Poisoning GNN-based Recommender Systems with Generative Surrogate-based Attacks

THANH TOAN NGUYEN[*], Griffith University, Australia
KHANG NGUYEN DUC QUACH[*], Griffith University, Australia
THANH TAM NGUYEN, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam
THANH TRUNG HUYNH, Griffith University, Australia
VIET HUNG VU, Hanoi University of Science and Technology, Vietnam
PHI LE NGUYEN, Hanoi University of Science and Technology, Vietnam
JUN JO, Griffith University, Australia
QUOC VIET HUNG NGUYEN, Griffith University, Australia

With recent advancements in graph neural networks (GNN), GNN-based recommender systems (gRS) have achieved remarkable success in the past few years. Despite this success, existing research reveals that gRSs are still vulnerable to *poison attacks*, in which the attackers inject fake data to manipulate recommendation results as they desire. This might be due to the fact that existing poison attacks (and countermeasures) are either model-agnostic or specifically designed for traditional recommender algorithms (e.g., neighbourhood-based, matrix-factorisation-based, or deep-learning-based RSs) that are not gRS. As gRSs are widely adopted in the industry, the problem of how to design poison attacks for gRSs has become a need for robust user experience. Herein, we focus on the use of poison attacks to manipulate item promotion in gRSs. Compared to standard GNNs, attacking gRSs is more challenging due to the heterogeneity of network structure and the entanglement between users and items. To overcome such challenges, we propose `GSPAttack`– a generative surrogate-based poison attack framework for gRSs. `GSPAttack` tailors a learning process to surrogate a recommendation model as well as generate fake users and user-item interactions while preserving the data correlation between users and items for recommendation accuracy. Although maintaining high accuracy for other items rather than the target item seems counterintuitive, it is equally crucial to the success of a poison attack. Extensive evaluations on four real-world datasets revealed that `GSPAttack` outperforms all baselines with competent recommendation performance and is resistant to various countermeasures.

CCS Concepts: • **Information systems** → *Information systems applications*; *Information retrieval*; **Social networks**; **Recommender systems**; • **Security and privacy** → *Social engineering attacks*; • **Computing methodologies** → *Neural networks.*

[*]Both authors contributed equally to this research.

Authors' addresses: Thanh Toan Nguyen, Griffith University, Australia, thanhtoan.nguyen@griffithuni.edu.au; Khang Nguyen Duc Quach, Griffith University, Australia, k.quach@griffith.edu.au; Thanh Tam Nguyen, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam, nt.tam88@hutech.edu.vn; Thanh Trung Huynh, Griffith University, Australia, thanhtrunghuynh93@gmail.com; Viet Hung Vu, Hanoi University of Science and Technology, Vietnam, hung.vv162050@sis.hust.edu.vn; Phi Le Nguyen, Hanoi University of Science and Technology, Vietnam, lenp@soict.hust.edu.vn; Jun Jo, Griffith University, Australia, j.jo@griffith.edu.au; Quoc Viet Hung Nguyen, Griffith University, Australia, quocviethung1@gmail.com.

1046-8188/2022/10-ART $15.00
https://doi.org/10.1145/3567420

Additional Key Words and Phrases: manipulative item promotion, graph neural networks, poison attacks, generative models, surrogate modeling

1 INTRODUCTION

In the age of information explosion, recommender systems (RSs) have become indispensable tools for social platforms and online businesses [13, 17, 37, 50, 56]. In the past decades, RSs have evolved from *matrix-factorisation-based* techniques [11, 15], to *association-rule-based* methods [61], to *deep-learning-based* algorithms [64]. Among contemporary deep learning algorithms, graph neural network (GNN) based RSs are undoubtedly the most attractive techniques due to their superior abilities in learning graph-structured data [41]—the fundamental structure for RSs in capturing collaborative signals between users and items [65]. With remarkable success in the past few years, GNN-based RSs have become not only new state-of-the-art techniques in the RSs research community. However, they have also been deployed to produce reliable recommendations in industries [22]. For example, Pinterest developed and deployed a GNN-based model named PinSage to help its users precisely locate their information in a graph of billions of nodes [60].

In a typical scenario, the suggestions from RSs are expected to be reliable and unbiased. Nonetheless, due to their open nature, they pose severe information-level security problems. More precisely, there is evidence that RSs are vulnerable to data poisoning attacks [42]—attacks that aim to inject carefully-designed fake users in order to manipulate or deceive the victim models in a particular way. While such vulnerabilities are necessary for researchers to study and assess the robustness of a recommender model, they might be harmful to the fairness of digital economic systems. In particular, many online service providers are excessively motivated to exploit poison attacks to promote their services and demote their competitors. Take Sony Pictures as an example; they used to manipulate the movie RS to recommend some of their movies by exploiting false users and movie reviews [1]. Therefore, from both research and industry points of view, there is an urgent call to thoroughly study poison attack problems in RSs in the fight against unethical attacks and facilitate the development of effective defense strategies.

Conventional poison attacks [42] in RSs are *agnostic* to the recommendation models. These attacks focus on generalization capabilities rather than being specially designed for a specific recommendation algorithm. Although they can be applied to a wide range of recommendation algorithms, they oftentimes achieve suboptimal performance [48]. To tackle this limitation, recent proposed attacks are optimised for a particular class of RSs, including matrix-factorisation-based [32], association-rule-based [42], neighbourhood-based [12], graph-based [18], deep-learning-based RSs [27]. However, the problem of designing an optimised poison attack for GNN-based RSs is still an open problem.

Because of some unique challenges, pioneer poison attacks in standard GNN domains [77] cannot be directly applied in the RSs domain. Firstly, some methods [76, 77] generally modify the attributes and structure of existing users in the system, which is impractical in real-world RSs because of *privacy concerns (C1)*. Secondly, although other attacks [53] against GNN models do not require modifying existing nodes, they are performed in a homogenous graph under the general setting. Therefore, implementing such poison attacks in RSs dealing with *heterogenity (C2)* of nodes is more challenging. Finally, the critical factor that guarantees the power of a RS is the *data correlation (C3)* between users and items. Such correlation drives nodes in the user-item graph entangled, and thereby, injected users can propagate the effects to many other items rather than only the targeted item. Preventing such negative propagation from being identified by the detection methods introduces another critical challenge.

In this paper, we propose a sequential poison attack for GNN-based RSs to overcome these challenges. To ensure the data correlation (C3), we first leverage the representations of both users and items to learn an optimal

[1]http://news.bbc.co.uk/1/hi/entertainment/1368666.stm

attack on a *surrogate model*. The framework shall automatically synthesize new users with the target item as prior knowledge rather than modify existing users because the production environment restricts such modification due to privacy concerns (C1). Then the discrete edges will be generated to associate fake users into a heterogeneous graph (C2) between users and items before feeding the poisoned graph as an input for the optimisation.

We summarise our contributions, as well as the structure of the paper, following some background (Section 2.1), as follows:

- `GSPAttack` *Framework:* Section 3.3 presents our proposed data poisoning attack framework, a sequential attack that is optimised for GNN-based RSs. It includes three components: a *GAN-like* user generation, an edge generation, and a surrogate model component.
- *Unnoticeable User Generation:* Section 4 presents a *GAN-like* algorithm that generate fake users that remain *unnoticeable* by preserving important data characteristics.
- *Discrete Edge Generation:* Section 5 applies the Gumbel-Top-k technique to jointly capture the correlation effect between structure and users' features to generate malicious edges.
- *Surrogate Model Training:* Section 6 introduces a training algorithm on a surrogate model to maximise the attackers' goals using the generated edges and users.

In Section 7, we evaluate our approach using public datasets in the RSs domain. The results indicate that our proposed framework successfully promotes the target item without degrading the performance of the victim model while being unnoticeable to detection methods. Finally, we discuss related works in Section 8 before concluding in Section 9.

2 PRELIMINARIES

In this section, we first present the fundamental settings of GNN-based RSs and general RSs (Section 2.1). We then demonstrate the poison attack process using a running example (Section 2.2) before formally outlining the research problem of poison attack against GNN-based RSs (Section 2.3).

2.1 GNN-based Recommender Systems

General Recommender Systems. General RSs assume that users hold static preferences for an item or a set of items, and the recommendation algorithms try to model such preferences using explicit (i.e., ratings) or implicit feedback (i.e., views or purchases). A common approach is to learn hidden representations of both users and items and reconstruct the users' historical preferences using these hidden representations. In particular, the general recommendation model is formulated as follows: Let $\mathcal{U}$, $\mathcal{I}$ denote the sets of users and items, respectively. The historical preferences of users on items are recorded in the preference matrix $\mathcal{R} : \mathcal{U} \times \mathcal{I}$, where any interaction $r_{ui} \in \mathcal{R}$ is **1** if user u has interacted with item i (i.e., a positive instance), and is **0** if there are no historical interactions between them (i.e., negative instance). The general RSs try to learn h_u, h_i which are hidden representations of user u and item i, respectively. Then, the recommender model shall estimate a user's preference using the following formulation:

$$\hat{r}_{ui} = f(h_u, h_i) \tag{1}$$

where the estimation function $f(.)$ could be dot product, cosine similarity, or multi-layer perceptions, and $\hat{r}_{ui}$ is the estimated preference score of user u on item i, which is presented in the form of probability.

GNN-based Recommender Systems. The basic idea of GNN RSs is to map users' historical preferences to a bipartite user-item interaction graph and to learn the hidden representation of nodes (i.e., users and items) by feature smoothing over the graph [8]. To this end, the recommender models iteratively perform graph convolution—aggregating and transforming features from the neighbours to construct the new representation of a target node—over the user-item graph. Such neighbourhood aggregating and transforming operations could be formulated as follows:

$$\mathbf{h}_u^{(k+1)} = \mathbf{f}\left(\mathbf{agg}(\mathbf{h}_u^{(k)}, \{\mathbf{h}_i^{(k)} : i \in \mathcal{N}_u\})\right), \\ \mathbf{h}_i^{(k+1)} = \mathbf{f}\left(\mathbf{agg}(\mathbf{h}_i^{(k)}, \{\mathbf{h}_u^{(k)} : u \in \mathcal{N}_i\})\right). \tag{2}$$

where $\mathbf{h}_u^{(k)}$, $\mathbf{h}_i^{(k)}$ are respectively the hidden representation of user u and item i after k iterations, $\mathcal{N}_u$ is the set of items interacted with by user u, while $\mathcal{N}_i$ is the set of users who interact with item i. The aggregation function $\mathbf{agg}(.)$ is the heart of a graph convolutional network [36, 38, 49, 55], and $\mathbf{f}(.)$ is a nonlinear activation function. The preference estimation between the user u and the item i is defined based on their hidden representations as follows:

$$\hat{r}_{ui} = \mathbf{f}(\mathbf{h}_u^{(k+1)}, \mathbf{h}_i^{(k+1)}) \tag{3}$$

where $\mathbf{f}(.)$ is a scoring function that is similar to general recommendation models, and $\hat{r}_{ui}$ is adopted as the ranking score of items to users in the recommendation list.

2.2 Running example

To provide a better intuition of how a poison attack is performed against an RS, we demonstrate the most common attack approach as a running example in Fig. 1. Similar to existing works [18, 32, 45], we assume that the attacks follow a simple two-step process:

(1) **Fake users creation**: Based on careful analysis, fake user profiles are manually designed or automatically generated in such a way that they are similar to genuine ones.
(2) **Behaviours injection**: The attackers inject behaviours of fake users into the targeted system in order to manipulate it towards the attack goal.

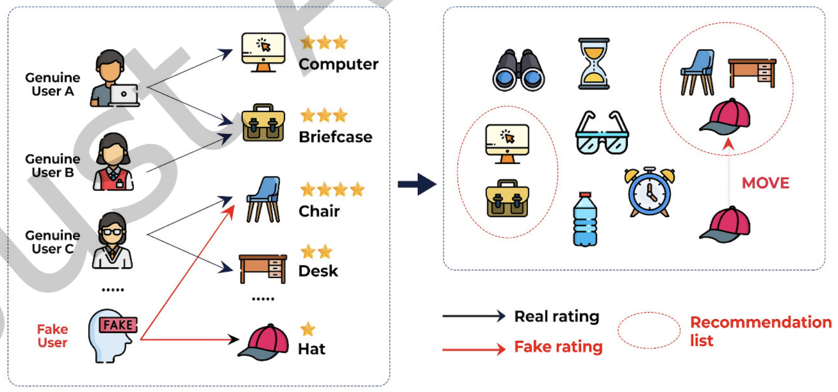


Fig. 1. A running example on poison attack.

In the example, interactions between users and items are represented using a bipartite graph, in which a node denotes a user or an item, and an edge denotes an interaction between them. Based on that graph-based representation, a GNN-based RS will learn the hidden representation of nodes (i.e., users and items) by feature transforming over that graph. These hidden representations shall be later used to compute the recommendation list. As we can see in the example, the behaviours in this scenario are fake interactions. By implanting these fake interactions to a set of specific items (i.e., the hat and the chair), the attacker could promote the hat (with a low rating) to appear in the recommendation list of the RS. Although behaviours of injection may differ between diverse attack approaches, most of the attacks follow the same two-step process. Thereby, this running example provides exciting readers with a quick intuition of how most poison attacks in RSs work.

2.3 Poison Attacks

Following the same process as presented in the running example, we assume that the attacker can register a set of new users (which we denote as *malicious users*) and manipulate their behaviors. As we aim to generate these malicious users automatically and keep them unnoticeable to the filter system, a number of first epochs are reserved for training the users' generator only. After a pre-defined epoch, the attack is launched, and all malicious users are trained to identify the best interactions to achieve the attack goal. The problem is formulated as follows:

Problem 1 (**Poison Attack against GNN-based RSs**). *Given a GNN-based RS (gRS) parameterised by Θ, our poison attack aims to promote a target item $i^* \in \mathcal{I}$ by injecting a set of malicious users $\mathcal{U}^*$. Note that each injected user will interact with a number of items that form an injected interaction matrix $\mathcal{R}^*$. Our problem is to generate $\mathcal{U}^*$ and $\mathcal{R}^*$, such that they satisfy the following optimisation:*

$$\max_{u \in \mathcal{U} \backslash \mathcal{U}^*} gRec(u_{i^*}|\Theta) \tag{4}$$

where $gRec(u_{i^*}|\Theta)$ is the probability that i^* appears in the recommendation list to user u.

3 MODELS AND APPROACH

This paper aims to develop an effective strategy to perform poison attacks against GNN-based RSs. In this section, we first present the base victim model specification (in Section 3.1), then describe the details of the threat model (in Section 3.2) that the attackers can obtain. Finally, we reveal the approach overview of our proposed framework (in Section 3.3).

3.1 The Base Victim Model

We aim to design a poison attack framework that is able to apply to the majority of GNN-based RSs. Without loss of generality, we adopt the graph neural collaborative filtering (GNCF) [58]—a representative state-of-the-art GNN-based approach for recommendation—as the base recommender in this study. GNCF largely tailors the idea of the standard version of the graph convolution network (GCN) [29] to a collaborative filtering system. More precisely, the convolution operations in Eq. 2 are re-defined as:

$$\begin{aligned} \mathbf{h}_u^{(k+1)} &= \sigma\left(\mathbf{W}_1\mathbf{h}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{|\mathcal{N}|}\left[\mathbf{W}_1\mathbf{h}_i^{(k)} + \mathbf{W}_2(\mathbf{h}_i^{(k)} \odot \mathbf{h}_u^{(k)})\right]\right), \\ \mathbf{h}_i^{(k+1)} &= \sigma\left(\mathbf{W}_1\mathbf{h}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{|\mathcal{N}|}\left[\mathbf{W}_1\mathbf{h}_i^{(k)} + \mathbf{W}_2(\mathbf{h}_i^{(k)} \odot \mathbf{h}_u^{(k)})\right]\right). \end{aligned} \tag{5}$$

where $|\mathcal{N}| = \sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ is the symmetric normalisation of the cardinality of users set $\mathcal{N}_u$ and items set $\mathcal{N}_i$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable matrix for aggregation functions at each layer, $\odot$ is an inner product, and $\sigma(.)$ is a nonlinear function. It is worth noting that our approach is generic, and therefore compatible with a wide range of other GNN-based RSs including HashGNN [51], AGCN [59], and LightGCN [25].

3.2 Threat Model

With the base recommender defined above, in this section, we present the threat model of the proposed attacking framework. Our threat model consists of three dimensions, as outlined below.

Attack Goal. According to the attackers' intents, the goals of poison attacks can be categorised into: *(i) promotion attacks*, *(ii) demotion attacks*, and *(iii) availability attacks* [44]. The promotion (demotion) attacks aim to maximise (minimise) the availability of the target items in the recommendation list, and availability attacks typically aim to maximise the recommendation errors of the victim model. The proposed `GSPAttack` can be efficiently performed to achieve these goals by simply specifying different attack objectives for the surrogate model. For simplicity, we focus on promotion attacks in the current study.

Attack Knowledge. In this paper, the attackers are assumed to have one of the following levels of knowledge:

(1) *Full-knowledge:* In a full-knowledge setting, the attacker masters everything about the victim model, including the model's architecture, the parameter settings, and the users' historical interactions. We include the full-knowledge setting because it is still the setting of most existing works [42]. The reason for this is that many giant online providers, such as Amazon, Pinterest, and Netflix, continue to depict their recommendation algorithms [2]. In addition, a full-knowledge setting provides upper bound damage in the worst case, revealing the critical insights and implications for developing defense methods. Therefore, understanding the full-knowledge setting is a crucial and practical requirement.
(2) *Partial-knowledge:* In a partial-knowledge setting, the attacker knows only a portion of historical behaviours and the RS type. The attacker uses the partially available data to train a local surrogate model with the same algorithm as the victim model. The attacker leverages the full knowledge of the surrogate model to find the well-designed fake users and their interactions with existing items. This faked data will then be uploaded to the victim model to perform the attack during the model's update.

Attack Capability & Approach. In theory, any RS can be fragile if the attackers inject an adequate number of fake users. However, there is a trade-off between the attack capability and the system's protection ability in practical applications. In particular, injecting excessive fake users is hard to conduct and will inevitably create distancing markers from ordinary users, leading to being detected by system prevention. Given these considerations, the upper bound factors of the attacker's capability should be considered, as presented below.

- *Attack size (μ):* The number of *fake users* allowed to be implanted during the attacking process, denoted as $\mathcal{U}'$. This amount is much less than that of normal users (referred to as $\mathcal{U}$) in the system.
- *Profile size (σ):* The number of non-zero *fake ratings* permitted for each controlled user. The attacker can add at most σ ratings for each fake user because a regular user often interacts with a few items in a real-life system [2, 42].

Given the attack knowledge and capability, we train a local surrogate model towards the attack goal before transferring the attack to the victim system. Details of the approach are revealed in the next section.

3.3 Approach Overview

To address the problem of poison attacks in GNN-based RSs, we propose `GSPAttack`. Before diving into the details of the proposed framework, we summarize its underlying design principles.

Design principles. Reflecting on emerging applications of RSs [2], the designed attack model, and on the empirical results reported for state-of-the-art approaches [42], we derive the following requirements for a practical solution for RS poison attack:

- (**R1**) *Unnoticeblility.* The fake user shall be indistinguishable from the genuine users based on reasonable metrics.

- (**R2**) *Proper propogation.* Due to the entanglement among users and items, only proper prorogations between fake users and items should be performed to avoid any unexpectedly adverse effects on the overall performance of the targeted system.
- (**R3**) *Transferability.* The solution shall be re-used and transferred to multiple GNN-based RS approaches.

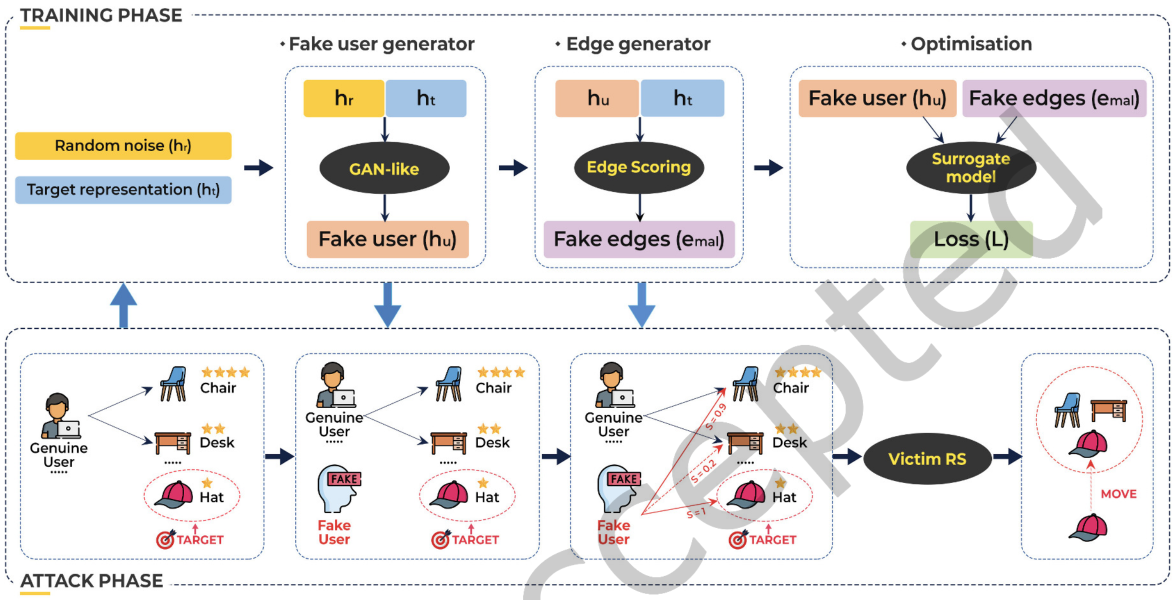


Fig. 2. The proposed framework.

Framework overview. Fig. 2 reveals the overall architecture of our proposed framework, `GSPAttack`. The upper part illustrates the main workflow of our attack. Although our framework allows generating μ fake users in one attack process, we demonstrate the main workflow for generating one fake user for simplicity. However, the full architecture of the framework can be easily recovered by repeating the generators μ times. More precisely, the attack workflow consists of three steps, as outlined below.

Step 1: Fake user generator. In the first component, we design a GAN-like algorithm to craft the fake user while making it indistinguishable from the genuine ones. Data filtering algorithms will easily recognize the injected user if it is significantly different from the current users [3]. Therefore, ensuring unnoticeability (**R1**) during the user generation becomes a must-have constraint to make the solution practical for real-world RSs. Details of this component are presented in Section 4.

Step 2: Edge generator. The malicious edge helps fake users spread their attack intent to proper items. Due to the limited attack budget and the requirement for proper propagation (**R2**), only a tiny portion of possible edges should be selected as the malicious edges. Such an edge selection process can be formulated as an optimisation problem. However, the discrete nature of edges results in a non-differentiable domain, which neglects a back-propagation gradient to build the optimisation of implanted edges. To overcome this challenge, we adopted the *Gumbel-Top-k* technique [30] to formulate the edge optimisation problem as a scoring-based model. We describe the component in detail in Section 5.

Step 3: Joint optimisation. The generated user and edges, together with the original graph, are fed into a surrogate model (**R2**) to optimise toward the attacking intent. The joint optimisation aids in depicting the coupling effect between the injected user and the edges of the previous generators. In addition, to ensure transferability (**R3**), the chosen architecture of the surrogate model contains the two most common designs—feature aggregation and nonlinear activation, which most GNN models adopt. Details of the optimisation are provided in Section 6.

The lower part sketches a workflow of using the attacked graph to the victim model during the model update. It is worth noting that our approach can be applied to any GNN-based victim model under a block-box attack setting.

4 UNNOTICEABLE USER GENERATION

Typically, in a poison attack scenario, the attackers try to modify the data input in such a way that the modifications are *unnoticeable* (**R1**). To this end, in this section, we present the fake user generation process based on the idea of a Generative Adversarial Network (GAN) [19] in Section 4.1 and discuss its complexity in Section 4.2.

4.1 Fake User Generation

In computer vision, the crafting of new images is easily verified by visually assessing and using some simple constraints (e.g., distant constraints) [19]. However, this is much harder to obtain in the graph setting due to the unsuitability of visually inspecting a sufficiently large graph. Facing such a different constraint, we approach the unnoticeability of fake users from the *feature statistics* perspective. More precisely, we learn the underlying distribution of existing users in the RS and use this exact distribution to sample fake users similar to benign ones. Next, we will present the process of fake user generation based on the idea of Generative Adversarial Network (GAN) [19].

Given an RS with n existing users ($\mathcal{U}$), the feature matrix of these users is denoted as $X \in R^{n \times d}$, where d is the dimension of the user features (each row of X corresponds to the features of a user). The idea is to generate a number of m fake users ($\mathcal{U}^*$) so that after the generation, the feature matrix becomes X' and its corresponding representation becomes: $X' = \begin{bmatrix} X \\ X_{fake} \end{bmatrix}$. The fundamental idea is to employ a discriminator to create fake users that seem like real users. To do this, we create a neural network with two fully connected layers and a softmax layer as the discriminator, which is written as:

$$D(X') = \textbf{softmax}(\sigma(X'W^{(0)})W^{(1)}) \tag{6}$$

where the **softmax** function is applied to every row of the output matrix. The discriminator determines a user is *real* or *fake* based on each element in $D(X')$.

To deceive the discriminator, we intend to create fake users with comparable characteristics to real users. Since the output of the discriminator is binary, we employ binary cross-entropy loss, which is defined as:

$$\mathcal{L}_{usr} = \sum_{u \in \mathcal{U}^* \cup \mathcal{U}} L(\hat{p}_u, y_u) \tag{7}$$

where $L(\hat{p}_u, y_u) = -(y_u \log(\hat{p}_u) + (1 - y_u)\log(1 - \hat{p}_u))$, $(y_u, \hat{p}_u)$ a binary indicator of ground truth (real or fake), and the predicted probability $\hat{p}_u$ of the user u, respectively.

After the discriminator is converged, we use the generator of GAN to generate a fake user from a random variable. The attack's goal is to promote the target item t, which is inherently related. Therefore, we concat the representation h_r of a random noise r, and the representation h_t of the target t as the generator's input. The generated fake user u is forwarded to the surrogate model to obtain its corresponding representation h_u.

4.2 Complexity

The complexity of GAN-like user generation per iteration of updating features of fake users costs $O(\mu d)$, where μ is the number of added users, and d is the dimension of users' features.

5 MALICIOUS EDGES INJECTION

Malicious edges help fake users to spread their attack intention to the required candidate items (**R2**). In this section, we first design a simple but effective optimisation-based approach to prove the attack feasibility and explore the upper bound performance (Section 5.1). Then, we present the popularity bias rooted in RSs (Section 5.2), and based thereon, we propose a practical generator of malicious edges in terms of both efficacy and generalizability (Section 5.3). Finally, we analyze the algorithm complexity (Section 5.4).

5.1 Optimisation-based edge generation

Malicious edge generation is the process of manipulating the adjacency matrix to associate the newly generated fake users into the original graph (G) in order to form the perturbed graph (G'). More precisely, the adjacency matrix is transformed from A to $A' = \begin{bmatrix} A & B^T \\ B & C \end{bmatrix}$. Initially, the B and C are set to 0 and I (identity matrix), respectively. Our goal is to find the B and C that help achieve the optimization goal—promoting the target item to as many users as possible.

The approach. Here, we design an optimized-based approach, denoted as `OPT`, to generate malicious edges to facilitate the item promoting attack problem. We specifically design the malicious edges to fake users as a free variable and inject these generated malicious edges into the original graph. Then, the attacked graph input is fed into a surrogate model and optimise towards the optimisation goal (Eq. 4) until convergence. It is worth noting that the free variables that represent the malicious edges are discrete variables. To overcome the challenge of stochastic gradient descent over a discrete variable, we adopt the *Gumbel-Top-k* technique [30], and its technical details shall be discussed in Section 5.3.

Highlight results. We execute `OPT` on two popular public datasets in RS, namely MovieLen [23] and Amazon [39]. The algorithm successfully promotes the target item to have appeared in the recommendation list of up to 100% of the users in the victim system. The results indicate that the performance of `OPT` significantly outperforms existing baselines over all datasets (details shall be provided in the evaluation section). The results also validate the feasibility and effectiveness of the optimisation-based malicious edge generation method.

Even though `OPT` achieves excellent performance, the edge selection from all possible edges still leaves a burden on the model training. In the next section, with the presence of popularity bias rooted in RS [1] we propose a strategy to achieve the attack goal more practically and effectively.

5.2 Popularity Bias

It is widely acknowledged that data-driven RSs are inherently biased toward popular items—items that are frequently visited [1], and GNN-based RSs are no exception. In this regard, we explore a cost-effective approach that takes advantage of only popular items to generate malicious interactions.

Popularity Information. To boost the target item's popularity, we need existing items' popularity information as prior knowledge. Fortunately, the most popular items can be directly extracted from the frequencies of historical interactions between users and items. Even when attackers cannot obtain the entire datasets due to access restrictions in some specific scenarios, modern online platforms provide many visible clues to retrieve the items' popularity information. For instance, e-commerce platforms such as Amazon and eBay present their highest popularity items in terms of hot sales and suggest them to every user on their home page under the showcase section. Another source for finding item information is fully public directories such as Yelp. Hence, the availability of popular items is ensured by fetching them from various data sources, making the capacity of GSPAttack remain unchanged from the initial budget and requirements.

Attack Intuition. The OPT approach requires fake users to manipulate a significant number of items in order to successfully degrade the target model. However, the attacker's capacity is limited to the *profile size (σ)*—the maximum number of items with which a fake user can interact with. With this regard and the existence of prior knowledge about the popular items, we designed a practical approach that takes advantage of popular items to craft the fake edges. In contrast with unpopular or long-tail items, interacting with popular items is more likely to receive a high ranking score as popular items are learned with much higher frequency by the surrogate model. The next step is to reveal our suggested approach to creating fake edges, which we call *popularity-based edge generation.*

5.3 Popularity-based edge generation

The malicious edges assist fake users in spreading their attack intent to the target items and the appropriate candidates. Herein, we limit the candidate items to a set of *popular items* due to the inherent popularity bias in RSs [1]. Due to the limited injected edge budget σ required by the *profile size*, for each fake user, we design a model to score on all candidate items and choose the best $k = \sigma$ items to construct the malicious edges. The impact of connecting a particular candidate item to the fake user on the attack performance is measured by the scores of selected edges.

Jointly modelling. In graph neural network approaches, both network structure and node attributes are equally crucial in recommendation tasks [8, 60]. In this regard, we jointly model the malicious edges in order to capture the coupling effect between the generated edges and the features of fake users. To this end, we incorporate the representation of the fake user and the target item into the edge generation process to guide the edge generation. Specifically, we forward the fake user u and the target item t through the surrogate model to obtain their representations h_u and h_t, respectively. By employing these representations as to the inputs, we utilise two-layer neural networks $\mathcal{F}$ to compute the probability of connecting across a set of candidate items, and to discretise the scores, we adopt the Gumbel-Top-k technique $\mathcal{G}$ as follows:

$$\begin{aligned} e_{mal} &= \mathcal{G}(\mathcal{F}(t, u, G; \theta^*)) \\ \mathcal{F}(t, u, G; \theta) &= \sigma([h_u || h_t] W_0 + b_0) W_1 + b_1 \end{aligned} \tag{8}$$

where $\theta = W_0, W_1, b_0, b_1$ are learning parameters.

Handling discreteness. To handle the non-differentiability of the discrete edges, we use the Gumbel-Top-k technique [30] to make the back-propagating gradients over discrete samples feasible. The Gumbel distribution G_i is widely used to differentially approximate sample discrete data[30]. Formally, the Gumbel-Softmax equation is formulated as follows:

$$\text{Gumbel-Softmax}(z)_i = \frac{\exp\left(\frac{z_i + G_i}{\tau}\right)}{\sum_{j=1}^{n} \exp\left(\frac{z_i + G_i}{\tau}\right)} \tag{9}$$

where τ denotes the temperature parameter to control the smoothness of the approximation, z denotes the output of the neural network $\mathcal{F}$, $G_i = -\log(-\log(U_i))$ and U_i = Uniform(0,1) is the uniform distribution. The Gumbel-Softmax technique explore the edge selection through the Gumbel distribution G_i. We further use the ϵ parameter to control the strength of exploration to encourage exploration intensity, as outlined below:

$$\text{Gumbel-Softmax}(z, \epsilon)_i = \frac{\exp\left(\frac{z_i + \epsilon G_i}{\tau}\right)}{\sum_{j=1}^{n} \exp\left(\frac{z_i + \epsilon G_i}{\tau}\right)} \tag{10}$$

The function $\mathcal{G}$ for Gumbel-Top-k is defined as:

$$\mathcal{G}(z) = \sum_{j=1}^{k} \text{Gumbel-Softmax}(z \times mask_j, \epsilon) \tag{11}$$

where k is to the profile size σ—the maximum edges that allow a fake user to connect to the set of candidate items, the $mask_j$ prevents the selected edges from being selected again. The obtained vector is sharp but not entirely discrete, which is advantageous for training. It is worth noting that, in this section, we present the joint modelling for edge generation over a specific fake user for simplicity. In reality, we optimize the process of edge generations for multiple fake users simultaneously by stacking multiple generation blocks into an end-to-end training pipeline.

5.4 Complexity Analysis

The complexity of each iteration of the edge generation costs $O(2d^2 + \sigma d)$, where μ is the number of added users, and d is the dimension of users' features.

6 SURROGATE MODEL TRAINING

We propose a sequential approach in this section, where we perform the poison attack a *surrogate model* first (Section 6.1), thus, leading to an attacked graph. During the process of attacking the surrogate model, both *explicit promotion* (Section 6.2) and *recommendation accuracy* (in Section 6.3) are optimised. Then, we jointly trained the learning process in an end-to-end manner (Section 6.4). Finally, the attacked graph is subsequently used to train the victim model to perform the attack.

6.1 Surrogate model

Most graph neural network (GNN) techniques follow the same propagation rules [8] to obtain the embeddings: (1) *feature transformation*, (2) *neighbourhood aggregation*, and (3) *nonlinear activation*. Following the *representative model [69]* strategy, we select a known representative model as the surrogate model, then transfer these attacks to other models that have a similar function to the target RS [25, 27, 40]. In the context of our work, we aim to design this attack approach as a check for transferability to assess the robustness of many GNN-based RSs. Based thereon, we chose a surrogate model with designs directly inherited from a standard GNN architecture. Recently, Neural Graph Collaborative Filtering (NGCF) [58] that covers all standard operations of a GNN was proposed and achieved the best performance for collaborative filtering. Therefore, we chose NGCF as the surrogate model for `GSPAttack`.

6.2 Explicit promotion

The ultimate goal of an item-promoting attack is to increase the visibility of the target item t as much as possible. Basically, in order to give the target item better exposure, the expected ranking score $\hat{r_{ut}}$ needs to be increased whenever a user u interacts with t. Within the attack budget in this study, the attack can control all malicious users. In this regard, we need to ensure the target items are visible to all malicious users as a minimum requirement. To achieve this goal, we explicitly boost the ranking score of t to all malicious users using the objective function as formulated below.

$$\mathcal{L}_{exPR} = - \sum_{u^* \in \mathcal{U}^*, t} \log \hat{r_{u^*t}} \tag{12}$$

Although the objective function is typical for RS, we define this objective function separately as we want to encourage extensive similarities between malicious users and the target item via a controllable coefficient (as further revealed in Section 6.4). Furthermore, for `GSPAttack` to gradually lead the victim model to anticipate positive feedback on the target item from other regular users similar to the controlled users.

6.3 Maintaining Plausible Performance

The main focus of our attack approach is to promote the target item gradually and silently without the intent of eradicating the target system. We maintain a realistic performance of the victim model by maximising the following objective since a significant degradation in performance is a highly abnormal phenomenon that is easily identified by both humans and systems.

$$\mathcal{L}_{per} = - \sum_{u \in \mathcal{U} \cup \mathcal{U}^*, i \in \mathcal{I}} r_{ui} \log \hat{r_{ui}} + (1 - r_{ui}) \log(1 - \hat{r_{ui}}) \tag{13}$$

where $\hat{r_{ui}}$ is the probability of observing an interaction between user u on item i. Thereby, the above cross-entropy loss function ensure high similarities between $\hat{r_{ui}}$ and the ground-truth r_{ui}.

6.4 Put-it-altogether

To capture the coupling effect between the attack goal and unnoticeable requirement, we define a loss function for `GSPAttack` to learn the model in an end-to-end manner. Rather than training each building block separately, the building blocks are jointly trained using a unified loss function formulated below.

$$\mathcal{L} = \mathcal{L}_{usr} + \alpha \mathcal{L}_{exPR} + \beta \mathcal{L}_{per} \tag{14}$$

where α and β are tunable and non-negative hyper-parameters to control the contribution of each block in the framework.

7 EXPERIMENT

In this section, we present the empirical evaluation of `GSPAttack`. More precisely, we aim to answer the following research questions:

- **RQ1** Does `GSPAttack` outperform diverse baseline techniques in an end-to-end comparison?
- **RQ2** What is the importance of each framework's component?
- **RQ3** How does `GSPAttack` perform transferring the attack to other GNN-based RSs?
- **RQ4** What is the effect of profile size on attack performance?
- **RQ5** What is the impact of each parameter of `GSPAttack`?

Before presenting our empirical results and findings, we first introduce the experimental setup.

7.1 Experimental Datasets

To evaluate the effectiveness of `GSPAttack`, we conducted experiments on four real-world datasets that are commonly used in security studies [18, 26, 67] for RSs. The statistics of four datasets are summarised as follows.

Frappe (FR). This is an app recommendation dataset that was conducted by [4]. The dataset contains 957 users and 4,082 items (i.e., apps) and their 96,203 interactions in different user contexts.

MovieLens (ML). This is a movie dataset [23] obtained from the public repository [2]. The dataset contains 6,040 users and 3,706 items (i.e., movies) and their interactions via 1,000,209 ratings.

Amazon (AM). Amazon-review is widely adopted as the evaluation dataset for RSs [39]. We selected Amazon-cellphone, which includes 13,174 users and 5,970 items (i.e., cellphone products) together with 103,593 ratings.

Last.fm (LF). This is a music dataset that implicitly quantifies the positive interactions between a user and a music artist via a tagging system. It contains 186,479 tag assignments given by 1,892 users to 17,632 music artists.

To avoid the "cold start" problem [28], we first employ a 10-core setting [24, 58] to ensure that each user and item has at least ten positive interactions. Then, we follow the settings of most previous attack methods [72, 77] to binarise their interactions as 1.0 for positive interactions and 0.0 for the remaining, and the ratio between negative and positive samples is kept at $q = \frac{1}{4}$. The age and occupation of users are used to construct the user features in the ML dataset. For other datasets, we use the learning-based feature initialisation method that has widely been used in the literature [16] to construct nodes' features. Finally, as `GSPAttack` requires popularity items during its edge generation process, we follow the same strategy as in recent works [72] to select the top 10% of items that receive the most number of interactions and label them as *popularity items*.

7.2 Baseline techniques

We compare `GSPAttack` to four different poison attack strategies in RSs. The first three are agnostic or independent of the recommendation model, and the last one is a data poisoning attack that is optimised for attacking graph neural network techniques. Notably, there has been no optimised attack method for GNN-based RSs.

Random Attack [21]. The attacker randomly selects a user in the target system and takes its attributes as the attributes of fake users with minor modifications. The attacker randomly selects μ filler items for edges and connects them to fake users.

Bandwagon Attack [21]. Bandwagon attack is similar to a random attack. The only difference is that it associates the filler items with popularity. The edges are crafted randomly between fake users and popular items.

Explicit Boosting [72]. The attacker directly optimises the explicit item promotion by minimising the $\mathcal{L}_{exPR}$ loss function.

Greedy-GAN [57]. This is a novel approach that applies "nodes attack" to GCN. We use the exact strategy of Greedy-GAN and modify the loss function to facilitate the recommendation task.

While GCMC Attack [75] and `GSPAttack` share some fleeting similarities, the prerequisites of each approach are substantially different. In particular, GCMC Attack requires a white-box setting, and the underlying recommender model has to apply a graph autoencoder. Therefore, the setting of GCMC Attack is orthogonal to the setting of the present study.

7.3 Evaluation Protocols

Our attack approach aims to promote a target item without sacrificing the performance of the targeted system. To evaluate the performance of `GSPAttack`, we use the two metrics which is widely employed in the literature [42, 72]:

Effectiveness on Promotion. To assess the effectiveness of the promotion attack, we employ the exposure rate at K (denoted as *ER@K*). More precisely, *ER@K* represents the fraction of users who receive the target item in their top-K recommendation list. Obviously, to maximise the promotion attack goal, the larger the *ER@K* achieved, the better the effectiveness of the approach is.

Effectiveness on Recommendation. It is crucial to validate that the attack approach does not destroy the underlying RS. To this end, we adopt the leave-one-out approach [70] and use the hit rate at K (denoted as *HR@K*) [42] to evaluate the accuracy of RSs with the existence of the poison attack. The higher this metric is the better performance the model can obtain.

[2]https://grouplens.org/datasets/movielens/

Effectiveness on Re-producing. We employ a precision metric (denoted as *Pres@K*) to evaluate how well our surrogate model can reproduce the results of the victim RS [74]. The metric is defined as the proportion of the top-K recommended items that match the victim RS's recommendations.

7.4 Experimental Settings

7.4.1 Hyperparameters. For GSPAttack, we set the embedding dimension, learning rate, and training epochs to 64, 0.01, and 100, respectively. We then examine the attack performance with different attack sizes by varying attack sizes $\mu \in \{0.5\%, 1\%, \text{and } 2\%\}$ depending on the sparsity of the datasets. The profile size σ is set to D_{avg}—the average degree of all users in the user-item graph.

7.4.2 Model Selection & Reproducibility. Neural Graph Collaborative Filtering (NGCF) [58]—a well-known GNN-based RS is chosen as the surrogate model for training the attack. The attack is then transferred to other GNN-based RSs, including HashGNN [51], AGCN [59], and LightGCN [25].

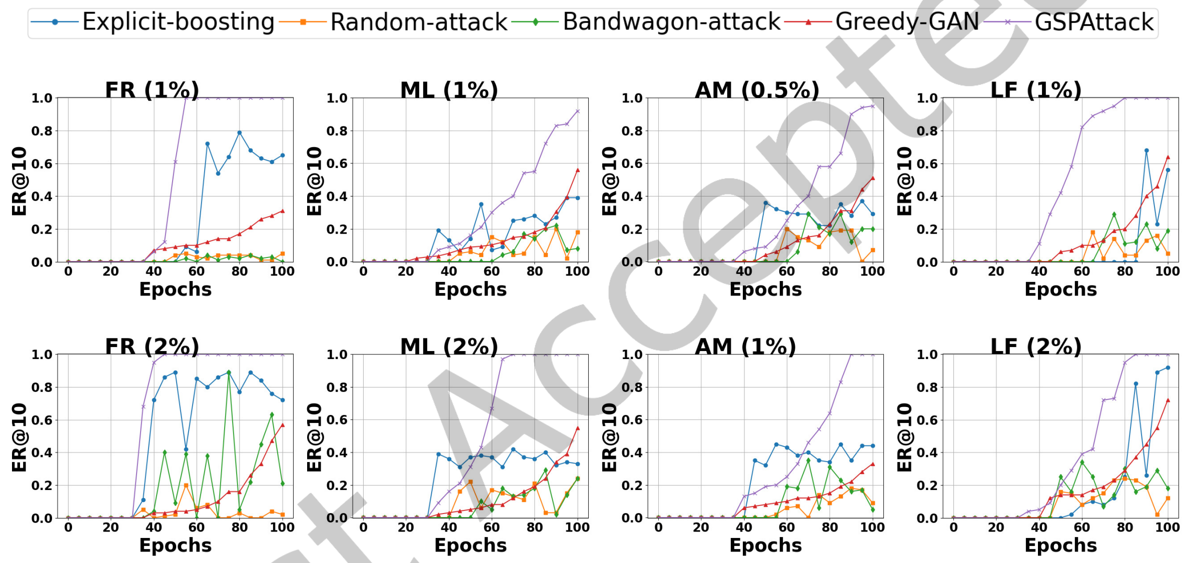


Fig. 3. End to end attack performance.

7.5 End to end comparison

In this section, we respond to **RQ1** by evaluating our framework in two key aspects of RSs: (i) attack performance, (ii) and recommendation performance.

General findings. It can be seen from Fig. 3 and Fig. 4 that GSPAttack outperformed other baselines in attack performance over all datasets while achieving comparable performance with others in recommendation task. More precisely, GSPAttack achieved an average improvement of 46% in attack performance over the best baseline, Greedy-GAN. This significant gain indicates that optimisation-based malicious edges facilitate attack performance.

Attack performance. To further assess the efficacy of GSPAttack in terms of promoting the target item, we present the performance *ER@10* over all datasets in Fig. 3. The number of epochs is represented on the X-axis, while the performance at each epoch is represented on the Y-axis. An amount of 0.5% to 2% of fake users is

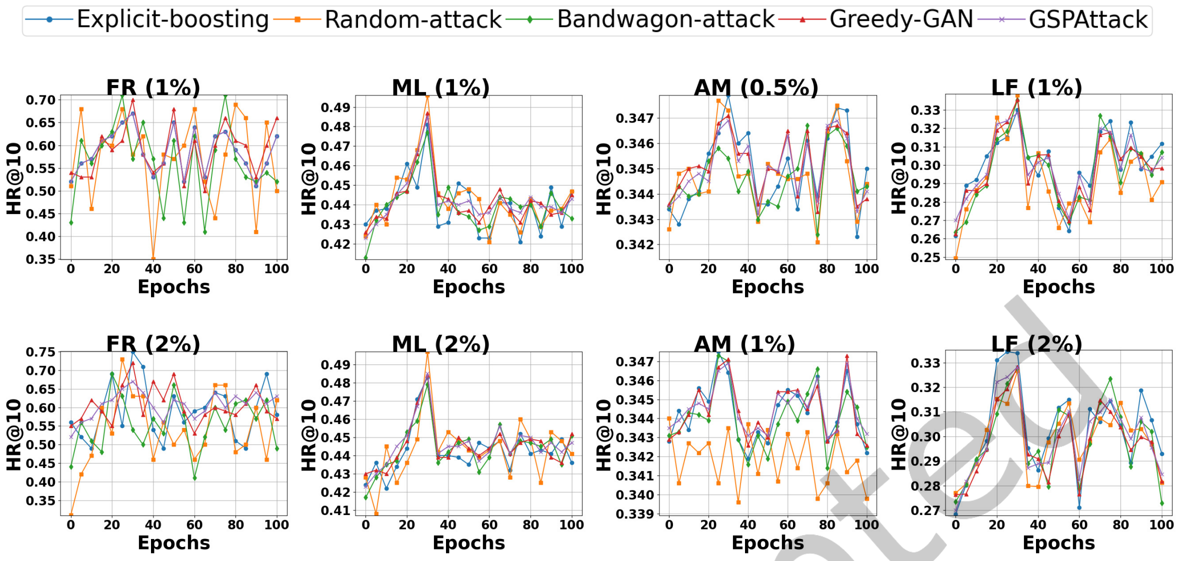


Fig. 4. End to end recommendation performance.

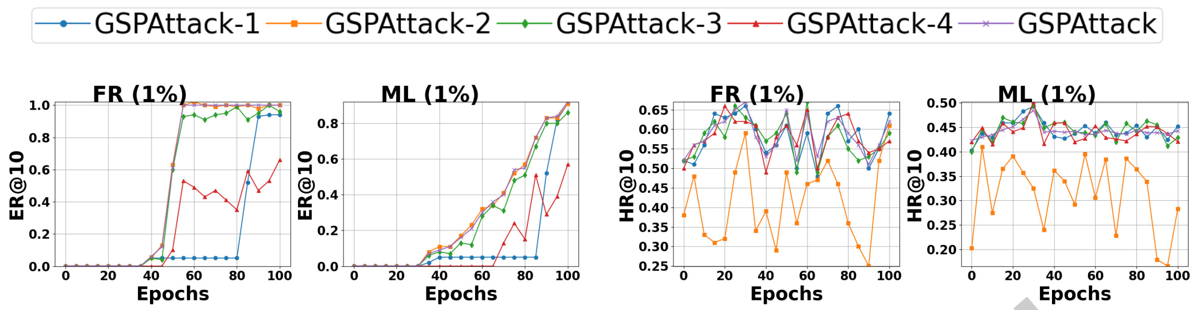
injected into each dataset for evaluation purposes. As the first 30 epochs are reserved for GSPAttack to train the unnoticeable user generation step, the attack for all baselines is scheduled to start after 30 epochs. The figure shows that GSPAttack consistently outperforms the baseline techniques, revealing a solid capability to successfully promote a low-popular target item to all users. Depending on the characteristics and sparsity of datasets, some datasets (i.e., FR and LF datasets) are easier to promote the target item to all users than others (i.e., ML and AM). However, with sufficient epochs, GSPAttack can achieve 100% in terms of *ER@K* in most cases, while the competing baselines suffer significant fluctuations. Another interesting finding is that model-optimised attacks (i.e., **Explicit Boosting** and **Greedy-GAN**) generally perform better than model-agnostic attack methods (i.e., **Random Attack** and **Bandwagon Attack**), revealing that the optimised paradigm outperforms simple agnostic attacks.

Recommendation performance. Maintaining a plausible performance, in addition to achieving the efficacy of promoting the target item, is critical to the attack's success. Although it may seem counterintuitive to promote other items rather than the target item in a promotional attack, we show that it is important for various reasons. First of all, significant decreases in the overall recommendation performance could lead to a failure of the attack as the filter system could detect such attacks as abnormal behaviours. In addition, only a recommender with high accuracy will have a large user base for promoting the target item and thus mutually enhance the attack goal.

As for the importance of recommendation performance, we present the performance curves of competing baselines on all datasets in Fig. 4. Overall, we notice that GSPAttack achieves competitive recommendation performance on all datasets. The result further confirms that GSPAttack is unharmful to the victim model, facilitating the attacks' success rate. While **random attack** causes more degradation and significant fluctuation in the recommendation performance than other baselines, other baselines achieve comparable results.

7.6 Ablation testing

We conduct an ablation study in this section to better understand the interplay of components in GSPAttack. We compare the performance of our proposed framework to several variants:



(a) Attack performance (b) Recommendation performance

Fig. 5. Ablation testing on attack performance.

- GSPAttack-1: We remove the constraint of explicit boosting $\mathcal{L}_{exPR}$ the target item.
- GSPAttack-2: The framework is optimised without the plausible performance preservation constraints ($\mathcal{L}_{per}$).
- GSPAttack-3: We deployed an anomaly detection method [3] to filter abnormal users based on users' features.
- GSPAttack-4: The filter system method is deployed similar to the GSPAttack-3 variant while we replace the unnoticeable user generator with a random user generator.

All experiments in this section are conducted using two datasets (FR and ML dataset) with the attack size of $\mu = 1\%$ (other datasets are omitted as they reveal the same trend). For each dataset, we present both attack and recommend performance of all variants, and the results are shown in Fig. 5.

Effect of explicit boosting. As shown in Fig. 5-a, the attack performance of GSPAttack-1 significantly drops when compared to the original version of GSPAttack. The reason is that GSPAttack relies on explicit boosting to help the target item gain visibility. The boosting effect is easily diluted by other popular items in the target system by removing this objective. An interesting finding is that GSPAttack-1 is more effective on FR than on ML datasets. A possible reason for that could be that FR is sparser and easier to manipulate without an explicit boost. In addition, the convergence speed of GSPAttack-1 to the attack goal is far behind the original GSPAttack. Therefore, it verifies the idea that employing the constraints of explicit boosting helps in achieving the attack goal with much less effort. The recommendation performance of GSPAttack-1 remains comparable to that of GSPAttack. The reason could be that the number of manipulated items is much smaller than regular items; therefore, the overall recommendation performance is not too much affected.

Effect of maintaining plausible performance. Despite having the best attack performance, GSPAttack-2 has the worst recommendation performance (Fig. 5-b) on all datasets without the constraints to maintain plausible performance. GSPAttack-2 demonstrates a strong capacity for attack performance because its underlying model mainly focuses on boosting the visibility of target items—optimising the explicit boosting loss during the attack process.

Effect of unnoticeability. We use GSPAttack-3 and GSPAttack-4 to quantitatively assess the ability of attack methods with or without deploying the unnoticeable constraint, respectively. We want to compare these variants' performance in handling with the defense method. We see that, without the constraints of unnoticeable, the attack performance of GSPAttack-4 significantly drops, indicating that the manipulation of some fake users is disabled from the attack process. To further verify the contribution of unnoticeable constraints, we visualise the user embeddings of GSPAttack-3 (Fig. 6-b) and GSPAttack-4 (Fig. 6-a) on FR dataset with the attack size

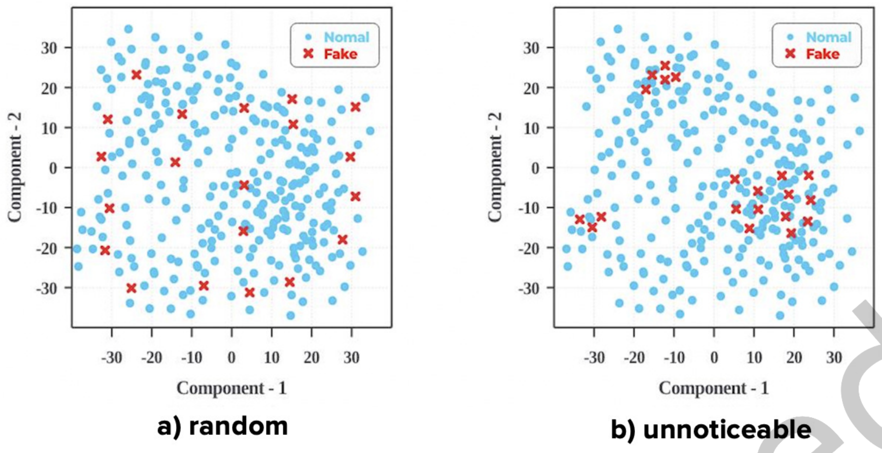


Fig. 6. Effect on noticeability.

($\mu = 2\%$). It is worth noting that closely normal users have been sampled to facilitate a better visualisation. We can see in Fig. 6 that GSPAttack-3 produces closer embeddings between the fake users and benign ones. In contrast, GSPAttack-4 generates many isolated users, which are easily filtered by an anomaly detection system. This qualitative evaluation further supports the degradation of the attack performance of GSPAttack-4.

Table 1. Transferability capacity.

	ER@10 (%)	HR@10 (%)	Pres@10 (%)
HashGNN [51]	**92.6%**	42.66%	92.75%
AGCN [59]	89.3%	43.29%	93.19%
LightGCN [25]	86.1%	**49.20%**	**95.12%**

7.7 Transferable capacity

To investigate the transferability capacity of GSPAttack, we replaced the victim model with the three most prevalent GNN-based RSs. The experiments are conducted on the ML dataset (2% attack size), and the attack and recommendation performance are recorded after 100 epochs and presented in Table 1 (the results for other datasets are omitted as they reveal the same characteristics). We can see that the attack effectiveness of GSPAttack is best transferred to HashGNN [51]. The possible reason is that HashGNN inherited the same paradigm as GNN architecture with a minor adjustment to the feature aggregation (i.e., using mean-pooling aggregation). AGCN [59] pursues traceability in their model by replacing the non-linear activation function with a linear function. Therefore, their transferred capability witnessed a decrease in performance but not significantly. LightGCN [25] removes both the original architecture's activation and transformation; therefore, it achieves a moderate result in a transferred attack. However, it outperforms other techniques in terms of recommendation performance since it uses a light architecture that converges faster. Besides, we see that our surrogate model shows a reasonable high precision in reproducing the results of the victim model. This finding is in line with prior literature [25, 27, 40] on transferability among models with similar functionalities.

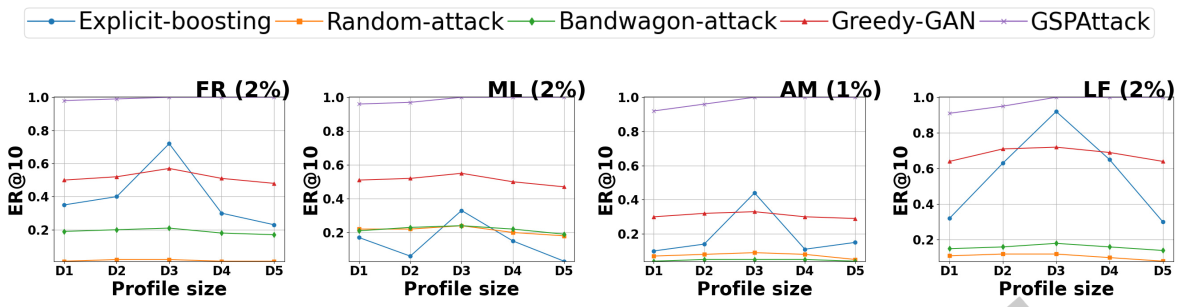


Fig. 7. Impact of profile size.

7.8 Impact of Profile Size

Given a specific attack size, the profile size with which each fake user interacts is another essential factor in an attack method. In this section, we explore the impact on attack performance by varying the profile size using five different levels $D_i = \{60\%, 80\%, 100\%, 120\%, 140\%\}$ of the average degree D_{avg}. From Fig. 7, we see a decrease in the attack performance of competing baselines when the profile size exceeds a certain threshold. While the finding appears to be counterintuitive, previous research on attacks against explicit feedback RSs discovers similar results [18, 35]. The possible reason could be that the number of "good" filler items is limited, and employing too many filler items could introduce adverse effects as they introduce extra noise to the target model. GSPAttack, on the other hand, is stable and effectively promotes the target item at different levels of profile size.

7.9 Influence of Hyperparameters

We answer **RQ5** by investigating the influence of hyperparameters on the attack performance over all datasets. The parameter is varied by setting it to different values of 20, 40, 60, 80, and 100, and the results are shown in Fig. 8. It is worth noting that the influence of β is omitted as it causes fewer variations in the model performance. The key finding is that the model delivers the best results when we put adequate emphasis on the explicit boosting loss to avoid the dilution effect caused by a significant number of normal items. For example, the framework performs the best at $\alpha = 60$ and $\alpha = 80$ in FR and ML datasets, respectively. Over-emphasising on the explicit boosting loss may disable the effects of other optimisation objectives, leading to attack performance degradation.

7.10 Discussion

To enhance the security of RSs, in this work, we propose GSPAttack as a practical tool for evaluating the robustness of GNN-based RSs against poison attacks. Regarding the end-to-end comparison, we empirically found that GSPAttack outperformed the competing baselines in attack performance while achieving comparable recommendation performance with other techniques. From comprehensive ablation testing, we found that the interplay of components of our proposed framework is the best combination compared to many different variants. The key finding regarding *transferable capability* is that our approach is generic and compatible with many other GNN-based RSs including HashGNN [51], AGCN [59], and LightGCN [25]. Not surprisingly, GSPAttack has been affected by the changes in profile size and hyperparameters; however, it is more stable and effective than other baselines in achieving the attack goals. These findings provide solid practical implications for different stakeholder groups from academia and industry to utilise GSPAttack as the core for developing appropriate countermeasures.

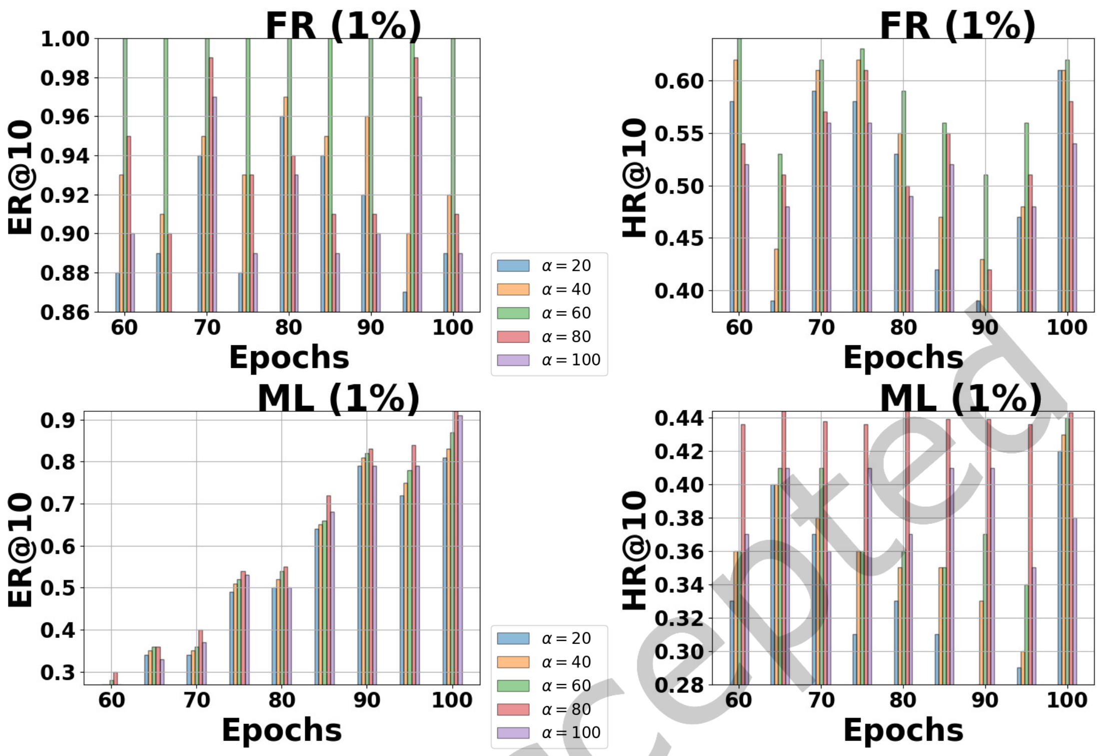


Fig. 8. Influence of hyperparameters.

8 RELATED WORKS

We first discuss the development of RSs, and then we discuss poison attacks in general and other works related to poison attacks in RSs.

8.1 Development of Recommender Systems

Recommender systems (RSs)—an integral part of the online platform to predict users' preferences[2]— have received plenty of attention in recent years. *Neighbourhood-based* RSs [43] recommend items based on the computing similarities between users or items. *Matrix-factorisation-based* RSs [11, 31] assume that a small number of latent factors are sufficient to represent the users' preferences and try to extract such latent factors using low-rank matrix factorization techniques. *Association-rule-based* RSs [14] apply the idea of finding the co-occurrence patterns in items based on users' past preferences and use this trait to suggest items to new users. With the advancement of a deep neural network, *deep-learning-based RSs* [70] adopt a variety of different deep learning building blocks—including Multilayer Perceptron [26], Autoencoder [63], Deep Reinforcement Learning [66, 70], and Adversarial Network [20]—to model the collaborations between users and items. *Graph-neural-network* based RSs exploit the high-order structure information in the user-item graph and have recently achieved state-of-the-art recommendation results. In this work, we focus on GNN-based RSs, and develop a poison attack tool to assess the robustness of a GNN-based RS.

8.2 General Poison Attacks

A poison attack is a task of manipulating a machine model towards the attacker's desire by injecting well-crafted data during the model updates. They have been intensively studied against a wide range of machine learning models, including support vector machine [5], decision tree [7], regression models [6] deep neural networks [19], and graph neural network [9, 46, 47, 53]. While the data input is the main factor contributing to the success of a poison attack, the data input of an RS reveals distinct characteristics (i.e., heterogeneity, data correlation between users and items) that neglect attacks from other domains being directly applied to the RS domain at full power. With the advance of deep learning, many poison attacks [10, 76, 77] have been proposed to attack the node classification task in a standard GNN setting. However, these attack strategies cannot be directly applied to RS's domain due to privacy concerns. To fill this gap, this study explores a feasible and practical poison attack approach for a particular type of RS—the GNN-based RS.

8.3 Surrogate-based Attacks in RSs.

Surrogate-based attacks in RSs are primarily proposed to steal black box model weights to replicate a lived system [27, 54, 68, 69, 74]. Existing studies classified surrogate-based attacks in RSs into three main categories: *model extraction*, *item proximity*, and *representative model*. *Model extraction [68]* methods aim to extract the target model weights and idea often rooted in computer vision research [34]. *Item proximity [74]* methods estimate the item proximities which are learned by the target model. However, these attacks are suitable for item-based RS, which neglects the data correlation between users and items. *Representative model [69]* methods select a known representative model as the surrogate model, then transfer these attacks to other models that have a similar function to the target RS [27]. Following the recent success of the representative model category, a well-known GNN-based RS—Neural Graph Collaborative Filtering (NGCF) [58]—is chosen as the surrogate model for training the attack and then transfers the attacks to other GNN-based RSs such as HashGNN [51], AGCN [59], and LightGCN [25].

8.4 Poison Attacks in RSs.

Poison attacks in RSs have been intensively studied in recent years [44]. Pioneer works in RS poison attacks [35, 48] are *agnostic* to the recommendation models. These attacks focus on generalization capabilities rather than being specially designed for a specific recommendation algorithm. Although they can be applied to a wide range of classes of recommendation algorithms, they oftentimes achieve suboptimal performance [48]. To tackle this limitation, recent proposed attacks are optimised for a particular class of RSs, including matrix-factorisation-based [32], neighbourhood-based [12], graph-based [18], deep-learning-based RSs [27]. For example, Li et al. [32] attack method verifies the success of two popular factorisation-based CF algorithms—the *alternative minimisation* formulation and the *nuclear norm minimisation* technique. Furthermore, with full knowledge about the targeted system, the authors demonstrate how powerful the attacker can generate fake data to maximise the attacker's goals while at the same time being unnoticeable by mimicking normal users' behaviours. In more recent work, Chen et al. [12] proposed framework named UNAttack, which follows a similar theme to other model-instrinsic poison attacks —injecting well-crafted fake users into the target models in such a way that the target items are suggested to as many regular users as possible. In addition, the work discovers some interesting findings: (i) RSs with Euclide-based similarity have strong robustness and (ii) UNAttack can be transferable to other RSs such as neural CF and Bayesian personalised ranking systems. Given the advances in graph data structure [33], Fang et al. [18] aim to design a poisoning attack for a graph-based recommender system, users' rating scores are represented by a *user preference graph*. In the preference graph, a node represents a user or an item, and an edge weight represents a rating score of a certain user given to an item. Similar to most existing attacks, the authors carefully craft fake users together with their rating scores, and inject them into the targeted RS to manipulate the model towards the attack goal. In addition, determining the fake rating scores is formulated as an optimisation problem to maximise the attack impact. The determined score further supports the attack to disguise the defense methods. With the recent advancements in deep learning (DL), Huang et al. [27] propose a poison attack on DL-based RSs by injecting fake users with carefully crafted ratings into the targeted model. The work shows that this attack can be defined as an optimisation problem that can be approximated and resolved efficiently by incorporating many heuristic rules. Interestingly, the attack model is still practical when the attackers have limited access to only a tiny portion of user-item interactions. Similar to our work, Wu et al. [62] have recently applied surrogate-based attacks to RSs. Nevertheless, they exploit an auxiliary third-party data source (i.e., knowledge graph data); and therefore, their problem setting is orthogonal to our work.

To the best of our knowledge, there is no existing poison attack that can be applied to GNN-based RS and transferred to different variants of GNN-based RS models to serve as a benchmarking tool for model robustness evaluation.

9 CONCLUSION

This paper proposes GSPAttack, an end-to-end poison attack framework against GNN-based RSs. GSPAttack comprises three steps and starts by generating unnoticeable fake users. Then, the malicious edges connecting fake users and popular items are identified to form the attacked graph. Finally, the attacked graph is fed into a surrogate model to train for discovering the best-perturbed version before being transferred to attack the real victim model. Our experiments were conducted with four real-world datasets in various domains against four different baselines, and the results illustrated that our generative surrogate-based poison attack is:

- *Efficient*: despite a small number of fake users, the victim model successfully recommends the target to all users (Fig. 3). More precisely, GSPAttack achieved an average improvement of 46% in attack performance over the best baseline.
- *Transferable:* GSPAttack's efficacy is fully transferred to the most popular GNN-based RSs (Table 1). In particular, our approach is generic and compatible with many other GNN-based RSs including HashGNN [51], AGCN [59], and LightGCN [25].
- *Unnoticeable*: it is the best method for dealing with defense methods when compared to other variants (Fig. 5). Particularly, the attack performance of GSPAttack significantly drops without the constraints of unnoticeable.

For future works, we intend to extend our framework to more settings. In particular, how can we perform our framework in universal settings of GNN-based RSs such as Graph AutoEncoder [71, 73, 75]? Second, by using GSPAttack as a robustness evaluating tool, we will explore the feasibility of developing a defense against poison attacks on GNN-based RSs. In addition, we will apply active learning for fake users and malicious edge generations more effectively. Finally, we identify one limitation in our proposed framework: it is less effective on the "cold start" problem. Although this is the standard limitation in the literature on poison attacks [52, 74], we intend to provide more theoretical insights into the field of information systems, which motivate other stakeholders to understand this type of limitation further and develop more effective attack strategies.

ACKNOWLEDGMENTS

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE200101465).

REFERENCES

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286* (2019).
[2] Charu C Aggarwal et al. 2016. *Recommender systems*. Vol. 1. Springer.

[3] Mehmet Aktukmak, Yasin Yilmaz, and Ismail Uysal. 2019. Quick and accurate attack detection in recommender systems through user attributes. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 348–352.

[4] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014* (2015).

[5] Giuseppe Bonaccorso. 2017. *Machine learning algorithms*. Packt Publishing Ltd.

[6] Paula Branco, Luís Torgo, and Rita P Ribeiro. 2016. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)* 49, 2 (2016), 1–50.

[7] Mr Brijain, R Patel, MR Kushik, and K Rana. 2014. A survey on decision tree algorithm for classification. (2014).

[8] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[9] Taotao Cai, Jianxin Li, Ajmal S Mian, Timos Sellis, Jeffrey Xu Yu, et al. 2020. Target-aware holistic influence maximization in spatial social networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[10] Yi Chang, Zhao Ren, Thanh Tam Nguyen, Wolfgang Nejdl, and Björn W Schuller. 2022. Example-based Explanations with Adversarial Attacks for Respiratory Sound Analysis. In *Interspeech*. 1–5.

[11] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[12] Liang Chen, Yangjun Xu, Fenfang Xie, Min Huang, and Zibin Zheng. 2021. Data poisoning attacks on neighborhood-based recommender systems. *Transactions on Emerging Telecommunications Technologies* 32, 6 (2021), e3872.

[13] Tong Chen, Hongzhi Yin, Hongxu Chen, Lin Wu, Hao Wang, Xiaofang Zhou, and Xue Li. 2018. Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. In *2018 IEEE international conference on data mining (ICDM)*. 49–58.

[14] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. 293–296.

[15] Jingtao Ding, Guanghui Yu, Yong Li, Xiangnan He, and Depeng Jin. 2020. Improving implicit recommender systems with auxiliary data. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2020), 1–27.

[16] Chi Thang Duong, Thanh Dat Hoang, Ha The Hien Dang, Quoc Viet Hung Nguyen, and Karl Aberer. 2019. On node features for graph neural networks. *arXiv preprint arXiv:1911.08795* (2019).

[17] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[18] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 381–392.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[21] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. 2014. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review* 42, 4 (2014), 767–799.

[22] Lei Guo, Hongzhi Yin, Tong Chen, Xiangliang Zhang, and Kai Zheng. 2021. Hierarchical hyperedge embedding-based representation learning for group recommendation. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–27.

[23] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[24] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[25] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[26] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[27] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data poisoning attacks to deep learning based recommender systems. *arXiv preprint arXiv:2101.02644* (2021).

[28] Xiaowen Huang, Jitao Sang, Jian Yu, and Changsheng Xu. 2022. Learning to learn a cold-start sequential recommender. *ACM Transactions on Information Systems (TOIS)* 40, 2 (2022), 1–25.

[29] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).

[30] Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*. PMLR, 3499–3508.

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

[32] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* 29 (2016).

[33] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)* 51, 3 (2018), 1–34.

[34] Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 641–647.

[35] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. 2007. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology (TOIT)* 7, 4 (2007), 23–es.

[36] Tam Thanh Nguyen, Thanh Trung Huynh, Hongzhi Yin, Vinh Van Tong, Darnbi Sakong, Bolong Zheng, and Quoc Viet Hung Nguyen. 2020. Entity alignment for knowledge graphs with multi-order convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[37] Thanh Toan Nguyen, Thanh Tam Nguyen, Thanh Thi Nguyen, Bay Vo, Jun Jo, and Quoc Viet Hung Nguyen. 2021. Judo: Just-in-time rumour detection in streaming social platforms. *Information Sciences* 570 (2021), 70–93.

[38] Thanh Toan Nguyen, Minh Tam Pham, Thanh Tam Nguyen, Thanh Trung Huynh, Quoc Viet Hung Nguyen, Thanh Tho Quan, et al. 2021. Structural representation learning for network alignment with self-supervised anchor links. *Expert Systems with Applications* 165 (2021), 113857.

[39] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 188–197.

[40] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[41] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–23.

[42] Fatemeh Rezaimehr and Chitra Dadkhah. 2021. A survey of attack detection approaches in collaborative filtering recommender systems. *Artificial Intelligence Review* 54, 3 (2021), 2011–2066.

[43] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[44] Mingdan Si and Qingshan Li. 2020. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review* 53, 1 (2020), 291–319.

[45] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 157–168.

[46] Xiangyu Song, Jianxin Li, Qi Lei, Wei Zhao, Yunliang Chen, and Ajmal Mian. 2022. Bi-CLKT: Bi-graph contrastive learning based knowledge tracing. *Knowledge-Based Systems* 241 (2022), 108274.

[47] Xiangyu Song, Jianxin Li, Yifu Tang, Taige Zhao, Yunliang Chen, and Ziyu Guan. 2021. Jkt: A joint graph convolutional network based deep knowledge tracing. *Information Sciences* 580 (2021), 510–523.

[48] Agnideven Palanisamy Sundar, Feng Li, Xukai Zou, Tianchong Gao, and Evan D Russomanno. 2020. Understanding shilling attacks and their detection traits: A comprehensive survey. *IEEE Access* 8 (2020), 171703–171715.

[49] Nguyen Thanh Tam, Huynh Thanh Trung, Hongzhi Yin, Tong Van Vinh, Darnbi Sakong, Bolong Zheng, and Nguyen Quoc Viet Hung. 2021. Entity alignment for knowledge graphs with multi-order convolutional networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2323–2324.

[50] Nguyen Thanh Tam, Matthias Weidlich, Duong Chi Thang, Hongzhi Yin, and Nguyen Quoc Viet Hung. 2017. Retaining Data from Streams of Social Platforms with Minimal Regret. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2850–2856.

[51] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to hash with graph neural networks for recommender systems. In *Proceedings of The Web Conference 2020*. 1988–1998.

[52] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Fourteenth ACM conference on recommender systems*. 318–327.

[53] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single Node Injection Attack against Graph Neural Networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1794–1803.

[54] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*. 601–618.

[55] Huynh Thanh Trung, Tong Van Vinh, Nguyen Thanh Tam, Hongzhi Yin, Matthias Weidlich, and Nguyen Quoc Viet Hung. 2020. Adaptive network alignment with unsupervised and multi-order convolutional networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 85–96.

[56] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2022. Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal* 31, 5 (2022), 877–896.

[57] Xiaoyun Wang, Minhao Cheng, Joe Eaton, Cho-Jui Hsieh, and Felix Wu. 2018. Attack graph convolutional networks by adding fake nodes. *arXiv preprint arXiv:1810.10751* (2018).

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

[59] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. 2020. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 679–688.

[60] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260* (2020).

[61] Zhiang Wu, Changsheng Li, Jie Cao, and Yong Ge. 2020. On Scalability of Association-rule-based recommendation: A unified distributed-computing framework. *ACM Transactions on the Web (TWEB)* 14, 3 (2020), 1–21.

[62] Zih-Wun Wu, Chiao-Ting Chen, and Szu-Hao Huang. 2022. Poisoning attacks against knowledge graph-based recommendation systems using deep reinforcement learning. *Neural Computing and Applications* 34, 4 (2022), 3097–3115.

[63] Lianghao Xia, Chao Huang, Yong Xu, Huance Xu, Xiang Li, and Weiguo Zhang. 2021. Collaborative Reflection-Augmented Autoencoder Network for Recommender Systems. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–22.

[64] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)* 37, 3 (2019), 1–25.

[65] Jun Yang, Weizhi Ma, Min Zhang, Xin Zhou, Yiqun Liu, and Shaoping Ma. 2021. LegalGNN: Legal Information Enhanced Graph Neural Network for Recommendation. *ACM Transactions on Information Systems (TOIS)* 40, 2 (2021), 1–29.

[66] Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2021. RLPS: A Reinforcement Learning–Based Framework for Personalized Search. *ACM Transactions on Information Systems (TOIS)* 39, 3 (2021), 1–29.

[67] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial collaborative neural network for robust recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1065–1068.

[68] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *Fifteenth ACM Conference on Recommender Systems*. 44–54.

[69] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and Jing Gao. 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*. 2154–2164.

[70] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[71] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. 2021. Graph embedding for recommendation against attribute inference attacks. In *Proceedings of the Web Conference*. 3002–3014.

[72] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. 2022. PipAttack: Poisoning Federated Recommender Systems for Manipulating Item Promotion. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1415–1423.

[73] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 689–698.

[74] Yihe Zhang, Xu Yuan, Jin Li, Jiadong Lou, Li Chen, and Nian-Feng Tzeng. 2021. Reverse Attack: Black-box Attacks on Collaborative Recommendation. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 51–68.

[75] Qi Zhou, Yizhi Ren, Tianyu Xia, Lifeng Yuan, and Linqiang Chen. 2019. Data poisoning attacks on graph convolutional matrix completion. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 427–439.

[76] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1399–1407.

[77] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.