Sharpness-Aware Poisoning: Enhancing Transferability of Injective Attacks on Recommender Systems

Junsong Xie, Yonghui Yang, Pengyang Shao, and Le Wu, *Member, IEEE*

***Abstract*—Recommender Systems (RS) have been shown to be vulnerable to injective attacks, where attackers inject limited fake user profiles to promote the exposure of target items to real users for unethical gains (e.g., economic or political advantages). Since attackers typically lack knowledge of the victim model deployed in the target RS, existing methods resort to using a fixed surrogate model to mimic the potential victim model. Despite considerable progress, we argue that the assumption that *poisoned data generated for the surrogate model can be used to attack other victim models* is wishful. When there are significant structural discrepancies between the surrogate and victim models, the attack transferability inevitably suffers. Intuitively, if we can identify the worst-case victim model and iteratively optimize the poisoning effect specifically against it, then the generated poisoned data would be better transferred to other victim models. However, exactly identifying the worst-case victim model during the attack process is challenging due to the large space of victim models. To this end, in this work, we propose a novel attack method called Sharpness-Aware Poisoning (*SharpAP*). Specifically, it employs the sharpness-aware minimization principle to seek the approximately worst-case victim model and optimizes the poisoned data specifically for this worst-case model. The poisoning attack with *SharpAP* is formulated as a min-max-min tri-level optimization problem. By integrating *SharpAP* into the iterative process for attacks, our method can generate more robust poisoned data which is less sensitive to the shift of model structure, mitigating the overfitting to the surrogate model. Comprehensive experimental comparisons on three real-world datasets demonstrate that *SharpAP* can significantly enhance the attack transferability.**

***Index Terms*—Recommender systems, poisoning attacks, sharpness-aware optimization, adversarial transferability.**

Received 11 May 2025; revised 17 January 2026; accepted 4 April 2026. Date of publication 13 April 2026; date of current version 2 June 2026. This work was supported in part by the National Natural Science Foundation of China under Grant U23B2031 and Grant 62436003 and in part by the Fundamental Research Funds for the Central Universities under Grant JZ2025HGPB0248 and Grant PA2025IISL0106. Recommended for acceptance by J. Tang. *(Corresponding author: Le Wu.)*

Junsong Xie is with the Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology, Hefei 230009, China, and also with the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230009, China (e-mail: jsxie.hfut@gmail.com).

Yonghui Yang is with the Laboratory of NExT++, National University of Singapore, Singapore 119077, and also with the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230009, China (e-mail: yh_yang@nus.edu.sg).

Pengyang Shao and Le Wu are with the Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology, Hefei 230009, China (e-mail: shaopymark@gmail.com; lewu.ustc@gmail.com).

Digital Object Identifier 10.1109/TKDE.2026.3682785

I. Introduction

In THE digital age, Recommender Systems (RS) have emerged as indispensable tools for mitigating information overload across diverse platforms [42], [43], [45], [50], from e-commerce (e.g., Amazon, Taobao) to social media (e.g., Twitter, Xiaohongshu). By leveraging user behaviors such as purchase history and content consumption patterns, RS infers users' latent preferences and then recommends potentially interesting items. However, their growing significance has also drawn attention from adversarial entities seeking to exploit their vulnerabilities [29], [35], [36]. RS typically utilizes Collaborative Filtering (CF) to provide recommendations, meaning that users' recommendations are influenced not only by their own historical behaviors but also by interactions from other users. While this principle enhances recommendation accuracy by leveraging collective user preferences, it also introduces security risks [31], [32], [33]. Specifically, the openness of RS and collaborative patterns create opportunities for *injective attacks* [18], [34], [48].

In injective attacks, attackers take advantage of the system's openness by registering a limited number of fake user accounts and then designing their profiles elaborately [18], [21], [22]. As illustrated in Fig. 1, these fake user profiles, combined with real ones, form the *poisoned data*. This poisoned data is then used to attack the black-box victim model deployed in the target RS, manipulating it to produce recommendations that align with the attackers' goals [6], [47]. For example, full-user attacks aim to promote the recommendation probability of the target item across all real users [8], [36], whereas group attacks focus on a specific group [31]. To carry out injective attacks, existing works have proposed three main categories of attack methods: heuristic-based, neural network-driven, and gradient-based attacks. Heuristic-based attacks rely on manually crafted rules to design fake user profiles [1], [44]. Neural network-driven attacks optimize the parameters of neural networks to generate influential fake user behaviors to achieve the attack objective [5], [38]. Gradient-based attacks maximize the sophisticated attack objective by optimizing parameterized fake profiles via gradients [6], [18], [47].

Gradient-based attacks have emerged as the predominant methods for injective attacks, owing to their demonstrated performance advantages [3], [18]. As illustrated in Fig. 2, the victim model deployed in the target RS is typically invisible to attackers, making it challenging for gradient-based attacks to obtain its

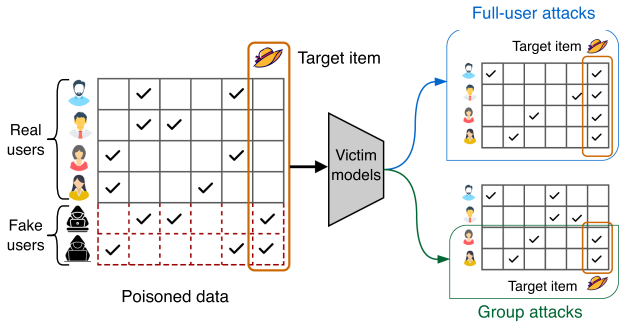


Fig. 1. Illustration of injective attacks on full-users and group users. Full-user attacks aim to increase the exposure probability of the target item among all real users, whereas group attacks focus on a specific group (e.g., the female user group).

gradients directly. To circumvent this limitation, they use a fixed surrogate model to mimic the potential victim model. The generated poisoned data is then expected to be effective against various unseen victim models. This "one poisons all" strategy, however, is crucially dependent on a precarious assumption: *Poisoned data generated for the surrogate model can be used to attack other victim models*. Existing methods relying on this greedy assumption neglect structural discrepancies between the surrogate and victim models [18], [29], [47]. This oversight fundamentally limits the transferability of their attacks. Our empirical validation in Fig. 2 also substantiates this limitation. Specifically, we use the representative gradient-based method RevAdv [29] as an example to inject fake users into a popular dataset Movielens-1M [9]. RevAdv employs WRMF [14] as the surrogate model. The objective is to increase the recommendation probability of five randomly selected target items across all real users. To evaluate the effectiveness of the attack, we adopt the Hit Ratio at 10 (HR@10) as the evaluation metric. Since we cannot control the retraining process of the poisoned data, it may be used in various victim models, such as the surrogate WRMF [14], the pairwise ranking-based BPR [26], and the GCN-based models (LightGCN [11] and SGL [41]). Baseline performance (denoted as "Clean") represents the results on the unattacked dataset. While effective against WRMF (↑32% compared to "Clean"), the poisoned data exhibits poor transferability in BPR (↑16%), LightGCN (↑22%), and SGL (↑18%). Fig. 2 verifies that relying solely on the inherent transferability of poisoned data is precarious and exhibits overfitting to the surrogate model. We should explicitly accommodate model structure shifts to enhance the transferability.

Intuitively, if we can iteratively optimize the poisoning effect against the **worst-case model** (the victim RS model that has the worst poisoning effect) rather than a fixed surrogate model, the generated poisoned data would be better transferred to other victim models [10]. However, exactly identifying the worst-case model among a large space of potential victim models during the attack process is computationally intractable. Meanwhile, sharpness-aware minimization has demonstrated notable success in enhancing model generalization by seeking the maximum loss within a local neighborhood of model parameters and then minimizing it [2], [7], [10]. Motivated by this insight, we propose a novel attack method called **Sharp**ness-**A**ware **P**oisoning (*SharpAP*), which seeks the approximately worst-case model via sharpness-aware minimization principle. Specifically, *SharpAP* reformulates the attack process by extending the original min-min bi-level optimization into a min-max-min tri-level optimization problem. The introduced maximization step performs a bounded perturbation to seek the worst-case model, leveraging the sharpness-aware minimization principle. Distinct from existing methods that generate poisoned data for a fixed surrogate model (i.e., without the maximization step), *SharpAP* dynamically targets the worst-case victim model, thereby mitigating the overfitting to the surrogate model. Crucially, based on the characteristics of RS models, we provide a theoretical analysis of the transferability of sharpness-aware attacks. In summary, our key contributions are as follows:

- To the best of our knowledge, we are the first to point out the challenge of overfitting to the surrogate model in injective attacks. Our findings reveal that when the victim model's structure shifts, the transferability of the poisoning effect inevitably suffers.
- To overcome this challenge, we propose *SharpAP*, which introduces sharpness-aware minimization principle to approximate the worst-case model. We then generate poisoned data for this worst-case model instead of a fixed surrogate model. Furthermore, we provide a theoretical analysis demonstrating that optimizing poisoned data against worst-case models can improve attack transferability.
- Technically, *SharpAP* formulates injective attacks as a sharpness-aware min-max-min tri-level optimization problem. Moreover, *SharpAP* can be seamlessly integrated with many existing gray-box attack methods to enhance the transferability.
- We conduct comprehensive experimental studies and demonstrate that *SharpAP* significantly enhances the transferability of many representative victim models.

II. Preliminaries

A. Recommender System under Injective Attack

In a recommender system under injective attack, there are three entity sets: a real user set $U^r = \{u_1^r, u_2^r, ..., u_{|U^r|}^r\}$, a fake user set $U^f = \{u_1^f, u_2^f, ..., u_{|U^f|}^f\}$ and an item set $V = \{v_1, v_2, ..., v_{|V|}\}$. User-item interactions are recorded as feedback data, categorized into explicit feedback (e.g., direct ratings like 5-star scores) and implicit feedback (e.g., indirect signals such as purchases or views). Due to the prevalence of implicit feedback in real-world applications, we focus on this type in our work. To represent user-item interactions, we define a binary matrix $\mathbf{R} = \begin{bmatrix} \mathbf{R}^r \\ \mathbf{R}^f \end{bmatrix} \in \{0,1\}^{(|U^r|+|U^f|)\times|V|}$, where $r_{uv} = 1$ indicates positive feedback from user u on item v, and $r_{uv} = 0$ denotes an unknown interaction. RS leverages historical user-item interactions to predict a relevance score matrix $\hat{\mathbf{R}} \in \mathbb{R}^{(|U^r|+|U^f|)\times|V|}$, where higher scores indicate stronger relevance between a user and an item. The objective of RS is

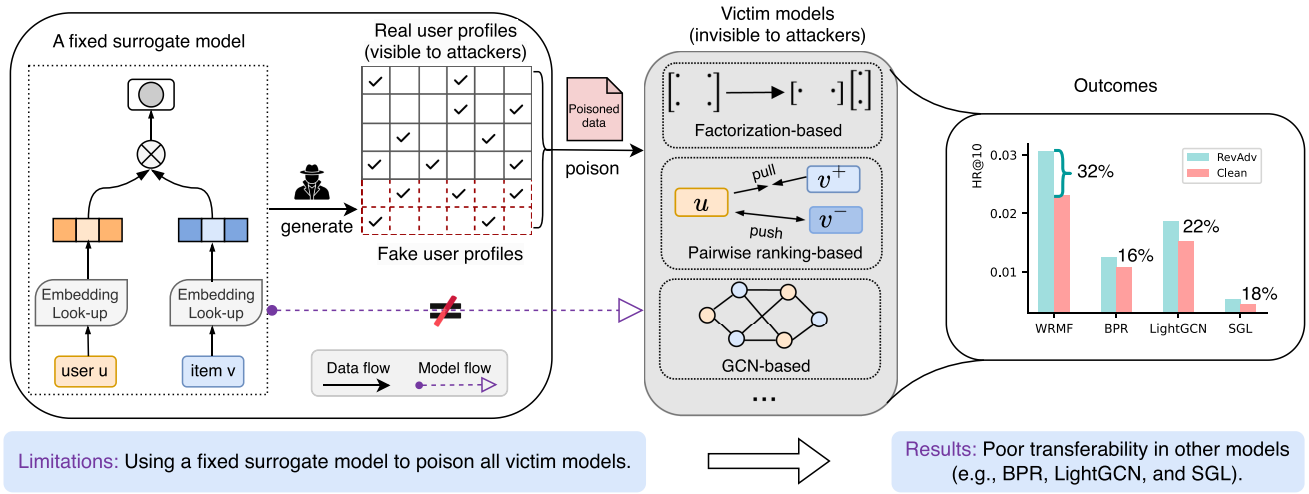


Fig. 2. Illustration of the origin of our work's motivation. Since the victim model is inaccessible to attackers, existing methods use a fixed surrogate model to mimic it for generating poisoned data. These methods first ensure the effectiveness of poisoned data on the surrogate model. Then, they hope it will be equally effective on all unseen victim models. We argue that these methods neglect structural discrepancies between the surrogate and victim models, which inevitably leads to impaired attack transferability. The experimental results presented on the right highlight the limitations. Specifically, the RevAdv attack [29] exhibits a good attack performance on the surrogate WRMF [14]. However, when the victim models are BPR [26], LightGCN [11], and SGL [41], the performance gains from the attack are significantly diminished, indicating poor transferability. "Clean" refers to the results on the unattacked dataset.

to rank items for each user based on these predicted relevance scores, ensuring that the most relevant items are prioritized.

B. Injective Attacks on Recommender System

In this section, we first present the formal definition of injective attacks. Then, we revisit the strategy employed by gradient-based attack methods under a gray-box setting. Given the real data $\mathbf{R}^r \in \{0,1\}^{|U^r|\times|V|}$, the target RS (a.k.a. victim model) $\mathcal{M}^v$, and a limited fake users $U^f = \{u_1^f, u_2^f, ..., u_{|U^f|}^f\}$, the corresponding fake data $\mathbf{R}^f \in \{0,1\}^{|U^f|\times|V|}$ can be learned to minimize the attack loss function $\mathcal{L}_{atk}$ as follows:

$$\min_{\mathbf{R}^f} \mathcal{L}_{atk}(\mathcal{M}^v(\theta^*; \mathbf{R}^r)), \tag{1}$$

$$\text{s.t.} \quad \theta^* = \arg\min_{\theta}(\mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^v(\theta; \mathbf{R})), \tag{2}$$

where $\mathbf{R} = \begin{bmatrix} \mathbf{R}^r \\ \mathbf{R}^f \end{bmatrix}$. $\mathcal{L}_{rec}$ is the recommendation training loss. $\mathcal{M}^v(\theta; \mathbf{R})$ and $\mathcal{M}^v(\theta^*; \mathbf{R}^r)$ represent the predictions of the victim model $\mathcal{M}^v$ for all users and real users, respectively, given parameters θ and θ^*. For clarity, we omit the attacker's capability constraints, $|U^f| \leq \delta|U^r|$ and $||\mathbf{R}^f[u^f]||_0 \leq N$, to keep the formula concise. δ represents the proportion of fake users to real users, and N denotes the maximum number of items each fake user can interact with (i.e., profile size). It is a non-trivial task to generate the exact optimal fake data $\mathbf{R}^f$ that achieves the minimum attack loss, owing to the exponential candidate search space $\mathcal{O}(|V|^{N\cdot|U^f|})$. To obtain an approximate optimal solution, there are three types of attack methods: heuristic-based, neural network-driven, and gradient-based attacks. In this paper, we focus on gradient-based attacks due to their demonstrated effectiveness in prior studies. We perform attacks under a gray-box setting, where attackers have access only to the real data $\mathbf{R}^r$, but no knowledge of the victim model $\mathcal{M}^v$. This setting is reasonable, as proprietary recommendation algorithms are rarely disclosed by platforms. Nevertheless, attackers can obtain user behavior data through open APIs or by scraping public user profiles [18], [31]. Under a gray-box setting, existing gradient-based attack methods use a fixed surrogate model $\mathcal{M}^s$ to replace the inaccessible victim model $\mathcal{M}^v$ in (1) and (2). After the replacement, the inner optimization (i.e., (2)) aims to mimic the retraining process with the poisoned data $\mathbf{R}$ using the surrogate model $\mathcal{M}^s$. The outer optimization (i.e., (1)) is responsible for optimizing the fake data $\mathbf{R}^f$ to fulfill the attack goals. Fig. 3 shows the flow chart of fake profiles instantiated. Although solving such a bi-level optimization problem ensures satisfactory attack performance on the surrogate model, the transferability of the attack to other victim models remains uncertain and largely uncontrolled.

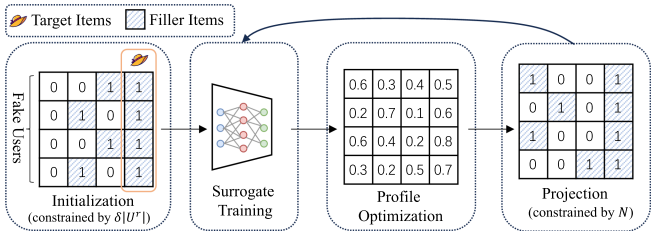


Fig. 3. A flow chart of fake profiles instantiated.

III. Methodology

A. The Objective of SharpAP

We begin by theoretically defining the victim model space, denoted as Ω. This space encompasses all potential victim models whose objective is to minimize the recommendation

training loss $\mathcal{L}_{rec}$ on the poisoned data $\mathbf{R}$:

$$\Omega = \{\mathcal{M}^v | \mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^v(\mathbf{R})) \leq \rho\}, \tag{3}$$

where $\mathcal{M}^v(\mathbf{R})$ represents the predictions of the victim model $\mathcal{M}^v$ for all users. Mathematically, the victim model space Ω is defined as the set of all models such that the recommendation loss does not exceed a predefined threshold ρ. This formulation ensures that Ω includes only those models that are well-trained on the poisoned data. Then, we identify the worst-case model $\mathcal{M}^{worst}$ in the space Ω by maximizing the attack loss $\mathcal{L}_{atk}$. The worst-case model $\mathcal{M}^{worst}$ can be formulated as:

$$\mathcal{M}^{worst} = \arg\max_{\mathcal{M}^v \in \Omega} \mathcal{L}_{atk}(\mathcal{M}^v(\mathbf{R}^r)). \tag{4}$$

According to (4), we can obtain:

$$\mathbb{E}\left[\mathcal{L}_{atk}(\mathcal{M}^v(\mathbf{R}^r))\right] \leq \mathcal{L}_{atk}(\mathcal{M}^{worst}(\mathbf{R}^r)). \tag{5}$$

Equation (5) states that $\mathcal{L}_{atk}(\mathcal{M}^{worst}(\mathbf{R}^r))$ is the upper bound of the expected attack loss over all victim models. By optimizing the fake data $\mathbf{R}^f$ to minimize this upper bound through:

$$\min_{\mathbf{R}^f} \mathcal{L}_{atk}(\mathcal{M}^{worst}(\mathbf{R}^r)), \tag{6}$$

we aim to reduce the average attack loss across all victim models. In other words, improving the poisoning effect on the worst-case model $\mathcal{M}^{worst}$ facilitates the propagation of the poisoning effect to other victim models, thereby enhancing the attack transferability.

However, directly solving (3) and (4) is computationally intractable, and it is challenging to obtain an exact solution. To address this, we leverage sharpness-aware minimization principles to approximate the worst-case solution [2], [7], [10]. Specifically, we approximate the worst-case model by performing a localized search in the neighborhood of the surrogate model, formulated as:

$$\mathcal{L}_{atk}(\mathcal{M}^{worst}(\mathbf{R}^r)) \approx \max_{||\theta^\Delta||_p \leq \epsilon} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^* + \theta^\Delta; \mathbf{R}^r)),$$
$$\text{s.t.} \theta^* = \arg\min_{\theta}(\mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^s(\theta; \mathbf{R}))). \tag{7}$$

As formulated in (7), we first train the surrogate model $\mathcal{M}^s$ on the poisoned data $\mathbf{R} = \begin{bmatrix} \mathbf{R}^r \\ \mathbf{R}^f \end{bmatrix}$ using a recommendation loss function, ensuring that the optimized parameter θ^* satisfies the constraint $\mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^s(\theta^*; \mathbf{R})) \leq \rho$. We then introduce a bounded perturbation θ^Δ to θ^*, constrained by $\|\theta^\Delta\|_p \leq \epsilon$, and maximize the attack loss $\mathcal{L}_{atk}$ over the perturbed $\theta^* + \theta^\Delta$ to seek the worst-case model. This perturbation explores local regions around θ^*, which are presumed to lie within Ω. Consequently, the perturbed model exhibits higher attack loss than θ^*, mimicking the worst-case behaviour while remaining within the plausible victim model space. By replacing the fixed surrogate model $\mathcal{M}^s$ with the worst-case model $\mathcal{M}^{worst}$, we formulate the **Sharpness-aware Poisoning**(*SharpAP*) problem as follows:

$$\min_{\mathbf{R}^f} \overbrace{\max_{||\theta^\Delta||_p \leq \epsilon} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^* + \theta^\Delta; \mathbf{R}^r))}^{\text{sharpness-aware worst-case optimization}}, \tag{8}$$
$$\underbrace{\qquad\qquad\qquad\qquad}_{\text{seek the worst-case model}}$$

Algorithm 1: Sharpness-aware poisoning attack (*SharpAP*).

Input: Real user data $\mathbf{R}^r$; learning rate for inner and outer objective: λ_1 and λ_2; max iteration for inner and outer objective: T and L.
Parameter: Parameter θ for the surrogate model $\mathcal{M}^s$
Output:Fake user data $\mathbf{R}^f$

Initialize fake user data $\mathbf{R}^f$ and the parameter θ of the surrogate model $\mathcal{M}^s$
for $l = 1$ to L **do**
  // (1) Surrogate model optimization
  for $t = 1$ to T **do**
    Optimize surrogate model parameter with SGD:
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \lambda_1 \nabla_{\theta^{(t-1)}}(\mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^s(\theta^{(t-1)}, \mathbf{R})))$
  end for
  $\theta^* \leftarrow \theta^{(T)}$
  // (2) *SharpAP*
  Seek the worst-case model according to (7)
  Solve θ^Δ according to (11)
  Compute gradients $\nabla_{\mathbf{R}^f}\mathcal{L}_{atk}$ according to (12)
  // (3) Projected gradient descent
  Update fake user data:
  $\mathbf{R}^f = Proj(\mathbf{R}^f - \lambda_2 \nabla_{\mathbf{R}^f}\mathcal{L}_{atk})$
end for
return $\mathbf{R}^f$

$$\text{s.t.} \theta^* = \arg\min_{\theta}(\mathcal{L}_{rec}(\mathbf{R}, \mathcal{M}^s(\theta; \mathbf{R}))). \tag{9}$$

Formally, *SharpAP* extends the bi-level optimization in (1) and (2) to a sharpness-aware tri-level optimization. The term in the outer optimization can be defined as the *sharpness-aware attack objective* (i.e., $\max_{||\theta^\Delta||_p \leq \epsilon} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^* + \theta^\Delta; \mathbf{R}^r))$). *SharpAP* improves existing methods by replacing their attack objective with a sharpness-aware counterpart. By iteratively optimizing the poisoning effect against the worst-case model (i.e., sharpness-aware worst-case optimization), *SharpAP* explicitly treats attack transferability as a learning objective. Consequently, it mitigates the overfitting issue and enhances the transferability. The detailed tri-level optimization process is shown in Fig. 4.

B. *The Optimization of* SharpAP

Compared to existing attack objectives, sharpness-aware attack objective introduces an additional worst-case perturbation θ^Δ to the surrogate parameter θ^*.

Therefore, we first need to solve for $\hat{\theta}^\Delta$. Specifically, we follow the approach in [7] to approximate the maximization problem via a first-order Taylor expansion, as follows:

$$\hat{\theta}^\Delta = \epsilon \cdot \text{sign}\left(\nabla_\theta \mathcal{L}_{atk}(\mathcal{M}^s(\theta^*; \mathbf{R}^r))\right) |\nabla_\theta$$
$$\mathcal{L}_{atk}(\mathcal{M}^s(\theta^*; \mathbf{R}^r))|^{q-1}$$
$$\cdot \left(\|\nabla_\theta \mathcal{L}_{atk}(\mathcal{M}^s(\theta^*; \mathbf{R}^r))\|_q^q\right)^{1/p}, \tag{10}$$

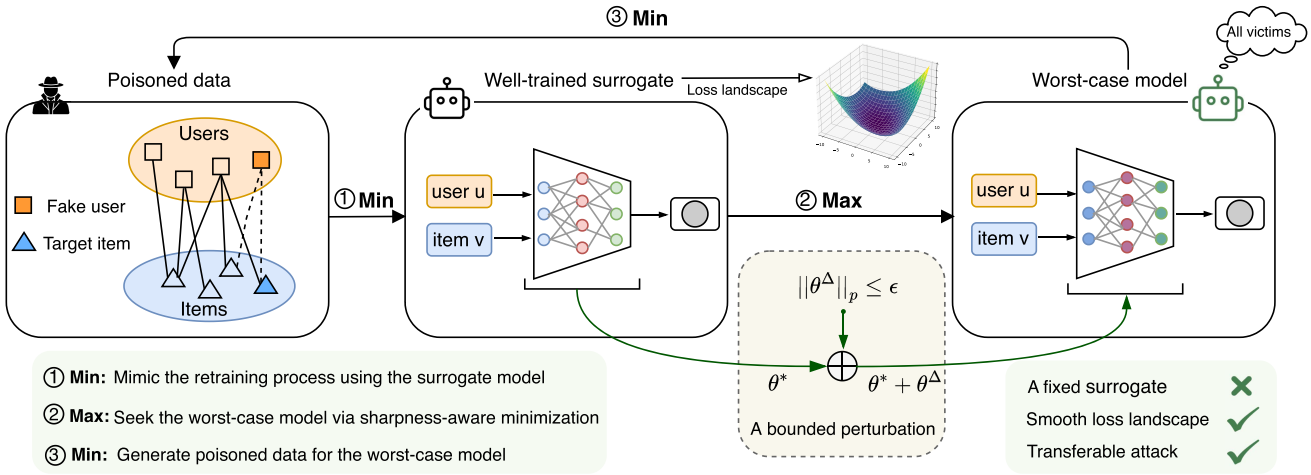


Fig. 4. The overall illustration of our method *SharpAP*. We propose a sharpness-aware tri-level optimization, which seeks the worst-case model (i.e., the victim model with the worst poisoning effect) under a bounded perturbation to generate robust poisoned data.

where $1/p + 1/q = 1$, and we set $p = 2$ as [7]. The computation of (10) can be further simplified as:

$$\hat{\theta}^{\Delta} = \epsilon \cdot \nabla_{\theta} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^*; \mathbf{R}^r)) \times \left(\|\nabla_{\theta} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^*; \mathbf{R}^r))\|_2^2\right)^{1/2}. \tag{11}$$

Then, we can have the approximation to calculate the worst attack effect of the parameter θ^* under a bounded perturbation via replacing θ^* with $\theta^* + \hat{\theta}^{\Delta}$, as follows:

$$\begin{aligned} &\nabla_{\mathbf{R}^f} \max_{||\theta^{\Delta}||_p \leq \epsilon} \mathcal{L}_{atk}(\mathcal{M}^s(\theta^* + \theta^{\Delta}; \mathbf{R}^r)) \\ &\approx \nabla_{\mathbf{R}^f} \mathcal{L}_{atk}(\mathcal{M}^s(\theta; \mathbf{R}^r))|_{\theta=\theta^*+\hat{\theta}^{\Delta}}. \end{aligned} \tag{12}$$

Since the fake data $\mathbf{R}^f$ to be generated are discrete and subject to the profile size N constraint, we introduce constrained gradient projection descent, as follows:

$$\begin{aligned} &\mathbf{R}^f = Proj(\mathbf{R}^f - \lambda_2 \nabla_{\mathbf{R}^f} \mathcal{L}_{atk}(\mathcal{M}^s(\theta; \mathbf{R}^r))|_{\theta=\theta^*+\hat{\theta}^{\Delta}}), \\ &\text{where} Proj(\hat{r}_{u^f v}) = \begin{cases} 1, & v \in N_{u^f}, \\ 0, & otherwise, \end{cases} \end{aligned} \tag{13}$$

where $\hat{r}_{u^f v}$ is the predicted score that fake user u^f gives to item v and N_{u^f} is Top-N largest item set of fake user u^f according to the score $\hat{r}_{u^f v}$. N is the maximum number of items each fake user $u^f \in U^f$ can interact with.

We present detailed *SharpAP* in Algorithm 1. Compared to existing gradient-based methods [18], [29], [47], our method only introduces the calculation of a worst-case perturbation for the surrogate model. This calculation does not require additional attack knowledge or capabilities, and the computational cost is almost negligible [7].

C. SharpAP *under Full-user and Group Attacks*

By replacing the attack loss function $\mathcal{L}_{atk}$ in (8), different attack methods targeting specific goals (e.g., full-user attacks or group attacks) can be realized [17], [18], [29], [47]. Here, we use RevAdv [29] as the backbone to implement the proposed *SharpAP* under full-user and group attacks as an example. The goal of full-user attacks is to increase the recommendation probability of target items $v^t \in V^t = \{v_1^t, v_2^t, ..., v_{|V^t|}^t\}$ for all real users in U^r. The attack loss function $\mathcal{L}_{atk}$ can be formulated as:

$$\mathcal{L}_{atk}^{full}(\hat{\mathbf{R}}^r) = -\sum_{v^t \in V^t} \sum_{u^r \in U^r} \log\left(\frac{\exp(\hat{r}_{u^r v^t})}{\sum_{v \in V} \exp(\hat{r}_{u^r v})}\right). \tag{14}$$

The goal of group attacks is to target a specific group while minimizing the impact on other groups. We categorize real users in U^r into two groups, U_0 and U_1, based on their binary attribute values. U_0 represents the set of users with an attribute value of 0, while U_1 represents the set of users with an attribute value of 1. Here, we attack the group U_0 as an example.

$$\mathcal{L}_{atk}^{group}(\hat{\mathbf{R}}^r) = \sum_{v^t \in V^t} \left(\frac{1}{|U_1|} \sum_{u \in U_1} \mathbb{I}_{v^t \notin \Gamma_u} \hat{r}_{uv^t} - \frac{1}{|U_0|} \sum_{u \in U_0} \hat{r}_{uv^t}\right), \tag{15}$$

where Γ_u is Top-K ranked items for user u, $\mathbb{I}$ is an indicator function, if target item v^t is not in the set Γ_u, then $\mathbb{I} = 1$, otherwise $\mathbb{I} = 0$. We can replace $\mathcal{L}_{atk}$ in (8) with (14) and (15) to implement sharpness-aware full-user attacks and group attacks, respectively. For $\mathcal{L}_{rec}$, we employ a widely used weighted mean squared error loss function [18], [29] associated with implicit feedback matrix factorization. The specific formulation is:

$$\mathcal{L}_{rec} = \sum_{u,v} c_{uv}(r_{uv} - \mathbf{u}_u^T \mathbf{v}_v)^2 + \lambda \left(\sum_u \|\mathbf{u}_u\|^2 + \sum_v \|\mathbf{v}_v\|^2\right), \tag{16}$$

where c_{uv} is the instance weight to differentiate observed and missing interactions. $\mathbf{u}_u$ and $\mathbf{v}_v$ are the user and item latent vectors, respectively. λ is the regularization parameter. The hyperparameter settings in the $\mathcal{L}_{rec}$ loss are kept consistent with those in [29] and [18] for fair comparison.

D. Theoretical Analysis

Representative RS models [11], [14], [26], [46] typically rely on learnable user and item embeddings to encode latent preferences. Despite differences in loss functions or aggregation mechanisms, these models map the same user–item interaction data into a common embedding space. This shared embedding-based representation implies that retraining different RS models on the same data leads to parameter solutions that may not be arbitrarily distant, but instead tend to reside in a relatively constrained region of the embedding space. Therefore, it is reasonable to assume that the optimal parameters (embeddings) of a victim model θ^v lie within a bounded neighborhood of the surrogate model parameters (embeddings) θ^*: $||\theta^v - \theta^*||_2 \leq \epsilon$.

Proposition 1 (Transferability Bound via Sharpness-Aware Minimization): Let θ^* and θ^v denote the parameters of the surrogate model and the unknown victim model, respectively. We assume the victim model resides within an ϵ-neighborhood of the surrogate model, i.e., $\|\theta^v - \theta^*\|_2 \leq \epsilon$ with $\|\zeta\| \leq \epsilon$. Furthermore, assume the attack loss function $\mathcal{L}_{atk}(\theta)$ is L-Lipschitz smooth, implying $\|\nabla\mathcal{L}_{atk}(\theta_1) - \nabla\mathcal{L}_{atk}(\theta_2)\|_2 \leq L\|\theta_1 - \theta_2\|_2$. Under these conditions, the attack loss on the victim model is upper-bounded by the surrogate loss plus a sharpness-related term:

$$\mathcal{L}_{atk}(\theta^v) \leq \underbrace{\mathcal{L}_{atk}(\theta^*)}_{\text{Surrogate Performance}} + \underbrace{\epsilon\|\nabla\mathcal{L}_{atk}(\theta^*)\|_2}_{\text{Local Sharpness}} + \frac{L\epsilon^2}{2}. \quad (17)$$

Furthermore, the maximization step in *SharpAP* serves as a proxy for minimizing this upper bound.

Proof: Since $\mathcal{L}_{atk}$ is L-smooth, satisfying the quadratic upper bound property, we perform a Taylor expansion around θ^*:

$$\mathcal{L}_{atk}(\theta^v) \leq \mathcal{L}_{atk}(\theta^*) + \nabla\mathcal{L}_{atk}(\theta^*)^\top(\theta^v - \theta^*) + \frac{L}{2}\|\theta^v - \theta^*\|_2^2. \quad (18)$$

Let $\zeta = \theta^v - \theta^*$. Since the victim model lies within the ϵ-neighborhood, we have $\|\zeta\|_2 \leq \epsilon$. By applying the Cauchy-Schwarz inequality to the first-order term, we obtain:

$$\nabla\mathcal{L}_{atk}(\theta^*)^\top\zeta \leq \|\nabla\mathcal{L}_{atk}(\theta^*)\|_2\|\zeta\|_2 \leq \epsilon\|\nabla\mathcal{L}_{atk}(\theta^*)\|_2. \quad (19)$$

Substituting this back into the quadratic upper bound yields:

$$\mathcal{L}_{atk}(\theta^v) \leq \mathcal{L}_{atk}(\theta^*) + \epsilon\|\nabla\mathcal{L}_{atk}(\theta^*)\|_2 + \frac{L\epsilon^2}{2}. \quad (20)$$

The *SharpAP* objective explicitly maximizes the loss within the perturbation ball:

$$\max_{\|\theta^\Delta\|\leq\epsilon} \mathcal{L}_{atk}(\theta^* + \theta^\Delta) \approx \mathcal{L}_{atk}(\theta^*) + \max_{\|\theta^\Delta\|\leq\epsilon} \nabla\mathcal{L}_{atk}(\theta^*)^\top\theta^\Delta = \mathcal{L}_{atk}(\theta^*) + \epsilon\|\nabla\mathcal{L}_{atk}(\theta^*)\|_2. \quad (21)$$

Thus, by minimizing the worst-case loss, we are effectively minimizing the upper bound of $\mathcal{L}_{atk}(\theta^v)$. □

Traditional attacks minimize only $\mathcal{L}_{atk}(\theta^*)$. If the landscape is sharp (large second term), $\mathcal{L}_{atk}(\theta^v)$ can still be high. *SharpAP* minimizes both surrogate performance and local sharpness, effectively flattening the loss landscape, thereby reducing the transferability gap.

E. Time Complexity

As shown in Algorithm 1, the computational cost per outer iteration consists of three components: inner min, inner max, and outer min. For inner min, it involves updating surrogate parameters via SGD. The cost is approximately $O(T \cdot |U| \cdot \bar{n} \cdot d)$, where $\bar{n}$ is the average number of interacted items per user and d denotes the embedding dimension. For inner max, unique to *SharpAP*, requires computing gradients to find $\hat{\theta}^\Delta$. The cost is $O(|U| \cdot \bar{n} \cdot d)$, as the cost of calculating $\nabla_\theta\mathcal{L}_{atk}$ is similar to a standard backward pass. For outer min, it requires one forward and backward pass, costing $O(|U| \cdot \bar{n} \cdot d)$. Summing these components, the approximate computational complexity for the *SharpAP* attack is: $O(L \cdot (T+2) \cdot |U| \cdot \bar{n} \cdot d)$. The computational overhead introduced by SharpAP is limited to one additional backward pass per outer iteration compared to the bi-level optimization method. Given that T dominates the computational cost, this small addition (i.e., $O(L \cdot |U| \cdot \bar{n} \cdot d)$) confirms that *SharpAP* incurs marginal time costs while significantly boosting transferability.

IV. Experiments

In this section, we present extensive experiments on three real-world datasets to evaluate the effectiveness of our proposed *SharpAP*. We begin by describing the experimental settings, followed by a comparison of the overall performance against state-of-the-art baselines. Finally, we provide a detailed analysis of *SharpAP*.

A. Experimental Settings

1) Datasets: We select three real-world datasets for the experiments, i.e., MovieLens-1M [9], Amazon-book [23], and Gowalla [4]. To convert MovieLens-1M into an implicit feedback dataset, we follow prior works [30], [53], treating interactions with a rating of 5 as positive feedback, while considering all other interactions as negative feedback. For Gowalla, we adhere to the data processing procedure outlined in [28], [29]. Specifically, we preprocess the raw data by removing cold-start users and items with fewer than 15 interactions. For Amazon-book, we follow [11] and use the 10-core setting to ensure that each user and item have at least 10 interactions. We adopt a standard training/validation/test split of 7:1:2 across all datasets. Table II summarizes the key statistics of the three datasets.

2) Evaluation Metrics: To quantitatively evaluate the effectiveness of full-user attacks, we follow [18], [47] and uniformly sample five items to form the target item set V^t. The attack's success is assessed using the hit ratio (H@K) on real users U^r. A hit occurs if at least one target item appears in the Top-K recommendations for any user $u \in U^r$. Additionally, we employ Normalized Discounted Cumulative Gain (NDCG) to assess the ranking quality of target items within the recommendation list. Specifically, we assign a relevance score of 1 to target items and

TABLE I
MATHEMATICAL NOTATIONS.

Notation	Description
u^r, u^f v, v^t	Real user and fake user. Item and target item.
U^r, U^f, U V, V^t	Real user set, fake user set and all user set. Item set and target item set.
$\mathbf{R}^r$, $\mathbf{R}^f$, $\mathbf{R}$ $\hat{\mathbf{R}}$	Real user data, fake user data, and all user data. Predicted scores.
$\mathcal{M}^s$, $\mathcal{M}^v$ $\mathcal{M}^{worst}$	Surrogate model and victim model. Worst-case model.
Ω	Victim model space.
θ^* θ^Δ	Optimal parameter of the surrogate model. A bounded perturbation to θ^*.
$\mathcal{L}_{rec}$ $\mathcal{L}_{atk}$	Recommendation loss. Attack objective loss.

TABLE II
STATISTICS OF THE THREE DATASETS. "AVG." REPRESENTS THE AVERAGE NUMBER OF ITEMS INTERACTED WITH BY EACH USER.

Datasets	Users	Items	Ratings	Avg.	Density
MovieLens-1M	6,014	3,232	226,310	38	1.17%
Gowalla	13,149	14,007	433,356	33	0.24%
Amazon-book	52,643	91,599	2,984,108	57	0.06%

0 to all other items for every real user. Both metrics are computed for each real test user individually, and then averaged over all real test users.

In group attacks, items are divided into two categories according to their popularity within the attack group (i.e., U_0) [31]. Specifically, we select the top 20% items as popular and the bottom 80% as unpopular. Then, we randomly select five popular items and five unpopular items as the target item set V^t from the popular category and the unpopular category, respectively. Our goal is to attack the U_0 group while minimizing the impact on the U_1 group. To this end, we measure the difference in hit rates for the target items between the two groups, as follows:

$$D@K = H_{U_0}@K - H_{U_1}@K, \tag{22}$$

where $H_{U_0}@K$ and $H_{U_1}@K$ represent the hit rates of target items for two groups. A higher $D@K$ value indicates stronger attack performance.

3) Baseline Attack Methods: To evaluate the effectiveness of *SharpAP*, we select several competing baselines, including heuristic-based, neural network-driven, and gradient-based poisoning attacks. The details are outlined as follows:

- *Clean:* Recommendation models are trained on an unattacked dataset.
- *Random Attack[16]:* In this attack, fake users interact with target items as well as a set of randomly selected items.
- *Popular Attack:* According to previous work, [25], [47], each fake user selects 10% of the most popular items and 90% of randomly chosen items as the interacted items.
- *CoVis Attack [44]:* This baseline attack is tailored for association-rule-based recommendation models. In this method, the attacker identifies items for generating fake interactions and injects artificial co-visitations by solving a standard linear programming problem.
- *PGA Attack [17]:* This method targets matrix factorization-based recommendation models. It defines an attack objective and utilizes projected gradient ascent to update the poisoned user's ratings in order to optimize this objective.
- *AUSH Attack [21]:* AUSH leverages Generative Adversarial Networks to tailor attacks on recommender systems based on budget and complex goals, such as targeting specific user groups.
- *UBA Attack [31]:* The attack emphasizes the importance of targeting specific users and frames the issue of varying attack difficulty across users using causal language, ultimately calculating the optimal allocation of fake user budgets.
- *RevAdv Attack [29]:* RevAdv is a classic gradient-based attack method, which argues that previous methods calculate the gradient inaccurately. It further proposes a more accurate computation method, achieving state-of-the-art performance.
- *RAPU Attack [47]:* Compared to RevAdv, RAPU focuses on situations with incomplete training data and introduces a different attack objective function.
- *DADA Attack [18]:* DADA introduces a difficulty- and diversity-aware attack objective that ensures easy-to-manipulate users from diverse groups receive more attention, enhancing the overall attack effectiveness.
- *CLeaR Attack [35]:* CLeaR employs a dual-objective strategy that promotes a smoother spectral value distribution to broaden user reachability while simultaneously optimizing a rank promotion objective to maximize the exposure of target items.
- *DDSP Attack [51]:* DDSP employs a dual-promotion objective to simultaneously promote both target items and user-preferred items.

4) Victim Models: In this section, we carefully select representative RS models as victim models to evaluate the effectiveness of the attack.

- *WRMF [14]:* It is a foundational and representative factorization-based model for RS using implicit feedback.
- *BPR [26]:* It is a classic collaborative filtering method that designs a pairwise ranking loss function, which is widely applied in recommendations based on implicit feedback.
- *LightGCN [11]:* It is a state-of-the-art GCN-based method that eliminates feature transformation and nonlinear activation functions in the GCN aggregator.
- *SGL [41]:* Compared to LightGCN, SGL further enhances recommendation performance by utilizing self-supervised graph learning.
- *SimGCL [46]:* SimGCL proposes simple graph contrastive learning and noise-based augmentation for graph recommendation.

5) Implement Details: For all attacks, without special explanation, we adopt the following attack settings. The percentage of fake users is fixed to 1% (i.e., $\delta = 1\%$). The maximum

TABLE III
A COMPARISON OF THE PERFORMANCE OF VARIOUS ATTACK METHODS ON FIVE REPRESENTATIVE VICTIM RS, EVALUATED ON **MOVIELENS-1M** DATASET. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	WRMF		BPR		LightGCN		SGL		SimGCL	
	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20
Clean	0.0614±5e-3	0.0063±2e-4	0.0294±3e-3	0.0031±1e-4	0.0417±7e-3	0.0044±3e-4	0.0171±2e-3	0.0016±1e-4	0.0152±4e-4	0.0014±1e-4
Random	0.0620±7e-3	0.0062±3e-4	0.0296±5e-3	0.0032±3e-4	0.0416±4e-3	0.0045±5e-4	0.0175±7e-3	0.0017±2e-4	0.0159±5e-4	0.0014±2e-4
Popular	0.0630±4e-3	0.0063±2e-4	0.0298±4e-3	0.0033±2e-4	0.0418±2e-3	0.0046±2e-4	0.0179±5e-3	0.0018±2e-4	0.0167±6e-4	0.0015±1e-4
CoVis	0.0641±3e-3	0.0064±2e-4	0.0305±2e-3	0.0036±2e-4	0.0423±4e-3	0.0051±3e-4	0.0182±2e-3	0.0021±1e-4	0.0164±3e-4	0.0015±2e-4
PGA	0.0672±2e-3	0.0069±3e-4	0.0298±1e-3	0.0036±3e-4	0.0434±5e-3	0.0050±3e-4	0.0191±2e-3	0.0020±2e-4	0.0180±5e-4	0.0017±3e-4
AUSH	0.0731±2e-3	0.0077±4e-4	0.0315±2e-3	0.0040±2e-4	0.0485±2e-3	0.0062±2e-4	0.0181±3e-3	0.0018±3e-4	0.0191±4e-4	0.0018±1e-4
RevAdv	0.0743±4e-3	0.0083±2e-4	0.0333±3e-3	0.0043±2e-4	0.0491±3e-3	0.0061±4e-4	0.0196±1e-3	0.0021±2e-4	0.0203±5e-4	0.0019±2e-4
+SharpAP	**0.0783±2e-3**	**0.0087±2e-4**	**0.0434±2e-3**	**0.0051±2e-4**	**0.0642±2e-3**	**0.0074±4e-4**	**0.0229±2e-3**	**0.0025±1e-4**	**0.0234±5e-4**	**0.0022±1e-4**
RAPU	0.0761±3e-3	0.0084±3e-4	0.0307±5e-3	0.0040±1e-4	0.0503±1e-3	0.0070±1e-4	0.0208±4e-3	0.0022±2e-4	0.0218±3e-4	0.0020±2e-4
+SharpAP	**0.0808±1e-3**	**0.0090±2e-4**	**0.0421±2e-3**	**0.0046±2e-4**	**0.0621±2e-3**	**0.0076±2e-4**	**0.0225±4e-3**	**0.0027±1e-4**	**0.0239±7e-4**	**0.0022±2e-4**
DADA	0.0759±4e-3	0.0085±2e-4	0.0344±3e-3	0.0042±2e-4	0.0539±4e-3	0.0068±2e-4	0.0217±3e-3	0.0024±2e-4	0.0205±1e-3	0.0018±1e-4
+SharpAP	**0.0798±3e-3**	**0.0091±2e-4**	**0.0442±1e-3**	**0.0052±1e-4**	**0.0604±2e-3**	**0.0071±2e-4**	**0.0240±3e-3**	**0.0028±2e-4**	**0.0240±1e-3**	**0.0023±2e-4**
CLeaR	0.0741±2e-3	0.0083±2e-4	0.0327±1e-3	0.0042±2e-4	0.0484±5e-3	0.0061±3e-4	0.0198±1e-3	0.0021±1e-4	0.0217±8e-4	0.0020±2e-4
+SharpAP	**0.0780±1e-3**	**0.0086±2e-4**	**0.0439±2e-3**	**0.0052±1e-4**	**0.0650±3e-3**	**0.0075±1e-4**	**0.0234±2e-3**	**0.0027±2e-4**	**0.0232±5e-4**	**0.0022±1e-4**
DDSP	0.0771±4e-3	0.0085±3e-4	0.0339±5e-3	0.0043±2e-4	0.0512±4e-3	0.0070±1e-4	0.0216±4e-3	0.0025±1e-4	0.0221±8e-4	0.0021±1e-4
+SharpAP	**0.0803±4e-3**	**0.0090±2e-4**	**0.0448±4e-3**	**0.0054±3e-4**	**0.0663±6e-3**	**0.0079±2e-4**	**0.0242±3e-3**	**0.0028±2e-4**	**0.0244±7e-4**	**0.0023±2e-4**

TABLE IV
A COMPARISON OF THE PERFORMANCE OF VARIOUS ATTACK METHODS ON FIVE REPRESENTATIVE VICTIM RS, EVALUATED ON **GOWALLA** DATASET. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	WRMF		BPR		LightGCN		SGL		SimGCL	
	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20
Clean	0.0119±5e-4	0.0010±2e-4	0.0145±6e-4	0.0016±1e-4	0.0099±6e-4	0.0019±3e-4	0.0097±5e-4	0.0011±1e-4	0.0110±4e-4	0.0009±1e-4
Random	0.0117±5e-4	0.0009±1e-4	0.0147±4e-4	0.0015±2e-4	0.0115±9e-4	0.0018±2e-4	0.0099±3e-4	0.0012±2e-4	0.0118±5e-4	0.0009±2e-4
Popular	0.0123±3e-4	0.0010±2e-4	0.0149±3e-4	0.0015±2e-4	0.0132±4e-4	0.0019±2e-4	0.0108±5e-4	0.0014±1e-4	0.0123±3e-4	0.0010±2e-4
CoVis	0.0125±4e-4	0.0010±1e-4	0.0152±2e-4	0.0017±1e-4	0.0131±4e-4	0.0018±2e-4	0.0112±4e-4	0.0015±2e-4	0.0125±4e-4	0.0010±1e-4
PGA	0.0128±2e-4	0.0011±2e-4	0.0156±5e-4	0.0018±2e-4	0.0157±6e-4	0.0021±3e-4	0.0124±3e-4	0.0016±3e-4	0.0132±5e-4	0.0011±2e-4
AUSH	0.0131±3e-4	0.0012±3e-4	0.0151±6e-4	0.0017±2e-4	0.0162±5e-4	0.0022±1e-4	0.0107±2e-4	0.0014±2e-4	0.0139±3e-4	0.0012±2e-4
RevAdv	0.0135±5e-4	0.0013±2e-4	0.0160±5e-4	0.0019±4e-4	0.0165±4e-4	0.0023±5e-4	0.0161±2e-4	0.0019±3e-4	0.0144±5e-4	0.0012±3e-4
+SharpAP	**0.0142±4e-4**	**0.0018±2e-4**	**0.0189±3e-4**	**0.0026±2e-4**	**0.0213±1e-4**	**0.0031±1e-4**	**0.0193±3e-4**	**0.0025±2e-4**	**0.0162±3e-4**	**0.0014±2e-4**
RAPU	0.0136±3e-4	0.0014±3e-4	0.0158±5e-4	0.0018±3e-4	0.0164±5e-4	0.0024±2e-4	0.0165±4e-4	0.0019±2e-4	0.0137±4e-4	0.0013±1e-4
+SharpAP	**0.0139±2e-4**	**0.0019±1e-4**	**0.0191±4e-4**	**0.0028±4e-4**	**0.0210±5e-4**	**0.0029±2e-4**	**0.0187±2e-4**	**0.0021±1e-4**	**0.0153±4e-4**	**0.0014±1e-4**
DADA	0.0141±5e-4	0.0022±2e-4	0.0173±3e-4	0.0021±3e-4	0.0198±4e-4	0.0027±3e-4	0.0170±5e-4	0.0021±3e-4	0.0140±3e-4	0.0012±2e-4
+SharpAP	**0.0163±4e-4**	**0.0024±2e-4**	**0.0199±5e-4**	**0.0030±2e-4**	**0.0214±2e-4**	**0.0033±4e-4**	**0.0185±4e-4**	**0.0024±2e-4**	**0.0162±2e-4**	**0.0014±1e-4**
CLeaR	0.0134±5e-4	0.0013±3e-4	0.0162±7e-4	0.0020±3e-4	0.0174±6e-4	0.0025±2e-4	0.0172±6e-4	0.0020±3e-4	0.0138±4e-4	0.0013±2e-4
+SharpAP	**0.0140±5e-4**	**0.0016±2e-4**	**0.0189±4e-4**	**0.0027±2e-4**	**0.0195±5e-4**	**0.0029±1e-4**	**0.0190±5e-4**	**0.0025±1e-4**	**0.0157±2e-4**	**0.0014±2e-4**
DDSP	0.0138±4e-4	0.0017±1e-4	0.0168±8e-4	0.0021±2e-4	0.0183±6e-4	0.0026±2e-4	0.0176±8e-4	0.0021±4e-4	0.0142±5e-4	0.0012±1e-4
+SharpAP	**0.0143±2e-4**	**0.0018±1e-4**	**0.0182±5e-4**	**0.0024±2e-4**	**0.0197±5e-4**	**0.0029±2e-4**	**0.0194±4e-4**	**0.0024±2e-4**	**0.0160±3e-4**	**0.0013±1e-4**

number of items each fake user can interact with, denoted as N, is set to the average number of items interacted with by real users in the dataset, as shown in Table. II. Following [6], [18], [29], we employ the representative factorization-based model WRMF [14] as the surrogate model and set the learning rate λ_2 in Alg.1 to 1. We also use a more complex model LightGCN [11] as the surrogate in Table VI. For our method *SharpAP*, we search the perturbation radius ϵ in (8) within $\{0.005, 0.02, 0.05, 0.1, 0.2\}$. All experiments are conducted on an NVIDIA A40 GPU with Pytorch-2.1.2. The reported results are averaged over ten runs with different random strategies (target item sampling, fake user initialization, and optimizer seeds).

B. Overall Performance

1) Full-user Attack Performance: We report the full-user attack performance of all methods under different Top-K settings from Tables III–VII, and have the following observations:

- All attack methods increase the exposure probability of target items. However, heuristic-based methods (e.g., Random, Popular, and CoVis) do not achieve impressive performance because they do not directly optimize the attack objective. Instead, they manually select fake user interaction records to enhance the co-occurrence between target items and other items.
- Although some gradient-based methods achieve good attack performance by directly optimizing the attack objective, they exhibit overfitting. Our *SharpAP* mitigates the overfitting issue and improves performance even on models beyond the surrogate model. Note that when using RevAdv as the base model, SharpAP boosts HR@20 on BPR and WRMF by 48% and 27%, respectively. We hypothesize that the pairwise objective intrinsically creates a loss landscape with higher curvature (sharper local minima) and greater sensitivity to input perturbations compared to WRMF.
- As shown in Table VI, when using LightGCN as the surrogate without SharpAP, the attack performs exceptionally

TABLE V
A COMPARISON OF THE PERFORMANCE OF VARIOUS ATTACK METHODS ON FIVE REPRESENTATIVE VICTIM RS, EVALUATED ON **AMAZON-BOOK** DATASET. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	WRMF		BPR		LightGCN		SGL		SimGCL	
	H@20(%)	N@20(%)	H@20(%)	N@20(%)	H@20(%)	N@20(%)	H@20(%)	N@20(%)	H@20(%)	N@20(%)
Clean	0.1615±2e-3	0.0134±3e-4	0.1164±4e-3	0.0101±9e-4	0.1577±3e-3	0.0143±9e-4	0.2090±1e-3	0.0198±1e-3	0.1623±4e-3	0.0158±2e-4
Random	0.1632±5e-3	0.0138±1e-3	0.1166±3e-3	0.0100±1e-3	0.1592±3e-3	0.0148±2e-4	0.2104±4e-3	0.0199±6e-4	0.1744±5e-3	0.0165±4e-4
Popular	0.1750±3e-3	0.0142±5e-4	0.1190±1e-3	0.0108±5e-4	0.1600±4e-3	0.0151±5e-4	0.2110±2e-3	0.0207±2e-4	0.1982±3e-3	0.0183±4e-4
CoVis	0.1741±4e-3	0.0140±7e-4	0.1183±5e-3	0.0105±3e-4	0.1598±5e-3	0.0147±4e-4	0.2106±6e-3	0.0201±3e-4	0.1710±4e-3	0.0162±5e-4
PGA	0.1793±2e-3	0.0151±9e-4	0.1222±5e-3	0.0113±6e-4	0.1663±7e-3	0.0156±8e-4	0.2199±2e-3	0.0210±5e-4	0.2174±4e-3	0.0209±2e-4
AUSH	0.1752±5e-3	0.0145±6e-4	0.1201±4e-3	0.0110±3e-4	0.1667±6e-3	0.0159±1e-3	0.2137±3e-3	0.0208±4e-4	0.1962±2e-3	0.0191±3e-4
RevAdv	0.1820±2e-3	0.0150±5e-4	0.1254±3e-3	0.0114±7e-4	0.1702±2e-3	0.0160±7e-4	0.2198±1e-3	0.0211±2e-4	0.2144±4e-3	0.0205±4e-4
+SharpAP	**0.1932±1e-3**	**0.0168±2e-4**	**0.1402±2e-3**	**0.0126±5e-4**	**0.1882±1e-3**	**0.0171±5e-4**	**0.2403±2e-3**	**0.0227±4e-4**	**0.2352±4e-3**	**0.0228±2e-4**
RAPU	0.1843±3e-3	0.0156±6e-4	0.1293±2e-3	0.0119±6e-4	0.1780±3e-3	0.0163±6e-4	0.2204±3e-3	0.0219±8e-4	0.2013±5e-3	0.0194±3e-4
+SharpAP	**0.1944±2e-3**	**0.0171±4e-4**	**0.1440±1e-3**	**0.0129±5e-4**	**0.1914±2e-3**	**0.0178±6e-4**	**0.2366±1e-3**	**0.0225±5e-4**	**0.2454±2e-3**	**0.0233±2e-4**
DADA	0.1905±5e-3	0.0160±6e-4	0.1337±7e-3	0.0123±1e-3	0.1808±4e-3	0.0166±4e-4	0.2189±2e-3	0.0210±4e-4	0.1970±5e-3	0.0182±4e-4
+SharpAP	**0.2029±4e-3**	**0.0178±5e-4**	**0.1497±2e-3**	**0.0134±7e-4**	**0.2019±3e-3**	**0.0181±8e-4**	**0.2378±1e-3**	**0.0228±3e-4**	**0.2268±3e-3**	**0.0215±1e-4**
CLeaR	0.1900±6e-3	0.0158±7e-4	0.1342±6e-3	0.0128±8e-4	0.1812±5e-3	0.0170±7e-4	0.2192±3e-3	0.0218±9e-4	0.2107±2e-3	0.0204±5e-4
+SharpAP	**0.2042±4e-3**	**0.0180±4e-4**	**0.1503±3e-3**	**0.0137±6e-4**	**0.2024±4e-3**	**0.0184±5e-4**	**0.2400±2e-3**	**0.0229±5e-4**	**0.2462±3e-3**	**0.0227±4e-4**
DDSP	0.1913±2e-3	0.0159±5e-4	0.1350±5e-3	0.0129±5e-4	0.1809±6e-3	0.0164±2e-3	0.2203±5e-3	0.0219±1e-3	0.2245±4e-3	0.0212±4e-4
+SharpAP	**0.2107±1e-3**	**0.0189±2e-4**	**0.1594±4e-3**	**0.0141±3e-4**	**0.2073±5e-3**	**0.0190±2e-3**	**0.2481±4e-3**	**0.0237±8e-4**	**0.2530±2e-3**	**0.0241±3e-4**

TABLE VI
A COMPARISON OF THE PERFORMANCE OF VARIOUS ATTACK METHODS ON FIVE REPRESENTATIVE VICTIM RS, EVALUATED ON **MOVIELENS-1M** DATASET. **LIGHTGCN** IS USED AS THE SURROGATE MODEL. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	WRMF		BPR		LightGCN		SGL		SimGCL	
	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20
Clean	0.0614±3e-3	0.0063±3e-4	0.0294±1e-3	0.0031±4e-4	0.0417±5e-3	0.0044±2e-4	0.0171±2e-3	0.0016±3e-4	0.0242±3e-3	0.0026±2e-4
RevAdv	0.0703±2e-3	0.0076±5e-4	0.0311±3e-3	0.0040±5e-4	0.0532±1e-3	0.0068±3e-4	0.0182±2e-3	0.0020±4e-4	0.0290±2e-3	0.0031±2e-4
+SharpAP	**0.0783±1e-3**	**0.0087±2e-4**	**0.0434±4e-3**	**0.0051±1e-4**	**0.0608±2e-3**	**0.0071±2e-4**	**0.0214±3e-3**	**0.0024±2e-4**	**0.0317±6e-4**	**0.0033±1e-4**
RAPU	0.0721±2e-3	0.0077±4e-4	0.0301±4e-3	0.0035±3e-4	0.0520±2e-3	0.0066±2e-4	0.0189±3e-3	0.0023±3e-4	0.0283±1e-3	0.0029±1e-4
+SharpAP	**0.0778±1e-3**	**0.0086±1e-4**	**0.0408±3e-3**	**0.0044±2e-4**	**0.0593±2e-3**	**0.0072±1e-4**	**0.0230±3e-3**	**0.0027±2e-4**	**0.0309±2e-3**	**0.0032±1e-4**
DADA	0.0733±2e-3	0.0080±3e-4	0.0336±2e-3	0.0041±1e-4	0.0544±3e-3	0.0069±3e-4	0.0200±2e-3	0.0022±1e-4	0.0296±2e-3	0.0031±1e-4
+SharpAP	**0.0764±2e-3**	**0.0083±2e-4**	**0.0401±1e-3**	**0.0048±1e-4**	**0.0582±1e-3**	**0.0070±2e-4**	**0.0252±2e-3**	**0.0026±2e-4**	**0.0322±8e-4**	**0.0034±2e-4**
CLeaR	0.0727±2e-3	0.0079±3e-4	0.0320±1e-3	0.0041±1e-4	0.0493±3e-3	0.0064±3e-4	0.0183±2e-3	0.0020±2e-4	0.0277±2e-3	0.0029±1e-4
+SharpAP	**0.0762±1e-3**	**0.0084±2e-4**	**0.0427±2e-3**	**0.0050±2e-4**	**0.0611±2e-3**	**0.0072±2e-4**	**0.0224±2e-3**	**0.0025±3e-4**	**0.0308±1e-3**	**0.0032±2e-4**
DDSP	0.0742±4e-3	0.0081±2e-4	0.0330±3e-3	0.0041±3e-4	0.0538±3e-3	0.0069±2e-4	0.0214±3e-3	0.0025±3e-4	0.0291±1e-3	0.0031±2e-4
+SharpAP	**0.0788±2e-3**	**0.0087±1e-4**	**0.0425±2e-3**	**0.0051±2e-4**	**0.0624±3e-3**	**0.0075±1e-4**	**0.0235±2e-3**	**0.0026±2e-4**	**0.0320±2e-3**	**0.0034±1e-4**

TABLE VII
A COMPARISON OF THE PERFORMANCE OF FULL-USER METHODS ON FOUR REPRESENTATIVE VICTIM RS, EVALUATED ON **MOVIELENS-1M** DATASET. THE TARGET ITEMS ARE SELECTED FROM THE POPULAR AND UNPOPULAR GROUPS, RESPECTIVELY. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	Popular item								Unpopular item							
	WRMF		BPR		LightGCN		SGL		WRMF		BPR		LightGCN		SGL	
	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20	H@20	N@20
Clean	0.0871	0.0099	0.0467	0.0046	0.0813	0.0086	0.0742	0.0078	0.0026	0.0002	0.0029	0.0003	0.0046	0.0004	0.0035	0.0003
RevAdv	0.1107	0.0108	0.0551	0.0058	0.1027	0.0099	0.1035	0.0117	0.0049	0.0006	0.0041	0.0006	0.0078	0.0007	0.0067	0.0005
+SharpAP	**0.1224**	**0.0114**	**0.0762**	**0.0079**	**0.1326**	**0.0141**	**0.1284**	**0.0130**	**0.0053**	**0.0007**	**0.0048**	**0.0007**	**0.0088**	**0.0009**	**0.0079**	**0.0006**
RAPU	0.1202	0.0110	0.0594	0.0062	0.1164	0.0128	0.1090	0.0121	0.0045	0.0004	0.0040	0.0005	0.0073	0.0008	0.0063	0.0005
+SharpAP	**0.1298**	**0.0133**	**0.0787**	**0.0081**	**0.1383**	**0.0150**	**0.1311**	**0.0139**	**0.0051**	**0.0005**	**0.0052**	**0.0006**	**0.0086**	**0.0009**	**0.0081**	**0.0007**
DADA	0.1310	0.0128	0.0604	0.0063	0.1105	0.0119	0.1106	0.0124	0.0050	0.0005	0.0047	0.0004	0.0074	0.0007	0.0069	0.0006
+SharpAP	**0.1423**	**0.0151**	**0.0831**	**0.0084**	**0.1351**	**0.0142**	**0.1287**	**0.0132**	**0.0062**	**0.0007**	**0.0058**	**0.0005**	**0.0084**	**0.0008**	**0.0078**	**0.0008**
CLeaR	0.1302	0.0124	0.0611	0.0060	0.1047	0.0118	0.1124	0.0128	0.0048	0.0004	0.0048	0.0004	0.0076	0.0006	0.0067	0.0007
+SharpAP	**0.1399**	**0.0147**	**0.0752**	**0.0078**	**0.1266**	**0.0134**	**0.1306**	**0.0138**	**0.0053**	**0.0005**	**0.0055**	**0.0005**	**0.0088**	**0.0007**	**0.0076**	**0.0008**
DDSP	0.1374	0.0138	0.0632	0.0065	0.1180	0.0123	0.1202	0.0130	0.0050	0.0005	0.0051	0.0005	0.0072	0.0007	0.0080	0.0007
+SharpAP	**0.1433**	**0.0159**	**0.0840**	**0.0085**	**0.1372**	**0.0150**	**0.1327**	**0.0142**	**0.0058**	**0.0006**	**0.0062**	**0.0006**	**0.0078**	**0.0008**	**0.0085**	**0.0008**

well against a LightGCN victim, which is intuitive. However, transferability to other victims (e.g., WRMF, BPR) degrades compared to using the WRMF surrogate. We attribute this to LightGCN's complex graph aggregation mechanism, which causes the generated poisoned data to overfit specific high-order graph artifacts.

- Table VII presents a comparison of attack performance when the target items are selected from popular (Top 20%) and unpopular (Bottom 80%) items, respectively. The H@20/N@20 for unpopular items is consistently lower. Attacking unpopular items is harder than attacking popular items across all methods. This is because unpopular items have sparse interaction data, resulting in less

TABLE VIII
A COMPARISON OF THE PERFORMANCE OF GROUP ATTACK METHODS ON THREE REPRESENTATIVE VICTIM RS, EVALUATED ON **MOVIELENS-1M** DATASET. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

Attacker	Popular item								Unpopular item							
	WRMF		BPR		LightGCN		SGL		WRMF		BPR		LightGCN		SGL	
	D@10	D@20	D@10	D@20	D@10	D@20	D@10	D@20	D@10	D@20	D@10	D@20	D@10	D@20	D@10	D@20
Clean	0.0269	0.0322	0.0112	0.0177	0.0351	0.0578	0.0313	0.0495	0.0002	-0.0046	-0.0009	-0.0028	0.0000	0.0009	0.0001	0.0005
AUSH	0.0344	0.0461	0.0138	0.0203	0.0425	0.0712	0.0387	0.0531	-0.0003	-0.005	-0.0006	-0.0017	0.0008	0.0012	0.0003	0.0009
UBA	0.0435	0.0519	0.0176	0.0324	0.0472	0.0800	0.0424	0.069	0.0026	0.0019	0.0034	0.0026	0.0043	0.0062	0.0035	0.0058
RevAdv	0.0412	0.0501	0.0153	0.0231	0.0498	0.0816	0.0412	0.0633	0.0014	0.0005	0.0025	0.0012	0.0039	0.0056	0.0026	0.0041
+SharpAP	**0.0537**	**0.062**	**0.0362**	**0.0578**	**0.0626**	**0.1082**	**0.057**	**0.0897**	**0.0105**	**0.0062**	**0.0068**	**0.0031**	**0.0075**	**0.0131**	**0.0058**	**0.0103**
RAPU	0.0395	0.0486	0.0164	0.0269	0.0511	0.0937	0.0445	0.0782	0.0012	0.0004	0.0031	0.0028	0.0042	0.0060	0.0034	0.0055
+SharpAP	**0.0485**	**0.0597**	**0.0337**	**0.0546**	**0.0723**	**0.1108**	**0.0671**	**0.1015**	**0.0113**	**0.0076**	**0.0070**	**0.0035**	**0.0084**	**0.0153**	**0.0061**	**0.0093**
DADA	0.0483	0.0551	0.0266	0.0347	0.0481	0.0807	0.0386	0.0558	0.0027	0.0012	0.0043	0.0030	0.0054	0.0088	0.004	0.0067
+SharpAP	**0.0631**	**0.0727**	**0.0421**	**0.0634**	**0.0712**	**0.1085**	**0.0614**	**0.0972**	**0.0095**	**0.0068**	**0.0074**	**0.0042**	**0.0096**	**0.0146**	**0.0063**	**0.0105**
CLeaR	0.0420	0.0514	0.0221	0.0305	0.0494	0.0828	0.0426	0.0758	0.0024	0.0009	0.0038	0.0029	0.0046	0.0071	0.0042	0.0070
+SharpAP	**0.0588**	**0.0690**	**0.0417**	**0.0603**	**0.0736**	**0.1093**	**0.0677**	**0.1103**	**0.0086**	**0.0052**	**0.0069**	**0.0038**	**0.0090**	**0.0122**	**0.0060**	**0.0101**
DDSP	0.0497	0.0560	0.0277	0.0338	0.0460	0.0775	0.0451	0.0776	0.0022	0.0007	0.0031	0.0024	0.0040	0.0069	0.0044	0.0073
+SharpAP	**0.0612**	**0.0709**	**0.0398**	**0.0587**	**0.0680**	**0.0993**	**0.0692**	**0.1286**	**0.0083**	**0.0046**	**0.0059**	**0.0036**	**0.0030**	**0.0108**	**0.0065**	**0.0122**

stable embeddings that are located farther from real users in the latent space [21], [29].

- Finally, as demonstrated in these Tables, our proposed *SharpAP* consistently achieves superior attack performance compared to all baselines across three benchmark datasets under varying victim models. The core advantage of *SharpAP* lies in its ability to enhance poisoning robustness through poisoning the worst-case model instead of a fixed surrogate model. This ability effectively alleviates the overfitting of poisoned data to the surrogate model, thereby enhancing the transferability of the attack.

2) Group Attack Performance: As mentioned in Section III-C, we can use the group attack objective to assist *SharpAP* in performing group attacks. Here, we compare the group attack performance of different methods in Table VIII. Since the MovieLens-1M dataset includes user profile information, we adopt it as the evaluation dataset and divide users into two groups based on gender: a female group ($|U_0| = 1{,}705$) and a male group ($|U_1| = 4{,}309$). Users with missing gender information are excluded. Our findings are as follows:

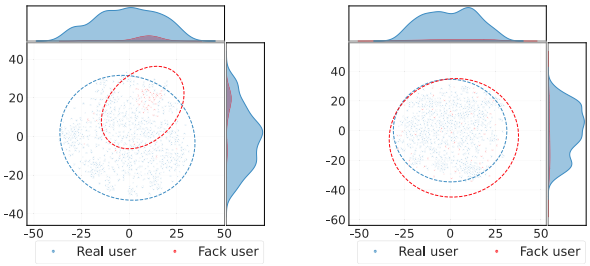
- All methods perform worse on unpopular target items compared to popular ones, meaning that cold items are harder to promote. This is likely because cold items are farther from real users in the latent space, making the attack more challenging. Similar findings have also been reported in [21], [29].
- Our proposed *SharpAP* consistently outperforms all baselines on popular and unpopular items. Specifically, *SharpAP* improves RevAdv by 30% in terms of D@10 for popular items and 600% for unpopular items in WRMF, respectively.
- On all three backbones (RevAdv, RAPU, and DADA) equipped with our method, we observed a significant performance advantage, even on unpopular items, which validates the effectiveness of our method.

C. Performance on Defense Models

To further validate the robustness of the proposed method, we evaluate full-user attacks and group attacks using existing defense models. We select two representative models, BPR and LightGCN, as victim models. The results on SGL and SimGCL are similar, we do not present them due to space limitations. We first evaluate the effectiveness of existing defense models against full-user attacks. We examine three representative defense models: PCA [24], APT [39], and PamaCF [49]. PCA detects and removes fake users from the training data, while APT injects fake users to enhance the model's robustness. PamaCF defends against poisoning attacks by dynamically assigning personalized adversarial perturbation magnitudes based on each user's embedding scale. Table IX presents the results of three representative attack methods and our *SharpAP*, evaluated on the MovieLens-1M dataset. From Table IX, we observe the following findings: 1) All defenses reduce the performance of all attackers, demonstrating their effectiveness in defending against attacks. 2) However, even with the defense models in place, *SharpAP* consistently achieves higher hit ratios than the baseline attackers. To understand *SharpAP*'s evasion capabilities, we conduct a visualization case study on the representative LightGCN victim model. Specifically, we randomly select 1000 real users and all fake users (i.e., 60) in the MovieLens-1M dataset, respectively. Fig. 5 shows that while RevAdv's fake users form tight clusters, *SharpAP*'s are dispersed among real users. This demonstrates that *SharpAP*'s curvature-aware optimization effectively minimizes statistical footprints. We also conduct experiments against group attacks, and the results are shown in Table X. The target items are randomly selected from the popular group. We can observe that all defense methods reduce the $D@10$ value, indicating effective resistance against attacks. The defense results prove the robustness of *SharpAP*, i.e., the sharpness-aware attack can generate more robust poisoned data.

TABLE IX
ALL-USER ATTACK PERFORMANCE UNDER THREE DEFENSE MODELS, EVALUATED ON MOVIELENS-1M DATASET USING H@10

Attacker	BPR				LightGCN			
	Attack	PCA	APT	PamaCF	Attack	PCA	APT	PamaCF
RevAdv	0.0125	0.0121	0.0113	0.0119	0.0186	0.0184	0.0170	0.0167
+SharpAP	0.0181	0.0162	0.0154	0.0146	0.0259	0.0241	0.0253	0.0216
RAPU	0.0121	0.0118	0.0115	0.0117	0.0191	0.0186	0.0179	0.0160
+SharpAP	0.0177	0.0173	0.0159	0.0147	0.0261	0.0255	0.0247	0.0223
DADA	0.0138	0.0120	0.0124	0.0118	0.0211	0.0198	0.0195	0.0188
+SharpAP	0.0194	0.0187	0.0176	0.0173	0.0248	0.0233	0.0219	0.0204

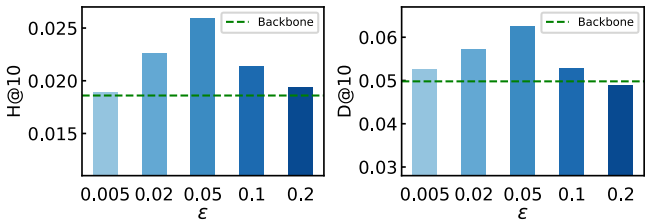


(a) RevAdv (b) RevAdv+SharpAP

Fig. 5. Visualization of user representations with t-SNE. The top and right curve graphs display the marginal distributions for two reduced dimensions.

TABLE X
GROUP ATTACK PERFORMANCE UNDER THREE DEFENSE MODELS, EVALUATED ON MOVIELENS-1M DATASET USING D@10

Attacker	BPR				LightGCN			
	Attack	PCA	APT	PamaCF	Attack	PCA	APT	PamaCF
RevAdv	0.0153	0.0136	0.0128	0.0123	0.0498	0.0463	0.0458	0.0445
+SharpAP	0.0362	0.0254	0.0279	0.0234	0.0626	0.0591	0.0574	0.0522
RAPU	0.0164	0.0143	0.0136	0.0128	0.0511	0.0416	0.0453	0.0390
+SharpAP	0.0337	0.0271	0.0224	0.0207	0.0723	0.0665	0.0582	0.0564
DADA	0.0266	0.0253	0.0248	0.0211	0.0481	0.0440	0.0421	0.0413
+SharpAP	0.0421	0.0322	0.0345	0.0304	0.0712	0.0593	0.0574	0.0558



(a) Full-user attacks (b) Group attacks

Fig. 6. Performance comparisons under different perturbation radius ϵ using RevAdv as the backbone.

D. Parameter Sensitivity

1) Perturbation Radius ϵ: The perturbation radius ϵ in (8) is a critical hyperparameter in our proposed method, defining the search space for the worst-case model. In all experiments, we set $\epsilon = 0.05$. To further analyze its effect on attack performance, we explore different values from the set $\{0.005, 0.02, 0.05, 0.1, 0.2\}$. We evaluate the effect of the perturbation radius ϵ on both full-user and group attacks, using RevAdv as the backbone and LightGCN as the victim model. The experiments are conducted on the MovieLens-1M dataset, and results are presented in Fig. 6. We have several observations from Fig. 6. First, a smaller ϵ causes the model to regress to the backbone attack due to the limited search space. Second, with the increase of ϵ, this implies the enhancement of perturbation space, *SharpAP* can explore a larger space to find the worst-case model, which achieves a better attack performance. However, a larger ϵ may cause the model to deviate from the training objective of the recommendation system, resulting in a poor attack performance. Finally, *SharpAP* achieves the best attack results with a moderate value of perturbation radius ϵ.

2) Fake User Budget Percentage δ: In this section, we investigate the effect of fake user budget percentage δ in the attackers' capability constraint $|U^f| \leq \delta |U^r|$. We systematically vary δ across $\{1\%, 3\%, 5\%, 7\%, 9\%\}$ and conduct evaluations on multiple attack scenarios using LightGCN as the victim model. To ensure clarity, we restrict our comparison to representative and powerful baselines: RevAdv, DADA, RAPU, and AUSH. As shown in Fig. 7(a) and (b), attack effectiveness exhibits a positive correlation with fake user budget percentage δ in full-user attack scenarios across both MovieLens-1M and Gowalla datasets. Notably, our method consistently outperforms all baselines, as evidenced by the superior positioning of its performance curve (blue). A critical observation reveals that performance gains scale more prominently on MovieLens-1M than on Gowalla with increasing δ. This phenomenon can be attributed to the inherent characteristics of the datasets, as MovieLens-1M has only 3,232 items whereas Gowalla contains 14,007 items, making it more difficult to promote target items to the Top-10 recommendation list. Fig. 7(c) presents group attack performance on MovieLens-1M. Consistent with previous findings, all attackers demonstrate enhanced performance with δ increasing. Meanwhile, our proposed *SharpAP* achieves the best attack performance. These two attack scenarios further validate the robustness of our method against varying fake user budget percentages.

3) Maximum number of interactable items N: Following prior works [18], [31], [47], we constrain the number of items that each fake user can interact with to be no greater than the average number of items interacted with by real users., i.e., $||\mathbf{R}^f[u^f]||_0 \leq N$ for each fake user u^f. To further investigate the effect of behavioral patterns, we systematically modulate the interaction capacity using scaling ratios $\alpha \in \{0.5, 1, 2, 3, 4\}$, yielding adjusted interaction thresholds αN. Fig. 8 reveals a non-monotonic relationship between the number of interactions and attack performance in both full-user and group attack scenarios. Specifically, performance initially improves as interaction capacity increases, reaching an optimal threshold, after which it begins to decline. This suggests that an excessive number of interactable items may introduce noisy signals, ultimately weakening the effectiveness of the attack. Furthermore, compared with these powerful baselines, our proposed method demonstrates robust superiority under different maximum numbers of items.

E. Visualization of Loss Landscape

To validate that our method effectively reduces the sharpness of the attack loss landscape, we visualize the full-user attack loss landscape using the methodology from [19]. First, we fix the surrogate model parameters after training as θ^*. Then, we generate two random perturbation vectors $w_1, w_2 \in \mathbb{R}^d$ (where $d = \dim(\theta^*)$) from the standard normal distribution. Next, we sample 20 equidistant values for m and n within $[-10, 10]$, forming a 20×20 grid. For each grid point (m_i, n_j), compute the perturbed parameters $\theta_{ij} = \theta^* + m_i w_1 + n_j w_2$, simulating the shift of model. Finally, we calculate the attack loss $\mathcal{L}_{atk}(\theta_{ij}; \mathbf{R}^r)$ for each θ_{ij}. As shown in Fig. 9, our method produces a markedly smoother loss landscape compared to the backbone. This smoothness indicates that the attack remains effective even when victim model parameters deviate from θ^*,

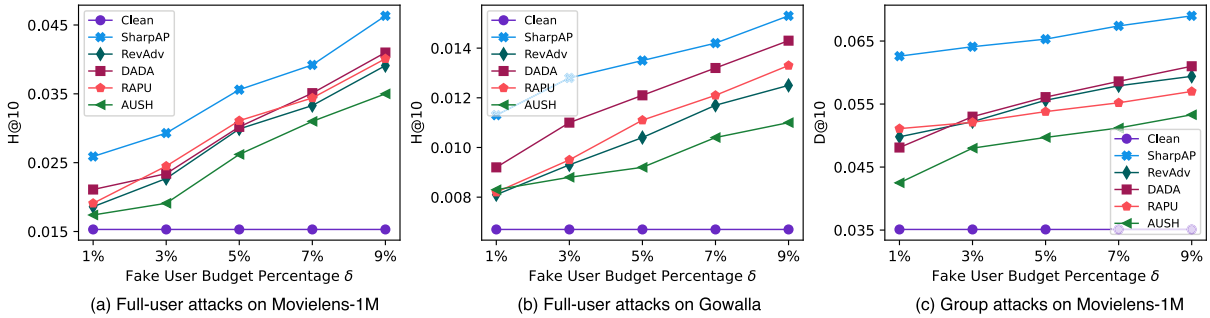


Fig. 7. The attack performance under different fake user budget percentages. δ is the percentage of fake users relative to the number of real users U^r.

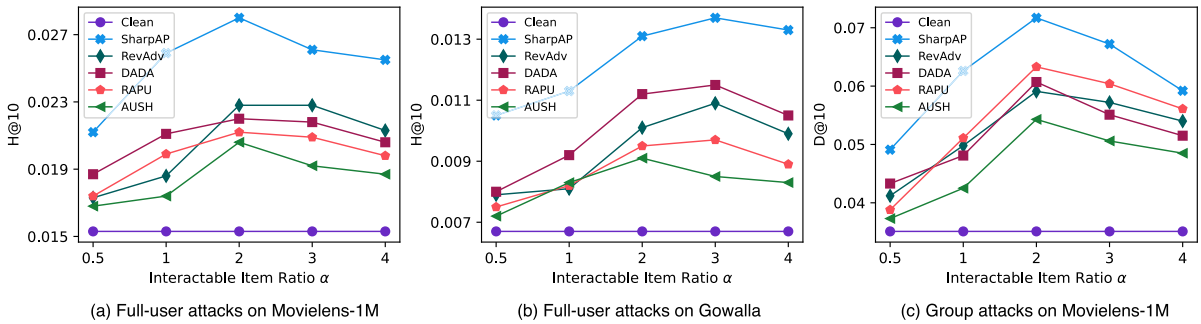


Fig. 8. The attack performance under different ratios of interactable items. α denotes the scaling ratio relative to the average number of items interacted with by real users.

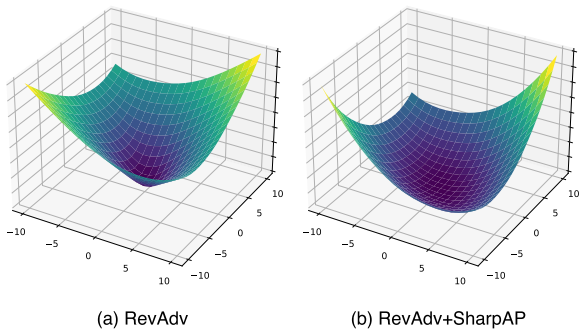


Fig. 9. Visualization of loss landscape on Movielens-1M trained with or without SharpAP.

corroborating our sharpness-aware design's ability to enhance the transferability.

F. Running Time Comparison

In this section, we report the running time of our proposed method on the MovieLens-1M dataset. All experiments were conducted on a server equipped with a single NVIDIA A40 GPU. As summarized in Table XI, the results indicate that *SharpAP* incurs a marginal computational overhead (approximately 5%) compared to the baseline methods. We argue that this modest increase in time is acceptable, given that *SharpAP* significantly enhances attack transferability.

TABLE XI
RUNNING TIME (S) ON MOVIELENS-1M

	RevAdv	RAPU	DADA	CLeaR	DDSP
Baseline	465.22	459.48	523.74	438.62	372.34
+SharpAP	483.09	480.37	538.16	455.04	388.91

V. Related Work

A. Recommender Systems

Modern recommender systems have shown remarkable performance in enhancing user experience by effectively aligning the diverse preferences of individual users with a wide range of available items [45], [50]. These systems predominantly utilize CF to generate recommendations. Given a user-item rating matrix, item-based CF approaches focus on calculating the similarity in item behavior, leveraging these similarity scores to make personalized recommendations [12], [27]. Subsequently, matrix factorization-based CF models have become increasingly prevalent due to their ability to better capture nuanced user preferences and deliver more accurate, personalized recommendations. A central challenge for these matrix factorization models is the learning of high-quality embeddings, which are crucial for optimizing recommendation performance. For instance, WRMF [14] is a well-established latent factor model that applies matrix factorization to derive user and item embeddings. Similarly, BPR [26] introduces a pairwise ranking

loss function, which has been widely adopted in recommendation systems that rely on implicit feedback. Since user-item interactions inherently form a bipartite graph, researchers have proposed neural graph-based models to capture higher-order collaboration signals in the learned embeddings. A notable example is LightGCN [11], which updates the embeddings of users and items iteratively by aggregating neighborhood embeddings from previous layers to encode these higher-order relationships. Additionally, to enhance the accuracy and robustness of GCNs for recommendation, SGL [41] augments the classical supervised recommendation task with an auxiliary self-supervised task. This task enhances node representation learning by maximizing the agreement between different views of the same node. As novel recommendation models continue to emerge, they present significant challenges to existing poisoning methods, particularly those that generate poisoned data based on a fixed surrogate model.

B. Injective Attacks

From an attacker's perspective, poisoning attacks are designed to manipulate RS. While untargeted attacks aim to erode overall recommendation quality, targeted attacks seek to promote or demote specific items within distinct user groups (i.e., group attacks) or across all users (i.e., full-user attacks) [36]. This paper focuses on targeted attacks, which are the most extensively studied category in RS [18], [35], [51]. Attackers can easily introduce bias into RS by injecting some fake users (i.e., injective attacks). Existing methods on injective attacks against RS can be broadly categorized into three paradigms: heuristic-based, neural network-driven, and gradient-based attacks. Heuristic-based attacks leverage the insight that similar users tend to share similar interests, and they rely on manually crafted fake profiles. For example, in a random attack[16], fake users interact with target items as well as a set of randomly selected items. Popular attack [25], [47] not only gives high ratings to the target item but also to several popular items, thereby enhancing the attack's effectiveness. However, heuristic attacks are unable to account for all recommendation patterns, leading to limited performance. Neural network-driven attacks propose neural networks to learn probability distributions of selected items for each fake user. Specifically, these methods aim to assign high scores to target items for fake users while ensuring that the generated fake profiles closely resemble real users as much as possible. For example, AUSH [21] employs a tailored GAN network to generate fake user profiles. However, LegUP [22] critiques AUSH for employing an indirect generation loss that is only loosely connected to the reconstruction of selected user data. This design choice may limit the overall effectiveness of the attack. To address this limitation, LegUP enhances AUSH by integrating a surrogate model, thereby further improving the poisoning performance. Gradient-based attacks relax the discrete fake user behaviors into continuous values and directly optimize by maximizing the attack objective. For example, RevAdv [29] uses a more accurate gradient calculation method. DADA [18] proposes a difficulty and diversity-aware objective function, which maximizes the attack performance. CLeaR [35] employs a dual-objective strategy that promotes a smoother spectral value distribution to broaden user reachability while simultaneously optimizing a rank promotion objective. DDSP [51] employs a dual-promotion objective to simultaneously promote both target items and user-preferred items, thereby ensuring attack stealthiness. Although these methods achieve good performance, they fail to explicitly model transferability. Table XII summarizes the existing attacks.

TABLE XII
SUMMARY OF EXISTING ATTACKERS

Attack Paradigm	Attacker	Mechanism for Transferability
Neural network–based	AUSH [21]	GAN
	LegUP [22]	GAN
Gradient–based	RevAdv [29]	Improved gradient calculation
	DADA [18]	Difficulty and diversity aware
	CLeaR [35]	Dispersion and rank promotion
	DDSP [51]	Diversity aware dual–promotion
	SharpAP (ours)	Sharpness–aware minimization

C. Sharpness-Aware Minimization

Sharpness-aware minimization [13], [37] is a highly effective regularization technique that enhances the model's generalization across various settings. A lower sharpness value is generally associated with better generalization performance. Specifically, sharpness-aware minimization achieves improved generalization by minimizing the maximum loss within a neighborhood of the current parameter, rather than optimizing the loss at a single point. This strategy yields solutions that are more robust to small parameter perturbations, thereby enhancing both generalization and robustness [20]. Building on this idea, recent studies [7] and [52] independently propose minimizing the loss in the direction of the worst-case perturbation from the current parameter to improve generalization. Similarly, the work [40] introduces a nearly identical method aimed at improving the robust generalization of adversarial training. In addition, ASAM [15] dynamically adjusts the perturbation region according to the scale of the model weights. ImbSAM [54] extends the applicability of sharpness-aware minimization to scenarios with highly imbalanced data distributions, effectively addressing the trade-off between sharpness minimization and data imbalance. Recently, in computer vision, the work [10] experimentally demonstrated that using the sharpness-aware principle can improve attack transferability when the perturbation is continuous. However, directly applying sharpness awareness in recommender systems faces challenges such as optimizing over discrete data and the lack of theoretical justification.

VI. CONCLUSION

In this paper, we propose *SharpAP*, a novel method to enhance the cross-model transferability of injective attacks on recommender systems. Specifically, we argue that existing methods for generating poisoned data based on a fixed surrogate model fundamentally rely on a precarious assumption. This reliance causes the poisoned data to overfit the surrogate model, resulting in poor transferability when encountering model structure shifts.

To address this issue, we introduce the sharpness-aware minimization principle to seek the approximately worst-case model during the attack process. By iteratively optimizing poisoned data against the worst-case model instead of a fixed surrogate model, we enhance transferability across various victim models. Our method can be formulated as a sharpness-aware min-max-min tri-level optimization problem, where the maximization performs a bounded perturbation to seek the worst-case model. Extensive experiments on three real-world datasets across representative attacks demonstrated the effectiveness of the proposed *SharpAP*. In the future, we will explore defense methods against sharpness-aware poisoning.

REFERENCES

[1] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2005, pp. 17–24.

[2] Z. Chen, J. Zhang, Y. Kou, X. Chen, C.-J. Hsieh, and Q. Gu, "Why does sharpness-aware minimization generalize better than SGD," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 72325–72376.

[3] L. Cheng, X. Huang, J. Sang, and J. Yu, "Towards Robust Recommendation: A Review and an Adversarial Robustness Evaluation Library," *IEEE Trans. Knowl. Data Eng.*, vol. 37, no. 9, pp. 5679–5698.

[4] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 1082–1090.

[5] W. Fan et al., "Attacking black-box recommendations via copying cross-domain user profiles," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 1583–1594.

[6] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to Top-N recommender systems," in *Proc. ACM Web Conf.*, 2020, pp. 3019–3025.

[7] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–20.

[8] S. Guo, T. Bai, and W. Deng, "Targeted shilling attacks on GNN-based recommender systems," in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 649–658.

[9] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.

[10] P. He et al., "Sharpness-aware data poisoning attack," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 1–21.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[12] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2004, pp. 337–344.

[13] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–17.

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

[15] J. Kwon, J. Kim, H. Park, and I. K. Choi, "ASAM: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5905–5914.

[16] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 393–402.

[17] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 1893–1901.

[18] H. Li, S. Di, and L. Chen, "Revisiting injective attacks on recommender systems," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 29989–30002.

[19] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 6391–6401.

[20] T. Li, P. Zhou, Z. He, X. Cheng, and X. Huang, "Friendly sharpness-aware minimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 5631–5640.

[21] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, "Attacking recommender systems with augmented user profiles," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 855–864.

[22] C. Lin, S. Chen, M. Zeng, S. Zhang, M. Gao, and H. Li, "Shilling black-box recommender systems by learning to generate fake user profiles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1305–1319, Jan. 2022.

[23] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 43–52.

[24] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *User Model. User-Adapted Interaction*, vol. 19, pp. 65–97, 2009.

[25] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Trans. Internet Technol.*, vol. 7, no. 4, pp. 23–es, 2007.

[26] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[28] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 565–573.

[29] J. Tang, H. Wen, and K. Wang, "Revisiting adversarially learned injection attacks against recommender systems," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 318–327.

[30] C. Wang, H. Zhu, C. Zhu, C. Qin, and H. Xiong, "SetRank: A setwise Bayesian approach for collaborative ranking from implicit feedback," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 6127–6136.

[31] W. Wang, C. Wang, F. Feng, W. Shi, D. Ding, and T.-S. Chua, "Uplift modeling for target user attacks on recommender systems," in *Proc. ACM Web Conf.*, 2024, pp. 3343–3354.

[32] Z. Wang, M. Gao, J. Li, J. Zhang, and J. Zhong, "Gray-box shilling attack: An adversarial learning approach," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, pp. 1–21, 2022.

[33] Z. Wang et al., "Id-free not risk-free: LLM-powered agents unveil risks in id-free recommender systems," in *Proc. 48th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2025, pp. 1902–1911.

[34] Z. Wang, M. Gao, J. Yu, S. Sadiq, H. Yin, and L. Liu, "When graph contrastive learning backfires: Spectral vulnerability and defense in recommendation," *ACM Trans. Inf. Syst.*, vol. 44, no. 2, pp. 1–30, 2026.

[35] Z. Wang, J. Yu, M. Gao, H. Yin, B. Cui, and S. Sadiq, "Unveiling vulnerabilities of contrastive recommender systems to poisoning attacks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2024, pp. 3311–3322.

[36] Z. Wang et al., "Poisoning attacks and defenses in recommender systems: A survey," 2024, *arXiv:2406.01022*.

[37] K. Wen, T. Ma, and Z. Li, "How sharpness-aware minimization minimizes sharpness," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 1–76.

[38] C. Wu, D. Lian, Y. Ge, Z. Zhu, and E. Chen, "Triple adversarial learning for influence based poisoning attack in recommender systems," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 1830–1840.

[39] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, and S. Yuan, "Fight fire with fire: Towards robust recommender systems via adversarial poisoning training," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 1074–1083.

[40] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2958–2969.

[41] J. Wu et al., "Self-supervised graph learning for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 726–735.

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

[43] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang, "Collaborative neural social recommendation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 464–476, Jan. 2021.

[44] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–15.

[45] Y. Yang et al., "Generative-contrastive graph learning for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 1117–1126.

[46] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? Simple graph contrastive learning for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1294–1303.
[47] H. Zhang et al., "Data poisoning attack against recommender system using incomplete and perturbed data," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 2154–2164.
[48] K. Zhang, Q. Cao, Y. Wu, F. Sun, H. Shen, and X. Cheng, "Improving the shortest plank: Vulnerability-aware adversarial training for robust recommender system," in *Proc. 18th ACM Conf. Recommender Syst.*, 2024, pp. 680–689.
[49] K. Zhang, Q. Cao, Y. Wu, F. Sun, H. Shen, and X. Cheng, "Understanding and improving adversarial collaborative filtering for robust recommendation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, vol. 37, pp. 120381–120417.
[50] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, 2019.
[51] Y. Zhao, T. Chen, J. Yu, K. Zheng, L. Cui, and H. Yin, "Diversity-aware dual-promotion poisoning attack on sequential recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2025, pp. 1634–1644.
[52] Y. Zheng, R. Zhang, and Y. Mao, "Regularizing neural networks via adversarial model perturbation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8156–8165.
[53] G. Zhou et al., "Deep interest network for click-through rate prediction," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1059–1068.
[54] Y. Zhou, Y. Qu, X. Xu, and H. Shen, "ImbSAM: A closer look at sharpness-aware minimization in class-imbalanced recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11345–11355.

Junsong Xie received the master's degree from the University of Science and Technology of China. He is currently working toward the PhD degree with the Hefei University of Technology, China. He has authored or coauthored several papers in referred conferences and journals, such as IJCAI and *Frontiers of Computer Science*. His research interests include data mining and recommender systems.

Yonghui Yang received the PhD degree from the Hefei University of Technology, China. He is currently a research fellow with the National University of Singapore. He has authored or coauthored more than 10 papers in referred journals and conferences, such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Big Data*, KDD, SIGIR, IJCAI, and ACM Multimedia. His research interests include data-centric recommendation and LLM safety.

Pengyang Shao received the bachelor's degree from the Hefei University of Technology, China, where he is currently working toward the PhD degree. He has authored or coauthored several papers in leading conferences and journals, including KDD, WWW, ACM TOIS, and SCIS. His research interests include data mining, and large language models.

Le Wu (Member, IEEE) received the PhD degree from the University of Science and Technology of China. She is currently a professor with the Hefei University of Technology, China. She has authored or coauthored more than 70 papers in leading journals and conferences, such as *IEEE Transactions on Knowledge and Data Engineering*, TOIS, WWW, SIGIR, KDD, and NeurIPS. Her research interests include data mining and knowledge engineering, personalized recommendation, trustworthy user modeling and applications. She is an associate editor for *IEEE Transactions on Big Data*, *AI Open*, and *Frontiers of Computer Science*.