducted experiments on the four datasets with various attack methods. As illustrated in Fig. 1, we observed a decline in the overall performance of the recommendation system after being subjected to attacks, which indicates that there is indeed an issue with the stealthiness of the attack methods. Furthermore, as the number of injected fake users increases, the performance of the recommendation system continues to decline. This also indicates that excessively strong attacks can lead to a decrease in stealthiness. Balancing attack effectiveness and stealthiness simultaneously poses a challenging task.

To address the aforementioned issues, this paper introduces a novel attack framework InfoAtk (InfoNCE Attack) aimed at mitigating the impact of attacks on overall performance, thereby enhancing attack stealthiness while ensuring attack effectiveness. In our attack, we design an item promotion component to maximize the similarity between the target item embedding and other user embeddings, aiming to outrank the last item in users' recommendation lists and facilitate the inclusion of the target items into the recommendation lists of most users, thereby enhancing attack effectiveness. Meanwhile, we design an embedding consistency component for stealthiness; the focus is on reducing the degree of alteration in the overall performance of the recommendation system. A straightforward approach involves using the change in metrics e.g., hit ratio and ndcg, before and after the attack as an optimization objective. However, this method is impractical due to the lack of differentiability if the metrics are treated as an optimization objective, because the metrics are discrete. Therefore, drawing inspiration from the concept of contrastive learning (Hadsell, Chopra, & LeCun, 2006), we consider the attacked graph as an enhancement of the original graph. Our aim is to minimize a differentiable function that quantifies the disparities between item embeddings before and after the attack, with the goal of reducing the impact of the attack on the embeddings of users and items in the recommendation system, thereby decreasing the variation in the performance of the recommendation system, as the recommendation results of the system depend on the embedding vectors. In summary, the contributions of this paper are as follows:

- To the best of our knowledge, we are the first to explore the stealthiness perspective of attacks. By minimizing the disparities between item embeddings before and after attacks, we aim to design improved attack methods and provide new insights for the defense of recommendation systems.
- We design attacks on recommendation systems as a bi-level optimization problem and introduce an innovative poisoning attack algorithm that effectively addresses the requirements of attack stealthiness while ensuring effectiveness.
- We demonstrated the stealthiness and effectiveness of our method through experiments, providing a detailed analysis of the factors contributing to the effectiveness of the attacks.

2. Related work

2.1. Graph-based recommendation systems

With the rise of machine learning techniques, the integration of recommendation systems and machine learning has gained prominence. Among the most notable approaches is matrix factorization modeling (He, Zhang, Kan, & Chua, 2016; Koren et al., 2009; Liu et al., 2018; Zhang, Lu, Chen, Liu, & Ling, 2017). Its core concept involves decomposing the user–item interaction matrix into the product of two matrices: the user matrix and the item matrix. This allows each user and item to have their own embedding vector. The magnitude of the inner product of user–item embeddings indicates the level of interest a user has in a particular item. During iterative updates, it is crucial to ensure that these inner products align closely with observed interactions or scores.

However, the aforementioned recommendation methods do not fully capitalize on the graph information inherent in the user–item interaction matrix. To address this gap, the introduction of graph neural networks into recommendation systems has emerged. Graph neural networks, rooted in topology, utilize a message-passing mechanism for generating the node embeddings. They have demonstrated effectiveness in tasks such as node classification (Huo, Jin, Li, He, Yang et al., 2023; Kipf & Welling, 2017; Zeng, Li, Gao, Zhao, & Li, 2023), graph classification (Alkuhlani, Gad, Roushdy, & Salem, 2023; Benslimane, Azé, Bringay, Servajean, & Mollevi, 2023; Ying, You, Morris, Ren, Hamilton et al., 2018), and link prediction (Chung & Whang, 2023; Wang, Liu, & Shen, 2023; Zhang & Chen, 2018). By leveraging the aggregation capabilities of graph neural networks, a better fit for user and item embedding vectors can be achieved. NGCF (Wang, He, Wang, Feng, & Chua, 2019) is a recommendation system that leverages graph neural networks to model user–item interactions, enhancing accuracy by capturing collaborative filtering signals in a graph-based representation. Building upon NGCF, LightGCN (He, Deng, Wang, Li, Zhang et al., 2020) eliminates the linear transformation matrix in graph aggregation, demonstrating through experiments its enhanced performance. Due to noise in the user–item interaction graph, the bias in user–item embeddings learned from a single view by graph neural networks needs to be eliminated. To address this issue, SGL (Wu, Wang et al., 2021) employs self-supervised learning techniques to enhance the learning of superior user–item embedding vectors through comparisons across various views. On the other hand, SimGCL (Yu, Yin, Xia, Chen, Cui et al., 2022) abandons the graph augmentation mechanism and instead introduces

uniform noise into the embedding space to generate contrasting views. In this paper, we use LightGCN and SGL as the target recommendation system to evaluate the performance of the attack methods, since these two recommendation models serve as representatives of graph-based recommendation models, they encompass fundamental characteristics of graph neural networks such as message aggregation and feature transformation. Attacks targeted at them can similarly be applicable to other graph-based recommendation models.

2.2. Attacks on recommendation systems

In the early stages of research on attacks against recommendation systems, attack methods primarily relied on manually designed heuristic algorithms. For instance, Lam & Ried (Lam & Riedl, 2004) introduced two fundamental attack algorithms: random attack and average attack. Both approaches involve deducing the probability distribution of ratings from the dataset and then randomly assigning ratings to fake users based on this distribution. These methods fall under the category of "shilling attacks". However, shilling attacks mostly generate fake users based on heuristic statistical characteristics, limiting their attack effectiveness. A detailed analysis of various shilling attack methods on recommendation systems is provided (Turk & Bilge, 2019).

Concurrently, as the field of graph adversarial attacks evolved (Chen et al., 2022; Dai et al., 2018; Nguyen Thanh et al., 2023; Wu, Wu, Qi, Huang, & Xie, 2022; Yeh, Chen, Yang, Lee, Philip et al., 2023; Yu, Liu, Wu, Yu, Yu et al., 2023; Zhang, Yin, Chen, Huang, Nguyen et al., 2022), many attacks targeting recommendation systems borrowed ideas from these methods. Treating recommendation system attacks as graph adversarial problems, these approaches leverage the unique attributes of recommendation systems and employ machine learning techniques to generate fake users. This often yields better results compared to heuristic shilling attacks.

Zhang et al. (2021) formulated the attack framework as a bi-level optimization problem based on incomplete and noisy user behavior data. The upper layer defines the overall attack objective, while the lower layer specifies the learning goals of the recommendation model using original and injected data. SUI-Attack (Huang & Li, 2023) focuses on single-user injection attacks, modeling them as a node generation task on the user–item bipartite graph of the victim RS. It constructs fake user profiles by generating user features and connecting fake users to items. AutoAttack (Guo, Bai, & Deng, 2023), targeting specific user groups while minimizing the impact on non-target users, employs spectral clustering based on the graph Laplacian matrix to guide the generation of fake user structures. A-ra (Rong, He et al., 2022) employs Gaussian distribution to model the embedding vectors of unknown users, thereby addressing the issue of limited knowledge about users faced by black-box attackers.

With the popularity of GANs and their ability to generate realistic samples, Lin et al. (2020) introduced the AUSH framework, utilizing GANs to generate fake user profiles. The generator acts as the attacker, using existing genuine user profiles as templates to generate fake ones, while the discriminator functions as a defender, distinguishing between genuine and fake users. Similarly, TrialAttack (Wu, Lian et al., 2021) also uses GANs but introduces an influence module to estimate the influence of each fake user. Chen (Chen, Bao, Qi, You, Liu et al., 2024) proposes a poisoning attack method based on deep learning, which utilizes GANs for adversarial learning of the data distribution of real users, thereby generating high-quality fake user attack vectors. Using GAN networks, Wang (Wang, Liu, Wang et al., 2023) demonstrated the feasibility of poisoning attacks on mainstream SSL-based recommender schemes as well as commonly used datasets.

Considering the black-box nature of recommendation systems, some researchers use reinforcement learning-based attacks to cope with this scenario. PoisonRec (Song et al., 2020) employs a reinforcement learning framework where the attack agent actively injects fake data into the recommendation system. It improves its attack strategy based on available reward signals in a strict black-box setting. CopyAttack (Zhang et al., 2023) achieves its goal of promoting target items by replicating the profiles of real users from the source domain to the target domain. KGAttack (Chen et al., 2022) is also a reinforcement learning-based attack method. It enhances the generation of fake user profiles by additionally utilizing knowledge graphs, and employs these profiles to attack black-box recommendation systems.

However, the attacks introduced above ignore stealthiness when emphasizing effectiveness, i.e., the damage caused by the attack methods to the performance of the recommendation system. This kind of damage can trigger the defender's awareness. To address this issue, we propose a novel attack framework InfoAtk that attacks the recommendation system while considering stealth. We will elaborate on this method in the fourth section.

3. Preliminaries

As our attack are targeted towards graph-based recommendation systems, in this section, we will introduce the paradigm of graph-based recommendation systems and poisoning attacks on recommendation systems.

3.1. The paradigm of graph-based recommendation

Graph-based recommendation systems employ a message propagation mechanism within the user–item interaction graph to learn embeddings for users and items (Liu, Yang, & Li, 2021; Song, Zhou, Wang, & Lin, 2023). Specifically, let U and I represent the set of users and items, respectively. Each user and item, denoted as u and i, belong to these respective sets. Additionally, A is the matrix corresponding to the user–item interaction graph, where $A_{ui} = 1$ if user u has interacted with item i. Within the context of graph-based recommendation systems, node embeddings in one layer are constructed through the aggregation of neighboring node embeddings in the previous layer. This aggregation process hinges on the concept of message propagation, wherein information is iteratively combined from connected nodes to enhance the embeddings. Mathematically, the aggregation formula is expressed as follows:

$$e_u^{(k+1)} = AGG(e_u^{(k)}, \{e_i^{(k)} | e_i^{(k)} \in \mathcal{N}_u\}), \tag{1}$$

where $e_i^{(k)}$ is the k-layer embedding of the user or item, $\mathcal{N}_u$ denotes the set of neighboring nodes of user u, and AGG refers to the aggregation function.

Once the final embeddings are obtained, the model predictions can be derived based on the inner product of user and item embeddings:

$$p = e_u^T e_i. \tag{2}$$

3.2. The paradigm of poisoning attacks on recommendation systems

Poisoning attacks on recommendation systems refer to the deliberate injection of carefully crafted fake user profiles into the system. For push attacks, the objective is to increase the frequency of appearance of target items in the recommendation list; for nuke attacks, the aim is the opposite. In the context of graph adversarial attacks, attacks on the graph are formulated as a bi-level optimization problem (Zügner et al., 2018). Similarly, poisoning attacks on recommendation systems can also be described as a bi-level optimization problem. Initially, given the user set U and item set I, with the user–item interaction matrix represented as $A \in \{0,1\}^{|U|\times|I|}$, the attacker introduces a set of fake users U_F and generates corresponding toxic data $A^F \in \{0,1\}^{|U_F|\times|I|}$ based on this matrix. Consequently, the attack problem for the recommendation system can be expressed in the following manner:

$$\begin{aligned} &\min_{A^F} \mathcal{L}_{\text{atk}}([A; A^F], \theta_R^*), \\ &\theta_R^* = \arg\min_{\theta_R} \mathcal{L}_{\text{train}}([A; A^F], \theta_R). \end{aligned} \tag{3}$$

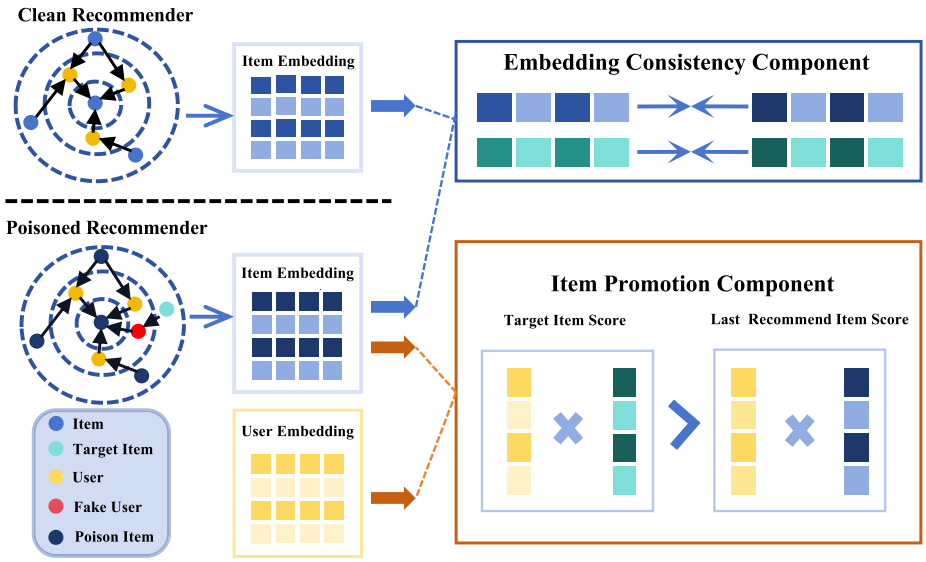


Fig. 2. The overall framework of InfoAtk. The embedding consistency component is employed to alleviate the disparities in item embeddings before and after the attack, whereas the item promotion component ensures the inclusion of target items in the recommendation lists of real users.

The equation involves R representing the recommendation system, θ denoting its parameter configuration, and $\mathcal{L}_{atk}$ and $\mathcal{L}_{train}$ respectively representing the attack loss function and the training loss function. These are determined by the specific attack and recommendation methods. In Formula (3), the upper-level optimization goal is the minimization of the attack loss function, while the lower-level optimization objective focuses on enhancing the performance of the recommendation system.

4. The proposed method

We propose a novel attack framework InfoAtk to achieve stronger attack stealthiness. In the attack framework, we design two components: a stealthiness component that reduces the embedding difference before and after the attack and an effectiveness component that makes more target items enter the recommendation list. The overall framework of the attack model InfoAtk is illustrated in Fig. 2. In this paper, our primary focus is on white-box attacks, as white-box attacks can be easily converted into black-box attacks by simply employing a surrogate model (Zhang et al., 2020).

4.1. Embedding consistency component

To ensure the stealthiness of InfoAtk, it is essential to ensure that the performance of the recommendation system does not undergo significant changes before and after the attack. Due to the non-differentiable nature of metrics, encompassing NDCG, recall, hit ratio, and others, they cannot be employed directly as the optimization objective (L_{atk}). Due to the recommendation performance being derived from the inner product of user and item embedding vectors, our approach does not directly optimize the recommendation performance; instead, we focus on optimizing the variation in item embeddings before and after the attack. Contrastive learning obtains better node representations by aligning the same node across different views, with the idea of contrastive learning, we view the post-attack interaction graph as an augmented form of the original graph. By controlling the difference in embeddings between the same nodes on the two graphs, we mitigate the overall performance variation. We capture the embeddings before and after the attack and utilize the InfoNCE loss function to quantify the extent of this variation. The specific formula is as follows:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{|I|}\sum_{i\in I}\log\frac{\exp s(e_i', e_i)}{\sum_{j\in I}\exp s(e_i', e_j)}, \tag{4}$$

where the symbol $s(\cdot)$ represents a similarity measurement function. In our study, we utilize the function $s(e_i', e_i) = e_i'^{T} e_i/\tau$ with τ as a hyperparameter. The symbols e_i and e_i' correspond to the embedding vectors of item i before and after the attack, respectively.

Our goal is to bring the pre-attack and post-attack item embeddings closer, thus ensuring that the recommendation performance of the underlying system remains relatively stable. Here, we refrained from considering the MSE (Mean Squared Error) loss function as a measure of similarity. This decision stems from the fact that the MSE loss function imposes stringent constraints on embedding similarity. In contrast, the InfoNCE loss places a greater emphasis on aligning positive samples (nodes from the same view), and opting for the InfoNCE loss function brings the advantage of narrowing the gap between the post-attack item embeddings and their pre-attack counterparts. This approach alleviates the constraints that the MSE loss function imposes on attack performance, thereby enhancing the effectiveness of the attack. We validate this point in the experimental section.

In conclusion, before launching an attack, we preserve the item embedding learned by the recommendation system as the original item embeddings. During the attack, using these original item embeddings as references, we recalibrate the recommendation system's item embedding vectors after each iteration. For each iteration, we compute the $\mathcal{L}_{\text{InfoNCE}}$ by evaluating the difference between the modified item embedding vector and the original item embedding vector using Formula (4). This loss is then employed to guide the attack method in generating more stealthy users.

4.2. Item promotion component

To ensure the effectiveness of InfoAtk, it is essential to maximize the appearance of the target items in users' recommendation lists. From an embedding perspective, this translates to maximizing the inner product between target items and users. Consequently, we employ the CW loss function (Rong, Ye et al., 2022) as a measure of the attack's efficacy. Specifically, we define $R(u)$ as the recommendation list for user u, I_{tar} as the set of target items, and e_u and e_i as the embeddings of user u and item i respectively. The CW loss function is defined as follows:

$$\mathcal{L}_{\text{CW}} = \frac{1}{|U|}\frac{1}{|I_{tar}|}\sum_{u\in U}\sum_{i\in I_{tar}}\min_{j\in R(u)\wedge j\notin I_{tar}}\{e_u^T e_j - e_u^T e_i\}. \tag{5}$$

The objective of the CW loss function is to ensure that the prediction value of the recommendation system for the target item is larger than

the last item in the user's recommendation list, This approach aims to facilitate the inclusion of the target item in the recommendation list.

To bolster the attack's effectiveness, during each iteration of the attack process, we retrieve the updated user and item embeddings from the modified recommendation system. Utilizing Formula (5), we then compute the $\mathcal{L}_{\text{CW}}$ specific to the ongoing attack, thereby providing a means to gauge the effectiveness of each individual attack iteration. It propels the attack method to generate users that possess greater attack potency.

4.3. Optimization

Based on the discussions presented earlier, we can determine the final form of the attack loss function as follows:

$$\mathcal{L}_{\text{Atk}} = (1-\beta)\mathcal{L}_{\text{CW}} + \beta\mathcal{L}_{\text{InfoNCE}}, \tag{6}$$

In Formula (6), both $\mathcal{L}_{\text{InfoNCE}}$ and $\mathcal{L}_{\text{CW}}$ serve as our optimization objectives, with equal importance assigned to each. If there is a substantial gap between them, one of the losses may become too small, leading to difficulties in training optimal parameters due to the vanishing gradients during backpropagation. To balance the sizes of the two loss functions, we allow parameter β to dynamically vary during the training process, setting its value to $\mathcal{L}_{\text{InfoNCE}}/(\mathcal{L}_{\text{CW}} + \mathcal{L}_{\text{InfoNCE}})$. It is crucial to emphasize that, in this context, β is constant within the loss function and does not participate in the backpropagation process. Our optimization variable is the interaction matrix A_F of fake users. Unfortunately, we cannot directly apply gradient descent to optimize $A_F \in \{0,1\}^{|U_F|\times|I|}$ due to its discrete nature. As a solution, we scale this matrix to the range $[0,1]$, transforming it into a continuous-valued matrix. Simultaneously, we employ PGD (Xu et al., 2019)(Projected Gradient Descent) to update A_F, ensuring that the updated values remain within the valid range. The iterative formula for PGD is as follows:

$$A_F^{(t+1)} = \text{Proj}_{[0,1]}(A_F^{(t)} - \mu\frac{\partial\mathcal{L}_{\text{Atk}}}{\partial A_F^{(t)}}), \tag{7}$$

where μ is the learning rate, $A_F^{(t)}$ represents the current interaction matrix at iteration t, $\text{Proj}_{[0,1]}$ is the projection onto the interval $[0,1]$, the underlying idea of this approach is that if the value after updating exceeds the valid range, the algorithm seeks the nearest point within the valid range to the updated value, so we have $\text{Proj}_{[0,1]}(x) = \arg\min_{\theta\in[0,1]}\frac{1}{2}\|\theta - x\|_2^2$, after simplifying the expression, we obtain:

$$\text{Proj}_{[0,1]}(x) = \begin{cases} 1 & x \geq 1 \\ x & 0 < x < 1 \\ 0 & x \leq 0 \end{cases}. \tag{8}$$

After obtaining the final best interaction matrix A^*, the subsequent step involves the process of discretizing this continuous matrix. For each fabricated user, we establish a predetermined interaction upper limit n. Subsequently, we perform a sampling procedure within A^*, where we designate the highest n values to be set as 1, while the remaining values are set to 0. This procedure essentially converts the continuous interactions into a discrete form, adhering to the specified upper limit for each fabricated user. The algorithm for InfoAtk is presented in Algorithm 1.

4.4. Analysis of time complexity

In order to explore the runtime of the proposed method, we conducted a time complexity analysis. In the algorithm, the outer loop iterates N times, while the inner loop iterates M times. Within the inner loop, we need to compute the gradient of each edge between fake users and items (regardless of whether this edge is connected). Assuming the time complexity of computing the gradient is gamma, the time complexity of this step is $O(\gamma|U_f||I|)$, where U_f is the set of fake users, and I is the set of items. In the projection gradient descent step, we need to project each edge from real numbers to binary integers based on its magnitude. Assuming the time complexity of the projection operation is θ, the time complexity of this step is $O(\theta|U_f||I|)$. Additionally, in the outer loop, apart from the inner loop, we also need to retrain the recommendation model. Assuming the time complexity of this step is ϕ, the overall time complexity of the algorithm is $O(N(M(\gamma+\theta)|U_f||I| + \phi))$.

Algorithm 1 Algorithm of InfoAtk

Input: Recommender model R, real user–item interaction matrix A, malicious user–item interaction matrix A_F, target item set I_{tar}, InfoNCE loss function weight β, the epoch of bi-level optimization N, learning rate μ and the epoch of PGD M.

Output: Best malicious user–item interaction matrix A_F^*.

$e_i^*, e_u = \arg\min_R \mathcal{L}_{\text{Train}}(A, R)$
for each $t = 1$ to N **do**
  $e_i, e_u = \arg\min_R \mathcal{L}_{\text{Train}}([A; A_F], R)$
  for each $j = 1$ to M **do**
    $\mathcal{L}_{\text{Atk}} = (1-\beta)\mathcal{L}_{\text{CW}}(e_i, e_u) + \beta\mathcal{L}_{\text{InfoNCE}}(e_i, e_i^*)$
    $A_F = \text{Proj}_{[0,1]}(A_F - \mu\frac{\partial\mathcal{L}_{\text{Atk}}}{\partial A_F})$
  end for
end for
$A_F^* = A_F$
return A_F^*

Table 1
Dataset statistics.

Dataset	#User	#Item	#Interaction	Sparsity
ML-100k	943	1,682	100,000	93.69%
ML-1M	6,040	3,952	1,000,000	95.81%
Douban	2,831	36,821	805,611	99.23%
Epinions	46,846	40,706	305,249	99.98%

5. Experiments

In this section, we address the following research questions. RQ1: How effective and stealthy is our proposed attack method? RQ2: What are the reasons for the performance of our proposed attack method? RQ3: What is the impact of hyperparameters on our attack method? RQ4: How is the long-term performance of the model over multiple attack iterations evaluated? RQ5: How does the scalability of our attack method? RQ6: How does our attack method perform when dealing with data of varying sparsity? RQ7: What is the impact of each loss on our attack method? RQ8: What is the probability of our attack method being detected? RQ9: How does the performance of our attack method in terms of runtime?

5.1. Experimental settings

Datasets. We use four datasets to validate our assertions, namely ML-1M, ML-100k, Douban (Zhao, Qian, & Xie, 2016) and Epinions. Detailed statistical information for these datasets is presented in Table 1.

Target Recommendation System. We select LightGCN (He et al., 2020) and SGL (Wu, Wang et al., 2021) to validate our findings. They are both recommendation systems based on GNNs (Graph Neural Networks) utilized in this approach. It employs normalized matrices for message aggregation and propagation, to generate more coherent user–item embeddings.

Attack Comparison Methods. To further illustrate the superiority of our attack method, we have selected multiple poisoning attack methods for comparative analysis. These include both classical heuristic-based attack methods and more recent poisoning attack methods proposed by researchers in the past few years:

- RandomAttack (Si & Li, 2020): Classic shilling attack methods involve creating fake users to interact with random items in order to manipulate the recommendation system.
- BandwagonAttack (Si & Li, 2020): Classic shilling attack methods involve boosting the ranking of target items by interacting with popular items.
- AUSH (Lin et al., 2020): This method utilizes GANs(Generative Adversarial Networks (Goodfellow et al., 2014)) to generate fake user interactions, thereby enhancing its stealthiness.
- RAPU-G (Zhang et al., 2021): This method assumes that the data obtained by the attacker is not complete and employs a probabilistic model to capture this incompleteness, thus enabling more accurate attacks.
- FedRecAttack (Rong, Ye et al., 2022): An attack method, with a focus on stealthiness, involves simulating user matrices to minimize the rating of the target item, making it as close as possible to the lowest rating among items in the recommendation list.
- GTA (Wang, Baisong et al., 2023): This approach encourages fake users to interact with items for which the recommender system predicts high ratings and the target item, thereby boosting the predicted ratings for the target item.
- PoisonRec (Song et al., 2020): PoisonRec employs a reinforcement learning framework, in which the adversarial agent proactively injects fake data into the recommendation system. It then refines its attack strategy by leveraging reward signals available in a strictly black-box setting.

Evaluation Metrics. In our experiments, we utilized two metrics, HR@K and NDCG@K (K=50), to evaluate the effectiveness of the attack. To be precise, the hit ratio signifies the proportion of user recommendation lists that include the target item. Its formula is as follows:

$$\mathrm{HR} = \frac{1}{|U|}\sum_{u\in U} \mathcal{I}\Big(\sum_{i\in I_{tar}} \mathcal{I}(i \in R(u)) \geq 1\Big), \tag{9}$$

where I_{tar} is the set of the target items, $R(u)$ is the recommended list of the user u, $\mathcal{I}(\cdot)$ denotes the indicator function, which yields a value of 1 if the input is true, and 0 otherwise. NDCG is a commonly used evaluation metric in the field of information retrieval. It is employed to assess the performance of ranking algorithms in recommendation systems or search engines. NDCG is the normalized counterpart of DCG (Discounted Cumulative Gain), whereby its value is obtained by dividing DCG by the ideal DCG value, IDCG. DCG computes the overall quality of a ranking outcome by assigning relevance scores to each item in the ranking result. Items with higher relevance scores receive higher ranks, while those with lower relevance scores receive lower ranks.

In the context of recommendation system attacks, the focus is primarily on the ranking of the target item. As such, the relevance score of the target item is set to 1, while all other items are assigned a relevance score of 0. Assuming that the recommended list $R(u)$ is sorted in descending order based on prediction values, the calculation of NDCG is expressed as follows:

$$\begin{aligned}
\mathrm{DCG} &= \sum_{k=1}^{|R(u)|} \frac{\mathbf{I}(i \in I_{tar})}{\log(k+1)}, \\
\mathrm{IDCG} &= \sum_{k=1}^{|I_{tar}|} \frac{1}{\log(k+1)}, \\
\mathrm{NDCG} &= \frac{\mathrm{DCG}}{\mathrm{IDCG}}.
\end{aligned} \tag{10}$$

We also utilized the metrics RecShift to quantify the reduction in the recommend list hit ratio of the recommendation system before and after attacks, this metric serves as an indicator of the stealthiness of the attack methods, its formula is as follows:

$$\mathrm{RecShift} = \frac{\mathrm{HR}_{\mathrm{attack}} - \mathrm{HR}}{\mathrm{HR}}, \tag{11}$$

where HR and $\mathrm{HR}_{\mathrm{attack}}$ represent the hit ratio of the recommendation system before and after the attack. To assess the impact of the attack, we introduced the generated poisoned data into the initial clean dataset, creating a training dataset for the recommendation system after the attack.

Parameters Settings. The batch size, embedding size, and number of layers for LightGCN and SGL are set to 256, 32, and 2, respectively. Additionally, the contrastive loss function ratio in SGL is set to 0.1. For the attack scenarios, we established the ratio of fake users as 0.01, 0.03, and 0.05. Each fake user was attributed a maximum interaction count equivalent to the average interaction count of users in the dataset. As part of the attack strategy, we identified 5 items with interaction counts falling within the lowest 20% of the dataset and considered them as the target items for the attack.

5.2. *Effectiveness and stealthiness* (*RQ1*)

We initiate our experimentation by evaluating the performance of various attack methods across four distinct datasets. The results are presented in Tables 2 and 3 (OOM means this method is out of memory on the dataset), wherein we have highlighted the most optimal performance data within each experimental set using bold formatting and designated the second-best performance data with an underline. Drawing insights from the experimental outcomes, we can arrive at the following conclusions:

- Regarding the aspect of stealthiness, in the majority of cases, our approach exhibits the least impact on the hit ratio of the recommendation system before and after attacks. This effectively demonstrates the superiority of our attack method in terms of stealthiness. However, the performance of other attack methods in terms of stealthiness is not flawless. For instance, in LightGCN, within the ML-1M dataset, our method achieves optimal stealthiness with a RecShift of −0.0009 when the proportion of fake users is 0.05.
- From the perspective of attack effectiveness, our attack method has the capability to maintain optimal or near-optimal attack performance. This highlights that our attack method retains significant attack potency while achieving a strong level of stealthiness. Especially on the larger datasets, such as DouBan and Epinions, the performance of our attack method is outstanding.
- Across various datasets and recommendation systems, we observed consistently favorable performance of InfoAtk. This observation suggests that the proposed method holds relative utility across multiple practical scenarios, indicating its potential applicability in diverse environments.
- Through comparisons across different datasets, we observed that the performance of attack methods is more significantly influenced by the number of fake users in datasets with higher sparsity. For instance, in the ml-1M dataset, our attack method achieved HR@50 values of 0.0733, 0.0958, and 0.1145 for LightGCN, while in the Epinions dataset, the corresponding HR@50 values were 0.0729, 0.1511, and 0.2007. The increase in performance is notably higher in the Epinions dataset compared to the results from the ml-1M dataset. This effect is observed across other methods as well. This phenomenon can be attributed to the higher sensitivity of each item to interactions in sparser datasets. Consequently, these items are more vulnerable to successful attacks from the perspective of attackers.

Due to the similar nature and structure of LightGCN and SGL, for the sake of brevity, we will only consider LightGCN in the subsequent experiments.

Table 2
The effectiveness and stealthiness of attack methods on LightGCN.

Dataset	AttackMethod	FakeUserRatio								
		0.01			0.03			0.05		
		HR@50	NDCG@50	**RecShift**	HR@50	NDCG@50	**RecShift**	HR@50	NDCG@50	**RecShift**
ML-100K	RandomAttack	0.5453	0.0689	−0.0523	0.5478	0.0751	−0.0597	0.5541	0.0772	−0.0623
	Bandwagon	0.5556	0.0712	−0.0476	0.5634	0.0724	−0.0532	0.5638	0.0779	−0.0553
	AUSH	0.5612	0.0723	−0.0523	0.5732	0.0727	−0.0604	0.5956	**0.0832**	−0.0608
	FedRecAttack	0.5720	0.0761	−0.0511	0.5804	**0.0784**	−0.0543	0.5947	0.0798	−0.0588
	RAPU-G	0.5384	0.0662	−0.0498	**0.6124**	0.0714	−0.0627	0.5981	0.0783	−0.0587
	GTA	0.5825	0.0749	−0.0623	0.5865	0.0756	−0.0547	0.5870	0.0745	−0.0670
	PoisonRec	0.5515	0.0716	−0.0465	0.5577	0.0728	−0.0480	0.5682	0.0728	−0.0568
	InfoAtk	**0.5962**	**0.0769**	**−0.0424**	0.6010	0.0775	**−0.0453**	**0.6124**	0.0787	**−0.0504**
ML-1M	RandomAttack	0.0446	0.0035	−0.0017	0.0600	0.0055	−0.0063	0.0892	0.0124	−0.0067
	Bandwagon	0.0630	0.0053	−0.0055	0.0595	0.0047	−0.0092	0.0820	0.0074	−0.0131
	AUSH	0.0707	0.0100	−0.0020	0.0764	0.0168	−0.0024	0.0949	0.0262	−0.0061
	FedRecAttack	0.0572	0.0096	−0.0063	**0.1049**	0.0221	−0.0069	**0.1479**	**0.0452**	−0.0083
	RAPU-G	0.0485	0.0088	−0.0064	0.0922	**0.0287**	−0.0066	0.1043	0.0434	−0.0085
	GTA	0.0605	0.0081	−0.0031	0.0824	0.0199	−0.0046	0.1109	0.0346	−0.0056
	PoisonRec	0.0651	0.0086	−0.0025	0.0861	0.0274	−0.0033	0.0806	0.0417	−0.0045
	InfoAtk	**0.0733**	**0.0101**	**−0.0009**	0.0958	0.0167	**−0.0022**	0.1189	0.0304	**−0.0034**
Douban	RandomAttack	0.0024	0.0002	−0.0084	0.0038	0.0003	−0.0136	0.0047	0.0005	−0.0195
	Bandwagon	0.0024	0.0002	−0.0116	0.0055	0.0005	−0.0249	0.0064	0.0007	−0.0405
	AUSH	**0.0093**	0.0012	−0.0096	0.0288	0.0029	−0.0126	**0.0465**	0.0056	−0.0141
	FedRecAttack	0.0056	0.0011	−0.0067	0.0175	0.0044	−0.0145	0.0367	0.0130	−0.0150
	RAPU-G	OOM	–	–	–	–	–	–	–	–
	GTA	0.0089	**0.0013**	−0.0098	0.0288	0.0056	−0.0109	0.0402	0.0074	−0.0130
	PoisonRec	0.0076	0.0011	−0.0103	0.0301	0.0060	−0.0113	0.0426	0.0096	−0.0149
	InfoAtk	0.0087	0.0008	**−0.0064**	**0.0314**	**0.0113**	**−0.0088**	0.0445	**0.0143**	**−0.0103**
Epinions	RandomAttack	0.0602	0.0096	−0.0116	0.1094	0.0182	−0.0154	0.1480	0.0248	−0.0196
	Bandwagon	0.0413	0.0129	−0.0057	0.0889	0.0298	−0.0201	0.1335	0.0496	−0.0255
	AUSH	0.0125	0.0087	−0.0090	0.0312	0.0236	−0.0136	0.0589	0.0411	−0.0158
	FedRecAttack	0.0628	0.0308	−0.0024	0.1281	0.0627	−0.0112	0.1971	0.1061	−0.0155
	RAPU-G	OOM	–	–	–	–	–	–	–	–
	GTA	0.0111	0.0093	−0.0098	0.0453	0.0232	−0.0154	0.0743	0.0490	−0.0179
	PoisonRec	0.0205	0.0101	−0.0074	0.0494	0.0292	−0.0150	0.0654	0.0396	−0.0184
	InfoAtk	**0.0729**	**0.0353**	**−0.0013**	**0.1511**	**0.0775**	**−0.0024**	**0.2043**	**0.1067**	**−0.0056**

5.3. Embedding analysis of the attack method (*RQ2*)

By comparing the changes in user and item embeddings before and after an attack, we aim to analyze and elucidate the reasons behind the performance of recommendation system attacks. We use the t-SNE method to visualize the high-dimensional user and item embedding vectors in a two-dimensional space. We illustrate this approach using the example of attacking the LightGCN model on the ML-100k dataset, with non-popular items as target items.

Initially, prior to the attack (as depicted in Fig. 3), user embedding vectors are represented in blue, popular item embeddings in orange, target item embeddings in red. It is observed that user embedding vectors are generally closer to popular items and farther from target items. This alignment influences the higher recommendation probabilities for popular items, attributed to their increased interaction frequency, causing their embeddings to be pulled closer to the user cluster. Conversely, the embeddings of non-popular items are distanced from the user cluster.

Next, we examine the distribution of embedding vectors after the attack. For clarity, we retain user embedding vectors, embeddings of popular items, and embeddings of target items in Fig. 3. Following the completion of attacks across all methods, it becomes evident that the embeddings of target items have moved closer to both popular item embeddings and user embeddings. This shift accounts for the increased occurrence of target items in the recommendation list post-attack.

Furthermore, we observed that even after the attack, there are still some target items whose embeddings remain distant from the user cluster. This implies that post-attack, their probability of being recommended remains low. These hard-to-attack items can result in the failure of the attack. For example, the attack on SGL in the ml-1M dataset may fail due to the difficulty of bringing the target items closer to the user cluster.

5.4. Hyperparameter analysis (*RQ3*)

To investigate the RQ3, we examined the impact of injecting fake users with varying proportions on attack performance. The examination, as illustrated in Fig. 4 utilizing the ML-1M dataset, indicates that an escalation in the proportion of users leads to an augmented attack performance. However, concurrently, RecShift demonstrates a continuous ascent in tandem with the escalating user proportion. This further emphasizes the significant influence of the fake user ratio on the overall performance variation of the recommendation system.

Meanwhile, we will further explore the impact of another crucial parameter, namely the influence of β in Formula (6). We conducted additional tests on the performance of the attack method under different values of β, specifically attacking LightGCN on both ML-100k and ML-1M datasets, differing from the optimization method in the paper, in this experiment, we set the value of β as a fixed constant. As depicted in Fig. 5, on the ML-1M dataset, with the increase in β, HR@50, NDCG@50, and RecShift of the attack method all decrease. The decline in the former implies a reduction in the attack intensity, while the decrease in the latter signifies an increase in the stealthiness of the attack. We can draw a conclusion that the stealthiness and attack intensity are mutually exclusive, aligning with common sense. Excessive attack strength can disrupt the original recommendation capabilities of the recommendation system, generating biased recommendations and consequently reducing overall recommendation performance.

To analyze the impact of different numbers of target items on attack performance, we conducted attack experiments on LightGCN using the ml-1M dataset with varying numbers of target items, and we compared our approach to two other baseline methods with suboptimal performance. We examined the attack performance for target item numbers ranging from 1 to 10, as shown in Fig. 6. Both HitRatio

Table 3
The effectiveness and stealthiness of attack methods on SGL.

Dataset	AttackMethod	FakeUserRatio								
		0.01			0.03			0.05		
		HR@50	NDCG@50	**RecShift**	HR@50	NDCG@50	**RecShift**	HR@50	NDCG@50	**RecShift**
ML-100K	RandomAttack	0.6429	0.1115	−0.0072	0.6576	0.1219	−0.0093	0.6681	0.1270	−0.0115
	Bandwagon	0.6471	0.1177	−0.0115	0.6613	0.1241	−0.0127	0.6720	0.1249	−0.0140
	AUSH	0.6641	0.1275	−0.0118	0.6711	0.1308	−0.0129	0.6742	0.1329	−0.0134
	FedRecAttack	0.6623	0.1273	−0.0092	0.6741	0.1291	−0.0131	0.6820	0.1360	−0.0145
	RAPU-G	0.6724	**0.1355**	−0.0096	0.6841	0.1391	−0.0117	0.6910	0.1427	−0.0159
	GTA	0.6631	0.1246	−0.0104	0.6718	0.1286	−0.0144	0.6834	0.1294	−0.0152
	PoisonRec	0.6690	0.1286	−0.0112	0.6802	0.1311	−0.0123	0.6873	0.1360	−0.0142
	InfoAtk	**0.6741**	0.1347	**−0.0041**	**0.6903**	**0.1418**	**−0.0085**	**0.7032**	**0.1434**	**−0.0094**
ML-1M	RandomAttack	0.0420	0.0034	−0.0032	0.0527	0.0042	−0.0065	0.0562	0.0051	−0.0072
	Bandwagon	0.0472	0.0036	−0.0029	0.0653	0.0052	−0.0050	0.0866	0.0078	−0.0067
	AUSH	0.0551	0.0059	−0.0047	0.0726	0.0119	−0.0082	0.0940	0.0170	−0.0110
	FedRecAttack	0.0518	0.0063	−0.0034	0.0671	0.0202	**−0.0049**	0.1034	**0.0415**	−0.0072
	RAPU-G	0.0533	0.0068	−0.0044	**0.0953**	0.0228	−0.0073	**0.1103**	0.0372	−0.0152
	GTA	**0.0573**	0.0073	−0.0035	0.0775	0.0167	−0.0086	0.0922	0.0184	−0.0095
	PoisonRec	0.0508	**0.0074**	−0.0043	0.0768	**0.0239**	−0.0057	0.1019	0.0330	−0.0089
	InfoAtk	0.0513	0.0051	**−0.0028**	0.0841	0.0218	−0.0057	0.1041	0.0413	**−0.0066**
DouBan	RandomAttack	0.0007	0.0000	−0.0126	0.0021	0.0002	−0.0240	0.0038	0.0009	−0.0261
	Bandwagon	0.0056	0.0005	−0.0157	0.0110	0.0009	−0.0236	0.0173	0.0029	−0.0253
	AUSH	0.0119	0.0025	−0.0100	0.0278	0.0048	−0.0101	0.0350	0.0048	−0.0135
	FedRecAttack	0.0024	0.0004	−0.0051	0.0065	0.0015	−0.0095	0.0101	0.0033	−0.0143
	RAPU-G	OOM	–	–	–	–	–	–	–	–
	GTA	0.0104	0.0025	−0.0103	**0.0308**	0.0059	−0.0193	**0.0484**	0.0105	−0.0271
	PoisonRec	0.0114	0.0022	−0.0069	0.0253	**0.0063**	−0.0154	0.0414	0.0093	−0.0228
	InfoAtk	**0.0129**	**0.0037**	**−0.0015**	0.0285	0.0035	**−0.0084**	0.0474	**0.0111**	**−0.0111**
Epinions	RandomAttack	0.1284	0.0629	−0.0049	0.1408	0.0843	−0.0064	0.1620	0.1154	−0.0076
	Bandwagon	0.1006	0.0276	−0.0048	0.2121	0.0651	−0.0253	0.2577	0.0853	−0.0259
	AUSH	0.0667	0.0312	**−0.0009**	0.1135	0.0648	−0.0037	0.1591	0.0973	−0.0063
	FedRecAttack	0.1323	0.0749	−0.0049	0.2031	0.1109	−0.0072	0.2461	0.1614	−0.0082
	RAPU-G	OOM	–	–	–	–	–	–	–	–
	GTA	0.0588	0.0266	−0.0048	0.1023	0.0589	−0.0065	0.1657	0.1098	−0.0183
	PoisonRec	0.0941	0.0474	−0.0043	0.1510	0.0910	−0.0050	0.2037	0.1134	−0.0071
	InfoAtk	**0.1720**	**0.1029**	−0.0014	**0.2473**	**0.1629**	**−0.0016**	**0.2682**	**0.1842**	**−0.0023**

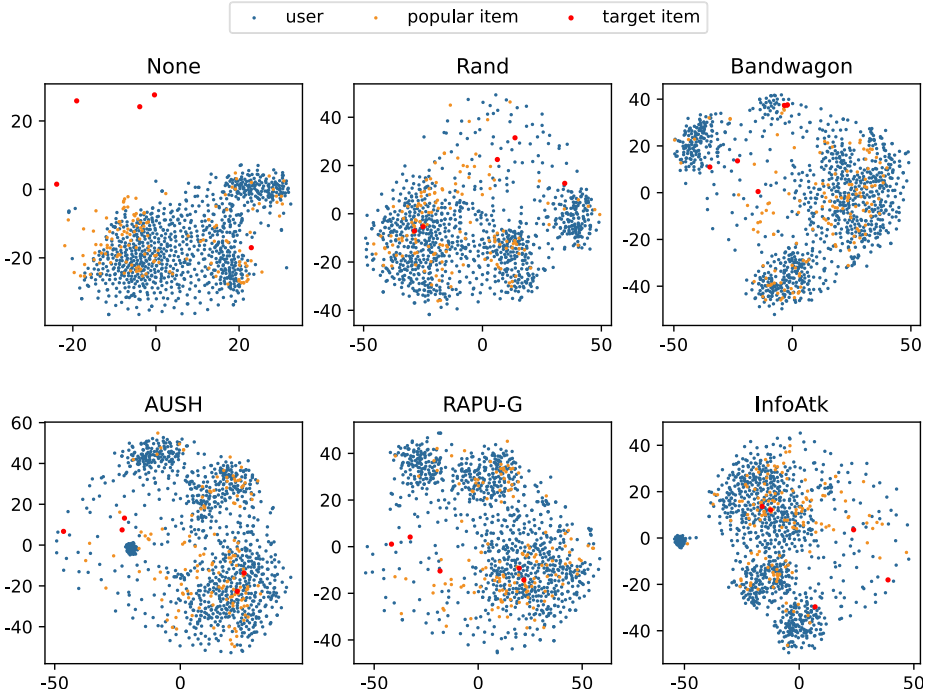


Fig. 3. Visualization outcomes of user embedding vectors and target embedding vectors before and after the attack. In the diagram, users are depicted as blue dots. To distinguish items further, we have divided them into popular items (orange dots) and target items (red dots). The aim is to investigate the relationship between target item embeddings, user embeddings, and popular item embeddings before and after the attack.

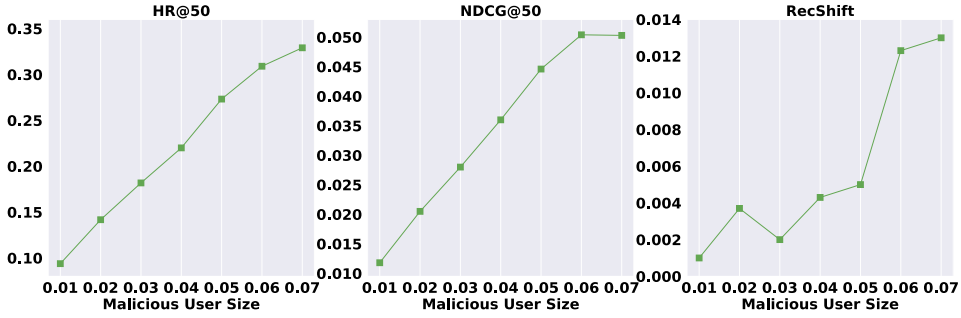


Fig. 4. Hyperparameter analysis of malicious user size on ML-1M.

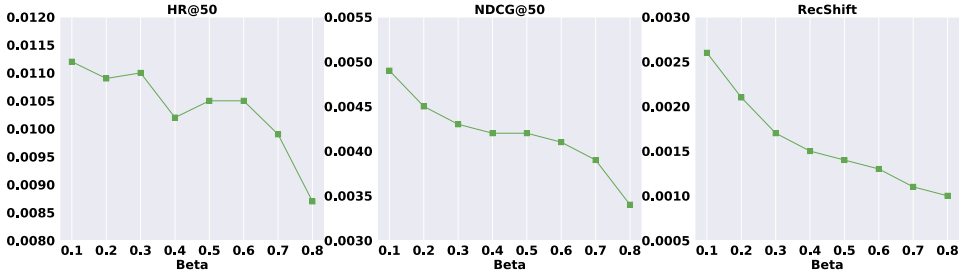


Fig. 5. Hyperparameter analysis of β on ML-1M.

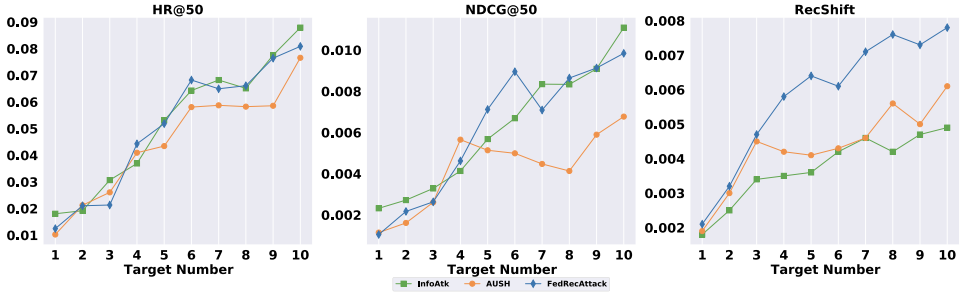


Fig. 6. Hyperparameter analysis of target number on ML-1M.

and NDCG increase as the number of target items grows, which is expected since a greater number of target items makes the attack easier. Additionally, RecShift exhibits a similar trend in the graph, indicating that increasing the number of target items leads to a decrease in attack stealthiness. Compared to the other two baseline methods, our approach consistently maintains optimal stealthiness while ensuring attack effectiveness across varying numbers of target items.

5.5. *Long-term performance* (*RQ4*)

To analyze the long-term performance of the model over multiple attack iterations, we conducted iterative attacks on LightGCN using the ml-1M dataset, and we also compared our approach to two other baseline methods with suboptimal performance. The experimental results depicted in Fig. 7 indicate that as the number of attack iterations increases, the performance of the attack steadily improves. However, the performance of the recommendation model consistently declines. This observation suggests that as the attack injects more fake users continuously, the stealthiness of the attack also diminishes. Our attack method consistently maintains the highest level of stealthiness, while also retaining optimal or near-optimal attack effectiveness.

Table 4
Attack performance with different recommendation systems.

RSModel	HR@50	NDCG@50	RecShift
LightGCN	0.0733	0.0101	−0.0009
SGL	0.0513	0.0051	−0.0028
NGCF	0.0615	0.0084	−0.0022
SimGCL	0.1183	0.0159	−0.0051

5.6. *Scalability analysis* (*RQ5*)

To assess the scalability of our attack method, we conducted experiments on two additional recommendation models, NGCF (Wang et al., 2019) and SimGCL (Yu et al., 2022), in addition to LightGCN and SGL. We performed attack experiments with a fake user ratio of 0.01 on these two recommendation models using the ml-1M dataset. The results are presented in Table 4, indicating that our attack method achieves similar performance on these two recommendation models as it does on LightGCN and SGL. This further demonstrates the adaptability of our attack method across various scenarios.

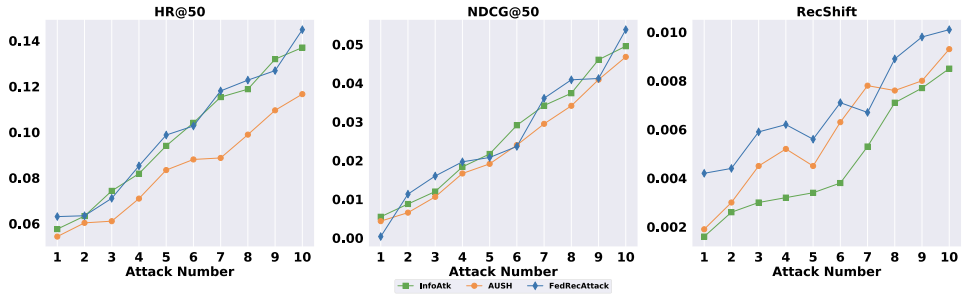


Fig. 7. Multiple attack on ML-1M.

Table 5
Attacks at various levels of sparsity.

Method	RemovalRatio											
	20%			40%			60%			80%		
	HR@50	NDCG@50	RecShift	HR@50	NDCG@50	RecShift	HR@50	NDCG@50	RecShift	HR@50	NDCG@50	RecShift
BandwagonAttack	0.0563	0.0048	−0.0047	0.0543	0.0053	−0.0070	0.0494	0.0055	−0.0115	0.0595	0.0092	−0.0126
AUSH	0.0569	0.0073	−0.0035	0.0702	0.0119	−0.0053	0.0530	0.0107	−0.0067	0.0530	0.0156	−0.0086
FedRecAttack	0.0613	0.0079	−0.0059	**0.0707**	**0.0166**	−0.0065	0.0613	0.0136	−0.0087	0.0722	0.0233	−0.0092
InfoAtk	**0.0645**	**0.0081**	**−0.0025**	0.0705	0.0124	**−0.0037**	**0.0691**	**0.0150**	**−0.0063**	**0.0750**	**0.0265**	**−0.0084**

Table 6
Performance with different loss functions.

Method	HR@50	NDCG@50	RecShift
None	0.8630	0.0875	–
InfoAtk + $\mathcal{L}_{CW}$	**0.9094**	0.0908	−0.0140
InfoAtk + $\mathcal{L}_{InfoNCE}$	0.8583	0.0823	−0.0042
InfoAtk + $\mathcal{L}_{CW}$ + $\mathcal{L}_{InfoNCE}$	0.8796	**0.0950**	**−0.0040**

5.7. Experiments with different sparsity (RQ6)

To investigate how the performance of our attack method varies with different levels of data sparsity, we targeted LightGCN and selected the ml-1M dataset. By randomly removing interaction data, we generated datasets with varying degrees of sparsity: 20%, 40%, 60%, and 80%. We also chose several attack methods for comparison in our experiments. The results, as shown in Table 5, indicate that our attack method performs well across different levels of sparsity. Particularly, we observed that as the dataset becomes sparser, the impact of the attack on recommendation performance becomes more severe. This further emphasizes the challenge of maintaining stealthiness in attacks on highly sparse datasets.

5.8. Ablation study (RQ7)

To investigate the significance of various components in our proposed methodology, we conducted ablation experiments. Specifically, as demonstrated in Table 6, we attacked LightGCN on ml-100k using a masking approach, excluding different loss functions to evaluate the individual impact of each component on the experimental results. When employing only the CW loss function, our attack method achieved optimal performance with a hit ratio of 0.9094. However, it exhibited the poorest performance in terms of stealthiness. It was only through the combined utilization of both the CW loss function and the InfoNCE loss function that we could attain the best results in terms of stealthiness while simultaneously achieving favorable effectiveness performance. It is evident from the results that upon removing the CW loss, the method's performance experienced a decline. Moreover, when the InfoNCE loss was omitted, the stability of the method decreased. This substantiates the functionality and importance of each loss term within our approach.

Furthermore, we straightforwardly substituted the similarity metric function of the Embedding Consistency Component with the MSE loss function. We then compared the performance of our method across different user ratios on the ML-1M dataset, as depicted in Table 7. It is evident from the results that the effectiveness of the MSE loss function as a similarity metric is notably inferior to the InfoNCE loss function employed in our approach. This further underscores the superiority of InfoNCE when used as a similarity metric function.

5.9. Attack detection (RQ8)

To assess the likelihood of detecting fake users injected by our method, we conducted attack detection experiments on the ML-1M dataset using FAP (Zhang, Tan, Zhang, Liu, Chua et al., 2015). The parameter k for this method was set to 100 for comparison with other methods, as shown in Table 8. Our method demonstrates the lowest detection rate among attack methods, with an F1 score of 0.1 in the detection method. Additionally, random attack yields suboptimal results, which can be attributed to the incorporation of a random selection process in the generation of fake user profiles. This randomness in the generation process results in fake user profiles lacking distinct characteristics. Consequently, the lack of discernible features makes it challenging for detection methods to effectively extract meaningful patterns during the identification process.

5.10. Runtime analysis (RQ9)

To analyze the runtime performance of our proposed method, experiments were conducted on the ml-1M dataset with the attack setting of LightGCN. The runtime of various attack methods was recorded on a server equipped with a GTX3090Ti graphics card and 24 GB of memory. As shown in Table 9, our method exhibited favorable runtime compared to most attack methods. It is noteworthy that heuristic methods such as random attack and bandwagon attack required the least amount of time, as they are based on simple rules for implementing attacks. On the other hand, GTA incurred the longest runtime because it involves iterative training of agent models, which prolongs the attack process.

Table 7
Performance with MSE loss function.

Method	fakerUserRatio								
	0.01			0.03			0.05		
	HR@50	NDCG@50	RecShift	HR@50	NDCG@50	RecShift	HR@50	NDCG@50	RecShift
InfoAtk With MSE	0.0603	0.0052	−0.0026	0.0772	0.0125	**−0.0020**	**0.1164**	0.0204	−0.0039
InfoAtk	**0.0733**	**0.0101**	**−0.0009**	**0.0958**	**0.0167**	−0.0022	0.1145	**0.0267**	**−0.0009**

Table 8
The FAP method detects fake users.

Attack method	P@100	R@100	F1
RandomAttack	0.100	0.167	0.125
Bandwagon	0.230	0.383	0.287
AUSH	0.200	0.333	0.250
RAPU-G	0.150	0.250	0.187
FedRecAttack	0.190	0.317	0.237
GTA	0.180	0.300	0.225
InfoAtk	**0.080**	**0.133**	**0.100**

Table 9
The runtime of various methods.

Attack method	RunTime/s
RandomAttack	5.6
Bandwagon	5.7
AUSH	178.2
RAPU-G	3473.9
FedRecAttack	585.0
GTA	14 723.4
PoisonRec	1078.6
InfoAttack	234.7

6. Conclusion and future work

In this work, we have analyzed a key aspect of recommendation system attacks: stealthiness. Taking this aspect into consideration, we propose a novel recommendation system attack algorithm. This algorithm involves injecting a limited number of fake users, enhancing the attack's effectiveness through the use of the CW loss function, ensuring stealthiness with the InfoNCE loss function, and ultimately optimizing the fake user interaction matrix. Finally, we validate our approach on four real-world datasets. Moving forward, our focus will be directed towards exploring improved stealthiness functions and optimization algorithms by more intricate architecture. We consider using influence functions to model the impact of each interaction of fake users on the stealthiness and effectiveness, thereby optimizing attack loss more efficiently. Our experimental results indicate that, under the same attack intensity, compared to the baseline methods, our approach ensures stronger stealthiness. This demonstrates the advantage of our method in terms of stealthiness. Furthermore, based on the conclusions drawn from our experimental findings regarding embedding distributions, it was observed that there are still some target items that did not effectively draw closer to the user cluster. In the future, we will also investigate this category of challenging-to-attack items. We will consider categorizing items based on the level of difficulty in attacking them and allocate higher weights to those difficult-to-attack items during the attack training process to emphasize their significance in the attack.

CRediT authorship contribution statement

Hao Ma: Conceptualization, Validation, Writing – review & editing. **Min Gao:** Investigation, Writing – original draft, Writing – review & editing. **Feng Wei:** Validation, Resources, Writing – review & editing. **Zongwei Wang:** Validation, Writing – review & editing. **Feng Jiang:** Project administration. **Zehua Zhao:** Supervision, Project administration. **Zhengyi Yang:** Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (62176028), the Major Project for Technology Innovation and Application Development of Chongqing Science & Technology Commission, China (CSTB2022TIAD-STX0006), and the Scientific and Technological Research Program of Chongqing Municipal Education Commission, China (KJZD-K202204402 and KJZD-K202304401).

References

Alkuhlani, A., Gad, W., Roushdy, M., & Salem, A.-B. M. (2023). GNNGLY: Graph neural networks for Glycan classification. *IEEE Access*.

Benslimane, S., Azé, J., Bringay, S., Servajean, M., & Mollevi, C. (2023). A text and GNN based controversy detection method on social media. *World Wide Web*, *26*(2), 799–825. http://dx.doi.org/10.1007/s11280-022-01116-0.

Chen, Z., Bao, T., Qi, W., You, D., Liu, L., & Shen, L. (2024). Poisoning qos-aware cloud API recommender system with generative adversarial network attack. *Expert Systems with Applications*, *238*(Part B), Article 121630. http://dx.doi.org/10.1016/J.ESWA.2023.121630.

Chen, J., Fan, W., Zhu, G., Zhao, X., Yuan, C., Li, Q., & Huang, Y. (2022). Knowledge-enhanced black-box attacks for recommendations. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 108–117). Washington DC USA: ACM, http://dx.doi.org/10.1145/3534678.3539359.

Chung, C., & Whang, J. J. (2023). Learning representations of Bi-level knowledge graphs for reasoning beyond link prediction. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 37*, (no. 4), (pp. 4208–4216). http://dx.doi.org/10.1609/aaai.v37i4.25538, arXiv:2302.02601.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. In *International conference on machine learning* (pp. 1115–1124). PMLR.

Dinnissen, K. (2022). Improving fairness and transparency for artists in music recommender systems. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. Madrid Spain: ACM, http://dx.doi.org/10.1145/3477495.3531681, 3498–3498.

Fan, X., Hu, Y., Zheng, Z., Wang, Y., Brézillon, P., & Chen, W. (2021). CASR-TSE: Context-aware web services recommendation for modeling weighted temporal-spatial effectiveness. *IEEE Transactions on Services Computing*, *14*(1), 58–70. http://dx.doi.org/10.1109/TSC.2017.2782793.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*: *vol. 27*.

Gunes, I., Kaleli, C., Bilge, A., & Polat, H. (2014). Shilling attacks against recommender systems: A comprehensive survey. *Artificial Intelligence Review*, *42*(4), 767–799. http://dx.doi.org/10.1007/s10462-012-9364-9.

Guo, S., Bai, T., & Deng, W. (2023). Targeted shilling attacks on GNN-based recommender systems. In *Proceedings of the 32nd ACM international conference on information and knowledge management* (pp. 649–658). ACM, http://dx.doi.org/10.1145/3583780.3615073.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition* (pp. 1735–1742). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2006.100.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 639–648). Virtual Event China: ACM, http://dx.doi.org/10.1145/3397271.3401063.

He, X., Zhang, H., Kan, M.-Y., & Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 549–558). Pisa Italy: ACM, http://dx.doi.org/10.1145/2911451.2911489.

Huang, C., & Li, H. (2023). Single-user injection for invisible shilling attack against recommender systems. In *Proceedings of the 32nd ACM international conference on information and knowledge management* (pp. 864–873). ACM, http://dx.doi.org/10.1145/3583780.3615062.

Huo, C., Jin, D., Li, Y., He, D., Yang, Y.-B., & Wu, L. (2023). T2-Gnn: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation. In *Proceedings of the AAAI conference on artificial intelligence: vol. 37*, (pp. 4339–4346).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. arXiv:1609.02907.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Lam, S. K., & Riedl, J. (2004). Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on world wide web* (pp. 393–402). New York NY USA: ACM, http://dx.doi.org/10.1145/988672.988726.

Lee, H., Hwang, D., Kim, H., Lee, B., & Choo, J. (2022). DraftRec: Personalized draft recommendation for winning in multi-player online battle arena games. In *Proceedings of the ACM web conference 2022* (pp. 3428–3439). Virtual Event, Lyon France: ACM, http://dx.doi.org/10.1145/3485447.3512278.

Li, Y., Jin, W., Xu, H., & Tang, J. (2021). Deeprobust: A platform for adversarial attacks and defenses. In *Proceedings of the AAAI conference on artificial intelligence: vol. 35*, (pp. 16078–16080).

Lin, C., Chen, S., Li, H., Xiao, Y., Li, L., & Yang, Q. (2020). Attacking recommender systems with augmented user profiles. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 855–864). Virtual Event Ireland: ACM, http://dx.doi.org/10.1145/3340531.3411884.

Liu, Y., Ma, J., Xie, Y., Yang, X., Tao, X., Peng, L., & Gao, W. (2022). Contrastive predictive coding with transformer for video representation learning. *Neurocomputing*, *482*, 154–162.

Liu, H., Yang, B., & Li, D. (2021). Graph collaborative filtering based on dual-message propagation mechanism. *IEEE Transactions on Cybernetics*, *53*(1), 352–364.

Liu, Y., Zhang, W., Dong, B., Fan, Y., Wang, H., Feng, F., Chen, Y., Zhuang, Z., Cui, H., Li, Y., & Che, W. (2023). U-NEED: A fine-grained dataset for user needs-centric E-commerce conversational recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval* (pp. 2723–2732). Taipei Taiwan: ACM, http://dx.doi.org/10.1145/3539618.3591878.

Liu, Y., Zhao, L., Liu, G., Lu, X., Gao, P., Li, X., & Jin, Z. (2018). Dynamic Bayesian logistic matrix factorization for recommendation with implicit feedback. In J. Lang (Ed.), *Proceedings of the twenty-seventh international joint conference on artificial intelligence* (pp. 3463–3469). ijcai.org, http://dx.doi.org/10.24963/IJCAI.2018/481.

Nguyen Thanh, T., Quach, N. D. K., Nguyen, T. T., Huynh, T. T., Vu, V. H., Nguyen, P. L., Jo, J., & Nguyen, Q. V. H. (2023). Poisoning GNN-based recommender systems with generative surrogate-based attacks. *ACM Transactions on Information Systems*, *41*(3), 1–24. http://dx.doi.org/10.1145/3567420.

Rani, S., Kaur, M., Kumar, M., Ravi, V., Ghosh, U., & Mohanty, J. R. (2023). Detection of shilling attack in recommender system for YouTube video statistics using machine learning techniques. *Soft Computing*, *27*(1), 377–389. http://dx.doi.org/10.1007/s00500-021-05586-8.

Rong, D., He, Q., & Chen, J. (2022). Poisoning deep learning based recommender model in federated learning scenarios. In L. D. Raedt (Ed.), *Proceedings of the thirty-first international joint conference on artificial intelligence* (pp. 2204–2210). ijcai.org, http://dx.doi.org/10.24963/IJCAI.2022/306.

Rong, D., Ye, S., Zhao, R., Yuen, H. N., Chen, J., & He, Q. (2022). FedRecAttack: Model poisoning attack to federated recommendation. In *2022 IEEE 38th international conference on data engineering* (pp. 2643–2655). IEEE.

Si, M., & Li, Q. (2020). Shilling attacks against collaborative recommender systems: A review. *Artificial Intelligence Review*, *53*(1), 291–319. http://dx.doi.org/10.1007/s10462-018-9655-x.

Song, J., Li, Z., Hu, Z., Wu, Y., Li, Z., Li, J., & Gao, J. (2020). Poisonrec: An adaptive data poisoning framework for attacking black-box recommender systems. In *36th IEEE international conference on data engineering* (pp. 157–168). IEEE, http://dx.doi.org/10.1109/ICDE48307.2020.00021.

Song, Y., Zhou, C., Wang, X., & Lin, Z. (2023). Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing. arXiv:2302.01524.

Turk, A. M., & Bilge, A. (2019). Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks. *Expert Systems with Applications*, *115*, 386–402. http://dx.doi.org/10.1016/J.ESWA.2018.08.001.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T. (2019). Neural graph collaborative filtering. In B. Piwowarski, M. Chevalier, E. Gaussier, Y. Maarek, J. Nie, & F. Scholer (Eds.), *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174). ACM, http://dx.doi.org/10.1145/3331184.3331267.

Wang, Z., Liu, B., Lin, C., Zhang, X., Hu, C., Qin, J., & Luo, L. (2023). Revisiting data poisoning attacks on deep learning based recommender systems. In *IEEE symposium on computers and communications* (pp. 1261–1267). IEEE, http://dx.doi.org/10.1109/ISCC58397.2023.10218302.

Wang, Y., Liu, Y., & Shen, Z. (2023). Revisiting item promotion in GNN-based collaborative filtering: A masked targeted topological attack perspective. In *Proceedings of the AAAI conference on artificial intelligence: vol. 37*, (pp. 15206–15214).

Wang, Y., Liu, Y., Wang, Q., Wang, C., & Li, C. (2023). Poisoning self-supervised learning based sequential recommendations. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval* (pp. 300–310). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3539618.3591751.

Wu, C., Lian, D., Ge, Y., Zhu, Z., & Chen, E. (2021). Triple adversarial learning for influence based poisoning attack in recommender systems. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1830–1840). Virtual Event Singapore: ACM, http://dx.doi.org/10.1145/3447548.3467335.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021). Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* (pp. 726–735). Virtual Event Canada: ACM, http://dx.doi.org/10.1145/3404835.3462862.

Wu, C., Wu, F., Qi, T., Huang, Y., & Xie, X. (2022). FedAttack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 4164–4172). Washington DC USA: ACM, http://dx.doi.org/10.1145/3534678.3539119.

Xia, Y., Wu, J., Yu, T., Kim, S., Rossi, R. A., & Li, S. (2023). User-regulation deconfounded conversational recommender system with bandit feedback. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2694–2704). Long Beach CA USA: ACM, http://dx.doi.org/10.1145/3580305.3599539.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., & Lin, X. (2019). Topology attack and defense for graph neural networks: an optimization perspective. arXiv:1906.04214.

Yeh, C.-Y., Chen, H.-W., Yang, D.-N., Lee, W.-C., Philip, S. Y., & Chen, M.-S. (2023). Planning data poisoning attacks on heterogeneous recommender systems in a multiplayer setting. In *2023 IEEE 39th international conference on data engineering* (pp. 2510–2523). IEEE.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems: vol. 31*.

You, X., Li, C., Ding, D., Zhang, M., Feng, F., Pan, X., & Yang, M. (2023). Anti-fakeu: Defending shilling attacks on graph neural network based recommender model. In *Proceedings of the ACM web conference 2023* (pp. 938–948). Austin TX USA: ACM, http://dx.doi.org/10.1145/3543507.3583289.

Yu, Y., Liu, Q., Wu, L., Yu, R., Yu, S. L., & Zhang, Z. (2023). Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI conference on artificial intelligence: vol. 37*, (pp. 4854–4863).

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022). Are graph augmentations necessary?: Simple graph contrastive learning for recommendation. In E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, & G. Kazai (Eds.), *SIGIR '22: the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1294–1303). ACM, http://dx.doi.org/10.1145/3477495.3531937.

Zeng, L., Li, L., Gao, Z., Zhao, P., & Li, J. (2023). Imgcl: Revisiting graph contrastive learning on imbalanced node classification. In *Proceedings of the AAAI conference on artificial intelligence: vol. 37*, (pp. 11138–11146).

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. In *Advances in neural information processing systems: vol. 31*.

Zhang, H., Li, Y., Ding, B., & Gao, J. (2020). Practical data poisoning attack against next-item recommendation. In *Proceedings of the web conference 2020* (pp. 2458–2464). Taipei Taiwan: ACM, http://dx.doi.org/10.1145/3366423.3379992.

Zhang, Y., Liu, Y., Xiong, H., Liu, Y., Yu, F., He, W., Xu, Y., Cui, L., & Miao, C. (2023). Cross-domain disentangled learning for E-commerce live streaming recommendation. In *2023 IEEE 39th international conference on data engineering* (pp. 2955–2968). Anaheim, CA, USA: IEEE, http://dx.doi.org/10.1109/ICDE55515.2023.00226.

Zhang, F., Lu, Y., Chen, J., Liu, S., & Ling, Z. (2017). Robust collaborative filtering based on non-negative matrix factorization and r_1-norm. *Knowledge-Based Systems*, *118*, 177–190. http://dx.doi.org/10.1016/J.KNOSYS.2016.11.021.

Zhang, Y., Tan, Y., Zhang, M., Liu, Y., Chua, T., & Ma, S. (2015). *Catch the black sheep: Unified framework for shilling attack detection based on fraudulent action propagation* (pp. 2408–2414). AAAI Press, URL http://ijcai.org/Abstract/15/341.

Zhang, H., Tian, C., Li, Y., Su, L., Yang, N., Zhao, W. X., & Gao, J. (2021). Data poisoning attack against recommender system using incomplete and perturbed data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 2154–2164). Virtual Event Singapore: ACM, http://dx.doi.org/10.1145/3447548.3467233.

Zhang, S., Yin, H., Chen, T., Huang, Z., Nguyen, Q. V. H., & Cui, L. (2022). PipAttack: Poisoning federated recommender systems for manipulating item promotion. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 1415–1423). Virtual Event AZ USA: ACM, http://dx.doi.org/10.1145/3488560.3498386.

Zhao, G., Qian, X., & Xie, X. (2016). User-service rating prediction by exploring social users' rating behaviors. *IEEE Transactions on Multimedia*, *18*(3), 496–506.

Zhu, H., Ge, H., Gu, X., Zhao, P., & Lee, D. L. (2023). Influential recommender system. In *2023 IEEE 39th international conference on data engineering* (pp. 1406–1419). IEEE.

Zügner, D., Akbarnejad, A., & Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2847–2856). London United Kingdom: ACM, http://dx.doi.org/10.1145/3219819.3220078.