Summary of Changes

We would like to thank the reviewers for providing many constructive comments and valuable suggestions which are immensely helpful for improvement of our work. In the following, we first summarize the major revisions we made in the revised manuscript and then give detailed responses, which explain how we addressed each concern or problem, and describe the changes we made in the revised manuscript.

- **Readability** (**R1W1** and **R3O1**): We add many examples and figures for illustration of various scenarios and concepts. We also revise various parts of the manuscript with intuitions and detailed explanation (as appropriate). We believe that the revised manuscript is now more accessible and easier to follow than the original submission.
- **Soundness of theoretical analysis** (**R1W2** and **R4O4**): We present a revised theoretical analysis. We present a new theorem with proof on the existence of equilibrium and a new theorem with proof on guarantee of convergence. Moreover, we point out the connection between our theoretical analysis and the proposed method.
- **Additional experiments** (**R4O1** and **R4O2**): We conduct additional experiments to include running time measurements to validate our time complexity analysis and present new experiment results on an additional dataset (LibraryThing).
- **Unclear descriptions or definitions** (**R3O2** and **R3O4**): We clarify some definitions and provide justifications regarding the attacker's capacity and experiment evaluation settings.
- **Omissions and typos** (**R3O3**, **R3O5**, and **R4O3**): We fix the typos and add missing information in abstract and references. We also thoroughly proofread the revised manuscript.
- **Contributions** (**R1W3**): We clarify that our problem setting is complex due to the need to explore scenarios more realistic than previous work. In the revision, we make our methodological contributions explicit and present additional contributions on new techniques introduced in the revision to trade off the cost to reach equilibrium for efficiency.
- **Concern on ICDE suitability** (**R1A0**): We indicate the importance of our research topic to the field of data engineering and strengthen our connection to ICDE as well as other data engineering venues with references to prior works on similar topics in these venues.

In addition, as several long responses go across multiple pages, we provide the following page references for quick navigation:

Response to Reviewer #1

W1 : **The problem setting is complex or presented in a complex manner; and the proposed approach is not explained very intuitively. The paper possibly needs examples and other figures like Fig. 1, to explain how the different components of the solution (e.g., the computed gradients and binarized vectors) work intuitively. The paper contains a lot of notation and terminology, which is not balanced by intuitive descriptions. It is thus difficult to read.**

Reply: Thank you for your helpful feedback on the readability of our paper. As suggested, we have made every effort to revise and improve the manuscript to make it easier to read.

The changes we make in the revised manuscript are mainly in two aspects: 1) we add many intuitive descriptions to enhance the coherence and readability in each section; 2) we include a lot more examples and figures for illustration of our approach and explain in detail the problem setting and the proposed method. Specifically, the changes in in response to your comments above are as follows.

1) In INTRODUCTION (Section I), we revise the explanation of our problem setting to include intuitive descriptions and motivation along with better figure illustrations. We also include high-level illustrations of our proposed MSOPDS method to accompany the respective descriptions.
2) In PROBLEM FORMULATION (Section III), we revise the definition of Injection Attack (Definition 3), Comprehensive Attack (Definition 4), and Multiplayer Comprehensive Attack (Definition 5) to include detailed intuitive explanations of the objectives $\mathcal{L}$ and capacity sets $\mathcal{C}$.
3) In THE MSOPDS METHOD (Section IV) we include high-level descriptions on the formulation of the importance vector as well as the two components MSO and PDS. We provide intuitions for MSO and intuitive figure examples for the explanations PDS mechanism to aid understanding.

In the following, we detail the changes made in the revised manuscript. Tables I and II provide a notation table and an abbreviation table for reference.

Revisions in INTRODUCTION for the problem setting.

Our problem addresses data poisoning attacks against Heterogeneous RecSys in a multiplayer setting. *Data poisoning attack* involves manipulating a machine learning system's training data to achieve a specific (malicious) goal. Attacks against recommender systems (RecSys) are very serious because abusing the recommendations can result in a significant financial loss and customer dissatisfaction, particularly in eCommerce. However, previous works have only considered the case of a single adversary attempting a data poisoning attack and have focused on basic RecSys, instead of the more advanced Heterogeneous RecSys.

In INTRODUCTION, Fig. 1 presents a comparison of the problem setting between our work and prior works, and Fig. 2

TABLE I: List of Notations.

symbol	descriptions
u, i	An individual user or an item.
i_t^p	The target item of player p.
$\mathcal{U}, \mathcal{I}$	The (real) user set and item set.
$\mathcal{U}_{fake}^p$	The set of fake users created by player p. $\mathcal{U}_{fake}^p \not\subseteq \mathcal{U}$
$\mathcal{U}_{TA}^p$	The target audience of player p.
$\mathcal{U}_{base}^p$	The customer base of player p.
$\mathcal{I}_{product}^p$	The company products of player p.
$\mathcal{I}_{compete}^p$	The competing items of player p.
Ξ, $\varnothing$	The set of ratings values $\{1, 2, 3, 4, 5\}$ or missing value
$\mathbf{R}$	The rating matrix ($\mathbf{R} \in \Gamma^{\lvert\mathcal{U}\rvert \cdot \lvert\mathcal{I}\rvert}, \Gamma \equiv \Xi \cup \{\varnothing\}$).
$\mathcal{G}_U$, $\mathcal{G}_I$	The social network and item graph.
$\mathcal{G}$	The heterogeneous graph information ($\mathcal{G} \equiv \mathcal{G}_U \cup \mathcal{G}_I$).
$\mathcal{R}(\boldsymbol{\theta}, \mathcal{G})$	The rating predictions of a RecSys with parameter $\boldsymbol{\theta}$.
$\hat{\mathbf{R}}, \hat{\mathcal{G}}$	The poisoned ratings and the poisoned graph data.
$\mathcal{R}_{(u,i)}$	The ratings predictions (between user u and item i).
$\boldsymbol{h}_u, \boldsymbol{h}_i$	The initial embeddings of user u or item i.
$\boldsymbol{h}_u^{(f)}, \boldsymbol{h}_i^{(f)}$	The final embeddings of user u or item i.
$\mathcal{L}_{train}$	RecSys training loss, i.e., the Mean Square Error loss.
$\mathcal{L}^p$	The adversarial loss, i.e., the poisoning objective (of p).
$\mathcal{C}^p$	The capacity set, i.e., candidate poisoning actions (of p).
$\mathcal{X}^p$	The poisoning action plan of player p
$(\hat{\mathbf{X}}^p)$ $\mathbf{X}^p$	The (binarized) importance vector of player p.

TABLE II: List of Abbreviations.

acronym	full phrase and descriptions
Het-RecSys	Heterogeneous RecSys, a Recommender System that utilizes heterogeneous graph information $\mathcal{G}$. In contrast, basic RecSys only utilizes the rating records $\mathbf{R}$.
MCA	Multiplayer Comprehensive Attack. Our problem setting which concerns how to conduct data poisoning attack against Het-RecSys while anticipating for subsequent opponent poisonings.
CA	Comprehensive Attack. A poisoning attack setting that targets Het-RecSys but assumes single adversary.
IA	Injection Attack. Prior works' poisoning attack setting. Targets basic RecSys and assumes single adversary
MSOPDS	Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate, our proposed method.
MSO	Multilevel Stackelberg Optimization, provides update rules to approach equilibrium for an attacker facing opponents.
PDS	Progressive Differentiable Surrogate, provides surrogate model for gradient computation over RecSys training.
BOPDS	Bi-level Optimization over Progressive Differentiable Surrogate, an ablation of MSOPDS that replaces MSO with a bi-level framework.

illustrates our Multiplayer Comprehensive Attack (MCA) problem.

1) As shown in Fig. 1, our problem setting, MCA, differs from prior works in two ways:
   a) We consider multiple adversaries, while prior works only consider a single adversary.
   b) We target Heterogeneous RecSys, while prior works target basic RecSys.
2) Fig. 2 motivates the study of MCA.
   a) Because RecSys is open (may be poisoned) and public (data available to the public), multiple attackers may add poisoning edges to the system with different objectives. However, as the later-added poisoning edges may interfere with the effort made by the first attacker, it is important to consider the actions of subsequent adversaries from the perspective of the first attacker in this setting.
   b) The difference between a basic RecSys and a more advanced Heterogeneous RecSys (Het-RecSys) is that the Het-RecSys processes three types of information (rating records, social network, and the item graph), while the basic RecSys only processes rating records. In other words, the factors considered in producing a recommendation for Het-RecSys are different, and it is thus essential to explore the vulnerabilities of additional types of poisoning actions.

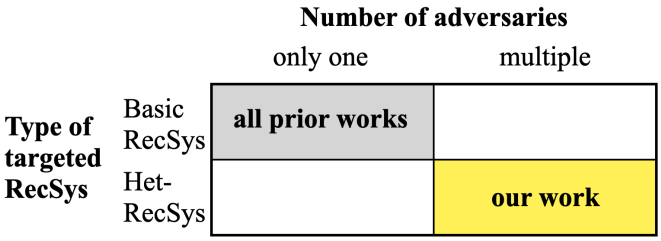


Fig. 1: Comparison of our problem setting with prior works. Our work studies data poisoning attack against Heterogeneous RecSys (Het-RecSys) under a multiple adversary setting.

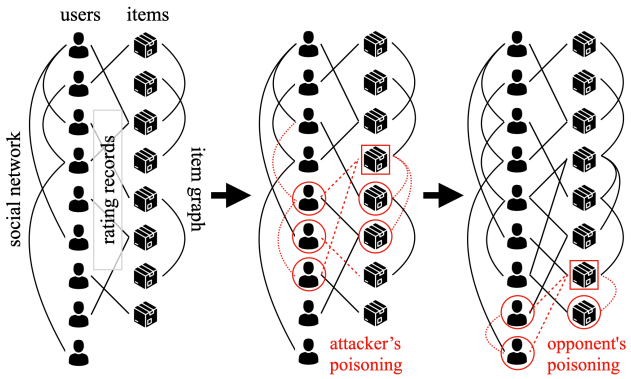


Fig. 2: Illustration of the Multiplayer Comprehensive Attack problem. Red lines indicate poison edges. After an adversary (attacker) poisons the RecSys for his objective, another adversary (opponent) may later poison the RecSys with a different goal. Thus, if the first attacker does not anticipate the *subsequent* opponent's poison actions, his poison effort may be voided by the following opponent's poisons.

Revisions in PROBLEM FORMULATION for the formal definition of MCA.

In PROBLEM FORMULATION, we rewrite the definitions of attack scenarios to streamline our explanation. Formally, we define all prior works as Injection Attack (IA) and our work as Multiplayer Comprehensive Attack (MCA). We also define Comprehensive Attack (CA) which targets Het-RecSys but

does not consider the problem setting of multiple adversaries. A comparison of these three attack scenarios is presented in Table III.

TABLE III: Comparison of attack scenarios.

	IA	**CA**	**MCA**
targeted RecSys	basic	Het-RecSys	Het-RecSys
# of adversary	1	1	multiple

In particular, both IA and CA can be solved using the *bi-level optimization framework* while MCA requires *multilevel optimization*. On the other hand, both CA and MCA targets Het-RecSys. Thus, they share the same capacity $\mathcal{C}$ and loss objective $\mathcal{L}$, which is different from IA. Our new revisions on the formal descriptions are presented as follows.

1) **Definition 2 (Bi-level formulation of data poisoning attack)** presents the optimization framework for the scenarios of *single attacker*, including Injection Attack (IA) and Comprehensive Attack (CA). We revise the manuscript to highlight that IA and CA ascribe to the single player scenario and is formulated by the Bi-level optimization framework by specifying the poisoning objective $\mathcal{L}$ and the capacity $\mathcal{C}$ as follows.

$$\begin{aligned} &\min_{\mathcal{X}\subseteq\mathcal{C}} \mathcal{L}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}), \end{aligned} \tag{1}$$

where the poisoning objective $\mathcal{L}$ is optimized by selecting a poisoning plan $\mathcal{X}$ from the capacity $\mathcal{C}$ in the first layer and the RecSys training is operated based on $\mathcal{L}_{train}$ in the second layer, with $\hat{\mathcal{G}}$ and $\hat{\mathbf{R}}$ denoting graph and rating records poisoned by $\mathcal{X}$. Intuitively, the attacker adjusts their poison data $\mathcal{X}$ (or poisoning action plan) to achieve their poisoning objective. However, to check the effect of the current poisoning, he needs to train the target RecSys with the poisoned data $\hat{\mathcal{G}}$ and rating records $\hat{\mathbf{R}}$. The training objective $\mathcal{L}_{train}$ describes how closely the RecSys predictions $\mathcal{R}$ match the poisoned records $\hat{\mathbf{R}}$, while the attacker's poisoning objective is derived on some aspect of the RecSys predictions results $\mathcal{L}(\mathcal{R})$. For instance, the objective may be to promote a particular item to all users through the recommendations of the RecSys. Finally, the attacker's poisoning plan is selected from a set of feasible actions $\mathcal{C}$ subject to a budget constraint.

2) **Definition 3 (Injection Attack (IA))**. IA subscribes to the above bi-level framework with the Injection Attack loss $\mathcal{L}_{IA}$ and the capacity set $\mathcal{C}_{IA}$. In particular, $\mathcal{L}_{IA}$, which describes the objective of promoting the target item i_t, to all users $u \in \mathcal{U}$, is written as follows.

$$\mathcal{L}_{IA} = -\frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \mathcal{R}_{(u,i_t)}. \tag{2}$$

On the other hand, Injection Attack only makes one type of poisoning action, i.e., manipulating ratings. Its capacity set $\mathcal{C}_{IA}$ can be written as follows.

$$\mathcal{C}_{IA} = \{(u, i, r) \mid u \in \mathcal{U}_{fake},\ i \in \mathcal{I}, r \in \Xi\}, \tag{3}$$

where a set of fake users $\mathcal{U}_{fake}$ is controlled to give fake rating r (selected from a rating set $\Xi \equiv [1, 2, 3, 4, 5]$) to any item $i \in \mathcal{I}$. The capacity set of IA allows the fake users ($\mathcal{U}_{fake}$) to give any fake rating (r out of 5 stars) to any item ($\mathcal{I}$). The budget constraints limit the number of items that each fake user can rate.

3) **Definition 4 (Comprehensive Attack (CA))**. CA can also be formulated with the bi-level optimization framework. However, to align with realistic scenarios, we design CA with the Comprehensive Attack loss $\mathcal{L}_{CA}$ to focus on promoting the target item i_t to its target audience $\mathcal{U}_{TA}$ over competing products $\mathcal{I}_{compete}$ as follows.

$$\mathcal{L}_{CA} = \frac{1}{|\mathcal{U}_{TA}|} \sum_{u\in\mathcal{U}_{TA}} \sum_{i_c\in\mathcal{I}_{compete}} SELU(\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)}), \tag{4}$$

where the scaled exponential linear units (SELU) [1] are used to emphasize the individual terms where the target item i_t losing to competing items $i_c \in \mathcal{I}_{compete}$. Note that the capacity set of CA, $\mathcal{C}_{CA}$, includes three types of poisoning actions: i) hiring real users from the customer base ($\mathcal{U}base$) to rate the target item (i_t) with a preset rating of 5, ii) hiring users from the customer base to connect to fake accounts in the social network, and iii) selecting real items from a set of company products ($\mathcal{I}_{product}$) to connect to the target item in the item graph. Formally, $\mathcal{C}_{CA}$ is written as follows.

$$\begin{aligned} \mathcal{C}_{CA} = &\{\,(u, i_t, \hat{r}) \mid u \in \mathcal{U}_{base}\,\} \\ &\cup \{\,(u, u_f) \mid u \in \mathcal{U}_{base},\ u_f \in \mathcal{U}_{base}\,\} \\ &\cup \{\,(i, i_t) \mid i \in \mathcal{I}_{product}\,\}. \end{aligned} \tag{5}$$

The budget constraints for $\mathcal{C}_{CA}$ limit the respective number of users and items that can be hired or selected. Overall, the capacity set of IA has one type of poisoning action. In contrast, the capacity set of CA has three types of actions, corresponding to the three types of information used by a Het-RecSys.

4) **Definition 5 (Multiplayer Comprehensive Attack (MCA)).** Finally, in the multiplayer setting, we introduce Multiplayer Comprehensive Attack (MCA). MCA involves an attacker attempting to conduct a data poisoning attack on a Het-RecSys. Still, it faces other adversaries who also attempt poisoning attacks to manipulate Het-RecSys to achieve their respective objectives. As a result, MCA cannot be solved using the basic bi-level optimization framework, as in (1). Instead, we formulate

the following multilevel optimization.

$$\begin{aligned}
&\min_{\mathcal{X}^p \subseteq \mathcal{C}^p_{CA}} \mathcal{L}^p_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\mathcal{X}^{q_1} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}^{q_1}_{CA}} \mathcal{L}^{q_1}_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\qquad \vdots \\
&\mathcal{X}^{q_N} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}^{q_N}_{CA}} \mathcal{L}^{q_N}_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}) + \lambda \|\theta\|^2,
\end{aligned} \tag{6}$$

where the opponents are denoted by $q_1 \dots q_N$. MCA utilizes the same objective $\mathcal{L}_{CA}$ and capacity $\mathcal{C}_{CA}$ as CA. However, MCA differs from CA in that it considers the perspective of an attacker facing multiple opponents (instead of a single attacker facing no other attackers). In Equation (6), the attacker adjusts his poison data, $\mathcal{X}^p$, to achieve his poisoning objective $\mathcal{L}^p$. To anticipate other adversaries' actions, he first simulates the response of each opponent q_i by consecutively optimizing his opponent's poison data $\mathcal{X}^{q_i}$ with his poison data already added to the system, and then trains the targeted RecSys with all players' poison data. Different levels of optimization in (6) represent the hierarchical sequence of actions, with the top level as the attacker selecting his poisoning actions from his capacity set, $\mathcal{X}^p \subseteq \mathcal{C}^p_{CA}$, and the next N levels as the opponents optimizing their respective plans from their capacities, $\mathcal{X}^{q_i} \subseteq \mathcal{C}^{q_i}_{CA}$. The bottom level represents the training of the targeted RecSys with all players' poison data.

The multiple levels in Equation (6) indicate the complexity of the multiplayer setting. From the perspective of the first attacker, p, whose objective is represented in the top layer of (6), we aim to find the optimal poisoning plan, $\mathcal{X}^{p*}$.

Revisions to INTRODUCTION for the proposed method.

In the INTRODUCTION section, the following high-level illustrations for the importance vector (Fig. 3, Fig. 4), Multilevel Stackelberg Optimization (MSO, Fig. 5), and Progressive Differentiable Surrogate (PDS, Fig. 6) are presented together as a full page figure with the respective captions combined. Details explanations to these figures are presented as follows.

To solve the Multiplayer Comprehensive Attack problem, we propose Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate (MSOPDS), which consists of two key components, i.e., Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), with *importance vectors* representing poisoning plans for attackers. In particular, the importance vector is utilized to record the *priority* of each poisoning action. The benefit of using importance vectors is straightforward: it is much easier to perform iterative optimizations on vectors than on sets of discrete actions.

Each element of the importance vector maps to a candidate poisoning action. Furthermore, heterogeneous types of actions can all be concatenated into the importance vector. As shown in Fig. 3, the green elements corresponds to adding poisoning edges in the social network, red elements corresponds to adding poisoning ratings, and blue elements corresponds to adding poisoning edges in the item connection graph.

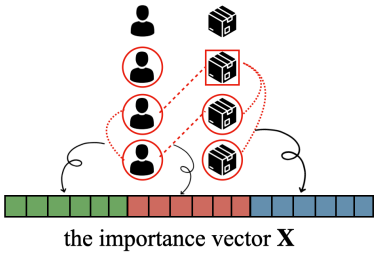


Fig. 3: Illustration of the importance vector, which collectively represents all candidate poisoning actions where each vector element corresponds to one candidate action.

However, due to the budget constraint, not all poisoning actions can be chosen. Therefore, we utilize a *binarizing step* to select a poisoning plan from the priorities recorded by the importance vector. As shown in Fig. 4, we copy the importance vector into a *binarized* version where the top values of each section (i.e., each action type) are set to 1 and the rest to 0. Then, we can obtain a poisoning plan that abides by the budget constraint by selecting the poisoning actions with 1. Therefore, if we iteratively optimize the value for each vector element on the importance vector $\mathbf{X}$ we can effectively adjusts and optimizes the poisoning action plan $\mathcal{X}$.

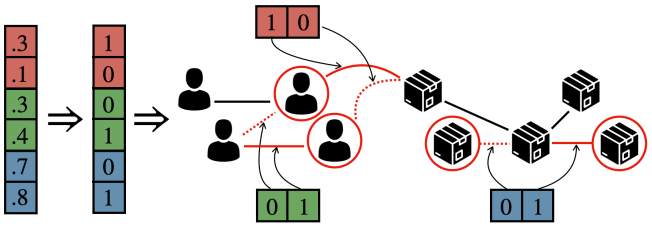


Fig. 4: Example of *binarizing* an importance vector $\mathbf{X}$. The binarizing step is the process we use to select a poisoning plan with the priorities recorded by the importance vector $\mathbf{X}$. The portrayed example shows the selection of a poisoning plan, including poisoning actions on the rating (dark red), the social network (green) and the item graph (blue). With a budget of selecting one out of two options for each of the three types of poisoning actions, the red solid lines indicate the selected actions while red dash lines indicate the candidate actions that are not selected.

Next, as shown in Fig. 5, MSO aims to iteratively approach the attacker's optimal solution (i.e., the optimal importance vector corresponding to an optimal poisoning plan) by *simultaneously* modeling both his and his opponent's actions to carefully examine the interactions between both players. In our problem setting, the attacker conducts his poisoning actions first, and then the opponent plans the subsequent

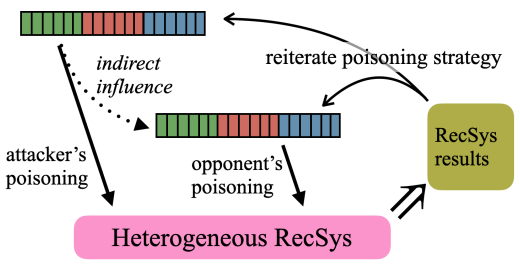


Fig. 5: Illustration of MSO. In each iteration, the opponent's vector is updated based on how the poison directly influences the opponent's objective. Note that attacker's vector also considers his *indirect influence* on the opponent since the opponent will observe the attacker's poison when deciding his poisoning plan.

poisoning action based upon the attacker's poison. Such a scenario forms a Stackelberg game. Thus, MSO aims to approach a Stackelberg equilibrium. By definition, under a game equilibrium, no player can change his plan to obtain a better gain for his objective. Intuitively, an equilibrium indicates that both players have obtained the optimal result and thus both follow the optimal poisoning plan.

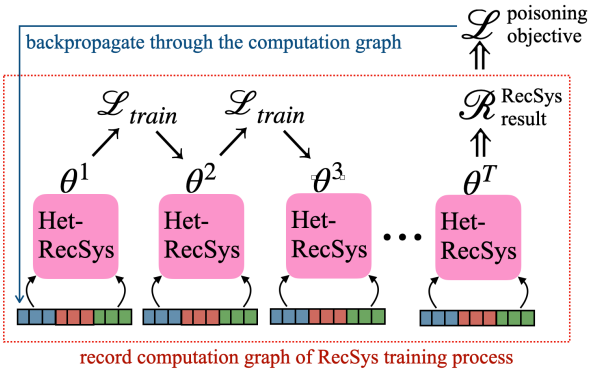


Fig. 6: Illustration of PDS. The PDS calculates the partial derivatives of the poisoning objectives w.r.t. importance vectors by incorporating it into the training process of a Het-RecSys. By recording the computation graph of the training process, the gradient is derived by backpropagation.

As detailed in the following section, MSO designs update rules composed of the partial derivatives of the poisoning objectives w.r.t. the importance vectors. As shown in Fig. 6, PDS finds the partial derivatives by incorporating the importance vectors into the training process of a Het-RecSys. This is because the training process determines the RecSys result $\mathcal{R}$, which in turn decides the poisoning objective as $\mathcal{L}(\mathcal{R})$. Specifically, to understand the process of training a RecSys and the impact of poisoning on its results, we record the computation graph of the training process. Then, by establishing how each element of the importance vector contributing to the training (poisoning) process of RecSys and to its final poisoned result $\mathcal{R}$, we can also understand how the importance vector affects the poisoning objective $\mathcal{L}$ via backpropagation. Intuitively, we consider the training process of the RecSys as a series of transitions from one iteration to the next, where the parameter obtained in each training iteration is represented by $\theta^1, \theta^2, \theta^3 \cdots \theta^T$ in Fig. 6 and the training loss denoted as $\mathcal{L}_{train}$. The RecSys result is a function of the final iteration parameter $\mathcal{R}(\theta^T)$. The poisoning objective, represented by $\mathcal{L}$, is a function of the RecSys result. Thus, by the chain rule, backpropagation through the entire process may yield the partial derivatives of the poisoning objectives w.r.t. the importance vectors.

Revisions in THE MSOPDS METHOD for the proposed method.

In THE MSOPDS METHOD (Section IV) we reorganize the subsections to introduce the importance vector in Section IV-A, MSO in Section IV-B, PDS in Section IV-C, and the full algorithm in Section IV-D. In each subsection, we present intuitions with figures and examples. Note that MSO and PDS are two components of MSOPDS, while the importance vector is used between them throughout our proposed method.

The Importance Vector.

In Section IV-A, we present the importance vector as follows. To iteratively adjusts intermediate plans by appropriate update rules, we formulate the *importance vector* $\mathbf{X} \in \mathbb{R}^{|\mathcal{C}|}$ where each element x indicates the *priority* of the corresponding candidate poisoning action in $\mathcal{C}$. Given the budget constraints for each type of poisoning actions, the importance vector $\mathbf{X}$ maps to a poisoning plan $\mathcal{X}$ consisting of the actions corresponding to the top-valued elements in $\mathbf{X}$.

Example. Following Fig. 4, suppose an attacker has the budget to hire 1 (out of 2) user to rate a product, 1 (out of 2) user to connect to another user account, and to form 1 (out of 2) item graph connections. His importance vector $\mathbf{X}$ would have 6 elements while his poisoning plan would consists of 3 poisoning actions including 1 poisoning rating, 1 poisoning edge on the social network, and 1 poisoning edge on the item graph. As shown in Fig. 4, the top elements with the top values are selected for each type of poisoning action.

Component 1: Multilevel Stackelberg Optimization.

In Section IV-B, we revise our explanation to clearly present the "update rules" and the intuition behind our design based on the following observation for an attacker facing each opponent.

1) MSO approaches the equilibrium between the attacker and the opponent by designing appropriate update rules for each player. Both players' poisoning objectives ($\mathcal{L}^p$ and $\mathcal{L}^q$) depend on the final RecSys trained with data poisoned by both players and are thus expressed as $\mathcal{L}^p(\mathbf{X}^p, \mathbf{X}^q)$ and $\mathcal{L}^q(\mathbf{X}^p, \mathbf{X}^q)$.
2) If the attacker's vector $\mathbf{X}^p$ is given, the opponent's vector $\mathbf{X}^q$ can be optimized using the standard gradient update [2]:

$$\mathbf{X}^q \leftarrow \mathbf{X}^q - \eta^q \frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}, \tag{7}$$

where η^q is the step size and $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ is the partial derivative of $\mathcal{L}^q$ with respect to $\mathbf{X}^q$. By using the partial derivative, the gradient update direction only considers the effect of the opponent's vector $\mathbf{X}^q$ and not $\mathbf{X}^p$.

3) After finding the optimal $\mathbf{X}^q$, a naive approach is to fix $\mathbf{X}^q$ and update $\mathbf{X}^p$ using the same approach. However, this naive approach does not take into account that the current $\mathbf{X}^q$ was solved according to the given $\mathbf{X}^p$. As a result, it would not be optimal for the opponent after $\mathbf{X}^p$ is updated. Intuitively, this would not lead to an equilibrium but rather a never-ending cycle where one player constantly tries to outmaneuver the other.
4) Instead, the attacker's update should anticipate the opponent's reaction and the impact the opponent's poison has on the target RecSys. This is because, as the first mover, the attacker indirectly influences the opponent's poisoning decisions by altering the context in which the opponent plans their poisoning actions. Specifically, the opponent attacks a target RecSys that the attacker has already poisoned. Therefore, we design the attacker's update rule using the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ (with η^p as step-size).

$$\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \frac{d\mathcal{L}^p}{d\mathbf{X}^p}. \tag{8}$$

This is because, by the chain rule, the total derivative can be broken down to two terms as follows.

$$\frac{d\mathcal{L}^p}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} + \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}, \tag{9}$$

where the first partial derivative term (i.e., $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$) considers how $\mathbf{X}^p$ directly affects $\mathcal{L}^p$ (ignoring $\mathbf{X}^q$), and the second term shows how the attacker's poison $\mathbf{X}^p$ influences the opponent's poison $\mathbf{X}^q$ (i.e., $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$) and how the latter in turn affects the attacker's objective $\mathcal{L}^p$ (i.e., $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}$).
5) To formulate a calculable formula, we further break down $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ according to the optimal opponent reaction. As illustrated in (Fig. 7, left), an optimality creates a correspondence between $\mathbf{X}^p$ and the optimal $\mathbf{X}^{q*}$. Accordingly, after each update on the attacker's vector $\mathbf{X}^p$, the opponents vector $\mathbf{X}^q$ is updated iteratively until it converges to the optimal solution $\mathbf{X}^{q*}$ according to the opponent's objective $\mathcal{L}^q$, i.e., $\partial \mathcal{L}^q / \partial \mathbf{X}^q = 0$. With this connection, we derive the total derivative used in the attacker's update rule. By the chain rule, we employ

$$\frac{d}{d\mathbf{X}^p}\left(\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}\right) = \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} + \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} = 0, \tag{10}$$

to obtain $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} = -\left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\right)^{-1}\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$. By applying $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ to the total derivative formula (9) above, we find

$$\frac{d\mathcal{L}^p}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\right)^{-1}\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}. \tag{11}$$

With (11), we present the practical computation steps as follows. In (11), first-order partial derivatives are gradient vectors (for the scalar function objective $\mathcal{L}$s) that the standard backpropagation may acquire. On the other hand, second-order partial derivatives are the Jacobian matrix that represents how an importance vector $\mathbf{X}$ may influence a gradient vector. For example, $\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} \equiv \frac{\partial}{\partial \mathbf{X}^p}\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ represents how the gradient step for the opponent $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$, is influenced by the attacker's poisoning vector $\mathbf{X}^p$. With the gradient vector $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}$ in the second term, we compute the *vector-Jacobian product* (vjp) with automatic differentiation packages [3]. Specifically, vjp estimations are executed within the conjugate gradient method [4] for solving $\xi\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}} = \frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$, where the vjp computations $\xi\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}$ are processed for each intermediate solution of ξ. After obtaining the solution, which is $\xi = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\right)^{-1}$, a final vjp computation of $\xi\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$ yields the second term of (11)
6) However, requiring the opponent to be updated until convergence after every attacker update is computationally expensive. Therefore, we follow the theoretical results of Fiez et al. [5], and change the procedure by updating both $\mathbf{X}^p$ and $\mathbf{X}^q$ *simultaneously* on the condition that *the step size for the attacker is smaller than that of the attacker* $\eta^p < \eta^q$. As shown in (Fig. 7, right), an intuitive explanation is as follows. Suppose the original procedure to update the opponent until convergence results in a pair of *optimal trajectory* where the opponent's vector is always at an optimal point. In the new procedure, the opponent update rule still nudges the opponent's vector toward the optimal at each stage. Nevertheless, the opponent's vector will slightly deviate from its optimal points in the relaxed setting because the attacker's vector update results in the landscape shifting in the opponent's loss. Thus, the step-size difference

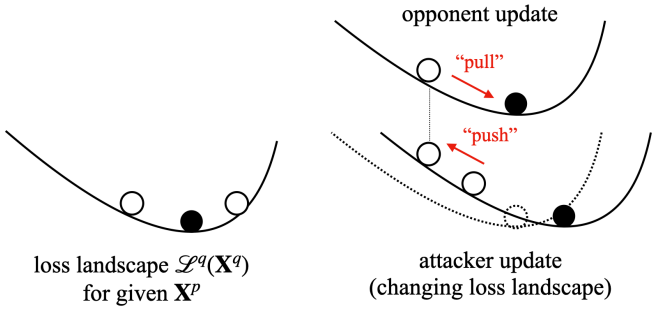


Fig. 7: Left: given the attacker's poison $\mathbf{X}^p$, the opponent's loss landscape $\mathcal{L}^q(\mathbf{X}^q)$ is fixed. Intuitively, there is a corresponding optimal opponent poison (solid circle) but arbitrarily many suboptimal points (outline circle). Right: An illustration of "push" vs. "pull". The opponent's update step takes it closer to the optimal, resembling a "pull" to the optimal position. However, the attacker's update step may shift the loss landscape such that the corresponding opponent's position is further away from optimal, resembling a "push".

is the key ingredient that guarantees the "pull" of the opponent's update step to overcome the "push" of the attacker's update, such that the new trajectory of $\mathbf{X}^q$ still asymptotically converges to the original *optimal trajectory*.

Component 2: Progressive Differentiable Surrogate.

In Section IV-B, we revise our paper to provide an intuitive explanation for the PDS. In particular, following GNN-based RecSys [6], [7], PDS exploits the embeddings of a user $\boldsymbol{h}_u$ and item $\boldsymbol{h}_i$ for rating prediction between the user u and the item i. It first obtains the final user $\boldsymbol{h}_u^f$ and item $\boldsymbol{h}_u^f$ embeddings through graph convolution on the social network and item graph, respectively. Then, the dot product between the final user and item embeddings is provided as the rating prediction.

PDS incorporates the importance vector into the training process of a surrogate Het-RecSys. In particular, the binarized element values, i.e., the decision to select an action or not, are utilized in the graph convolution process to transform the initial embedding into the final embedding.

$$\begin{aligned}\boldsymbol{h}_u^f &= \mathbf{W}_U^\top \left(\boldsymbol{h}_u \oplus \sum_{n \in \mathcal{N}_U(u)} \mathbb{1}_{\mathcal{C}} \hat{x}_U(u,n) \frac{\boldsymbol{h}_n}{|\mathcal{N}_U(u)|} \right) \\ \boldsymbol{h}_i^f &= \mathbf{W}_I^\top \left(\boldsymbol{h}_i \oplus \sum_{m \in \mathcal{N}_I(i)} \mathbb{1}_{\mathcal{C}} \hat{x}_I(i,m) \frac{\boldsymbol{h}_m}{|\mathcal{N}_I(i)|} \right),\end{aligned} \tag{12}$$

where $\mathbf{W}_U$ and $\mathbf{W}_I$ are trainable projection matrices, $\oplus$ denotes concatenation, $\mathcal{N}_U(v)$ and $\mathcal{N}_I(i)$ represent the first-hop neighbors of u and i in $\mathcal{G}_U$ and $\mathcal{G}_I$, respectively. $\hat{x}_U(u,n)$ and $\hat{x}_I(i,m)$ are element values of $\hat{\mathbf{X}}$ that corresponds to the poisoning edge between u and n in the social network and between i and m in the item graph, respectively, whereas $\mathbb{1}_{\mathcal{C}}$ is a selection function that defaults to one, but adopts element value of the binarized importance vector ($\hat{x}_U(u,n)$ or $\hat{x}_I(i,m)$) if the link ((u,n) or (i,m) is in the candidate set $\mathcal{C}$. In other words, the selection function applies the selection decision presented in the binarized element values into the graph convolution process and corresponds to whether the surrogate model perceives that the corresponding edge exists.

On the other hand, poisoning actions on the rating records affect the training loss.

$$\begin{aligned}\mathcal{L}_{train} = &\sum_{(u,i,r) \in \mathbf{R}} \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - r \right)^2 \\ &+ \sum_{(u,i) \in \mathcal{C}_R} \hat{x}_R(u,i) \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - \hat{r} \right)^2,\end{aligned} \tag{13}$$

where $\mathbf{R}$ is the real rating records, $\mathcal{C}_R$ is the set of candidate poisoning actions that include adding item ratings (with real or fake users), $\hat{x}_R(u,i)$ are element values of $\hat{\mathbf{X}}$ that correspond to the poisoning action of adding a rating $r_{u,i}$ with user u on item i, and $\hat{r}$ is a preset target rating value.

Next, we present Fig. 8 to illustrate the detailed mechanisms of the above equations step by step.

1) The importance vector is binarized according to budget constraints to find the current state of the poisoning plan. Here, Fig. 8 assumes a budget of selecting 1 from each type of poisoning actions. For example, the element values corresponding to the two poisoning actions on the rating record are 0.3 and 0.1. The result after binarizing is thus 1 and 0, corresponding to the plan of selecting the first but not the second option.
2) As described by (12), binarized element values 1 and 0 for the two candidate poisoning action on each of the social network and item graph are utilized in the graph convolution process. In Fig. 8, the dotted red lines in the social network and item graph corresponds to a binarized value of 0, while the solid red lines corresponds to 1.
3) On the other hand, the binarized element values corresponding to poisoning action on the rating records are applied to each candidate poisoning terms in the training loss (13). For instance, in Fig. 8, the dotted red curve represents a candidate poisoning rating with binarized value 0 while the other solid red curve represents 1. Thus, in (13), their $\hat{x}_R$ is $\hat{x}_R = 1$ for the former and $\hat{x}_R = 0$ for the latter case.
4) Note that in the above two steps, all candidate poisoning actions on the graphs and the rating records have already taken place within PDS to allow all elements to appear in the computation graph (even as a 0 value). Thus, all element values participate in the training process and are present in the recorded computation graph.

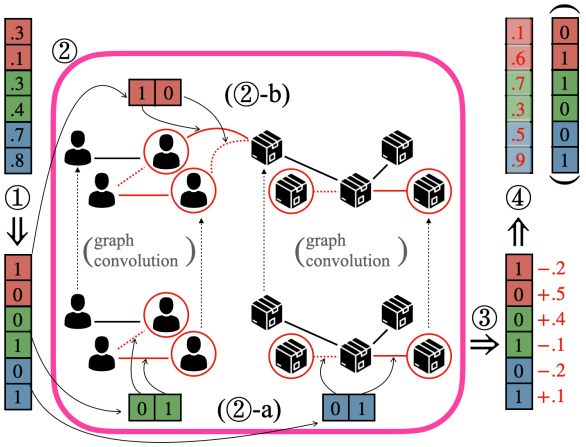


Fig. 8: Illustrative example of PDS. From the top left: ① the importance vector is binarized to select 1 of each type of poisoning action (budget constraint). ② the binary result is incorporated into the RecSys training process: (②-a) poison edges regulate the graph convolution on social network and item graph; (②-b) poison ratings modulate the training loss; ③ backpropagation through the recorded training process (in ②) computes the required derivatives to calculate the "gradient step" according to the update rule (9) or (10);④ finally, the importance vectors are updated by the gradient calculated from the update rule. Comparing the binarized result of before (bottom left) and after (top right) this iteration, we can see that the update results in some of the selections changed.

5) After calculating the poisoning loss objective $\mathcal{L}$ from the poisoned PDS RecSys results $\mathcal{R}$, the gradient can be obtained by backpropagation from $\mathcal{L}$ to $\mathcal{R}$ and eventually to each and every element.
6) The calculated update is applied to the importance vector. Here, Fig. 8 shows that the update results in a different binarized vector in the next iteration since the update changes the priority of each poisoning action.

An overview of the full MSOPDS algorithm.

Finally, in Section IV-D, we present an overview of the full MSOPDS algorithm with an illustrative example in Fig. 9. Corresponding to Fig. 9, a revised explanation is given as follows.

1) The importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ are binarized into $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^p$. Here, we present an example of selecting 1 out of the two candidate actions from three action types represented in red (poison rating), green (poison social network), and blue (poison item graph)
2) The binarized vectors are fed into the Progressive Differentiable Surrogate to simulate the poisoned RecSys result and to evaluate the poisoning objectives of the attacker $\mathcal{L}^p$ and the opponent $\mathcal{L}^q$.
3) We operate Multilevel Stackelberg Optimization to calculate the *total derivative* of $\mathcal{L}^p$ w.r.t. $\mathbf{X}^p$ and the *partial derivative* of $\mathcal{L}^q$ w.r.t. $\mathbf{X}^q$. As shown in the large blue dotted box, the total derivative includes a partial derivative term $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$ and a second term $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ that describes the attacker's *indirect influence* on the opponent.
4) Finally, the updates are applied to the importance vectors by adding the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ to the attacker's $\mathbf{X}^p$ and the partial derivative $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ to the opponent's $\mathbf{X}^q$.

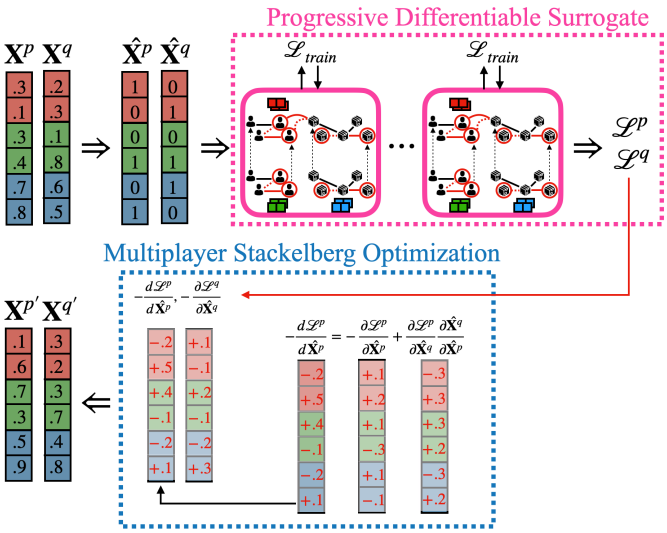


Fig. 9: An illustrative example of one iteration in the full MSOPDS algorithm. We highlight the two components of MSOPDS, namely, Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), by the blue box and the pink box, respectively. Note that for MSO, we also present the high-level ideas in Fig. 5 explaining that the update of the attacker's vector $\mathbf{X}^p$ considers its indirect influence on the opponent's vector $\mathbf{X}^q$, which is represented above as the two terms ($\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$ and $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}\frac{\partial \hat{\mathbf{X}}^q}{\partial \hat{\mathbf{X}}^q}$) adding up to become the total derivation $\frac{d\mathcal{L}^p}{d\hat{\mathbf{X}}^p}$. Furthermore, the high-level intuition for PDS is presented in Fig. 6 explaining that incorporating the vector into the training process and storing the resulting computation graph allows for the computation of the required partial derivative term. Besides, Fig. 8 presents an illustrative example for the detail mechanisms of how the PDS incorporates different sections of the importance vector corresponding to the three types of poisoning actions, i.e., poisoning the rating, poisoning the social network, and poisoning the item graph.

W2 : In Section V: The properties of the proposed solution should be discussed more carefully or formally. In particular, the proof of Theorem 2:
- Contains a strong assumption on the convergence of the recommender model ("Suppose the Het-RecSys model training converges [etc]"). This assumption should be in the conditions of the theorem, if it is really needed. In addition, it should be explained more clearly, as its meaning is currently not well explained.
- Does not contain a formal argument either for the uniqueness of the equilibrium (e.g., what if there are ties among the different options of the attackers, is that impossible?) or the guarantee of convergence (the argument should necessarily make reference to the procedure that is used for the optimization, which is missing from the current argument).

Reply: Thanks for the valuable comments. As suggested, we

revise Section V carefully.

1) We remove the strong assumption on the convergence of the recommender model and clarify the condition used in all theorems and proofs.
2) We revise Theorem 2 with the proof on the existence of equilibrium.
3) We add a new Theorem 3 to present the proof on the convergence of MSOPDS to equilibrium.

In the following, we first introduce the definition of Stackelberg equilibrium [8] and differential Stackelberg equilibrium [5]. Then, we present the theorems on existence of equilibrium and on the guaranteed convergence. Note that the theoretical analysis are connected to MSOPDS in two ways.

1) The update rules utilized in MSOPDS are aligned with the criterion utilized in the definition of differential Stackelberg equilibrium (Definition 7).
2) The learning rate of the attacker is set to be lower than that of the opponent in MSOPDS to meet the requirement for Theorem 3.

Definition 6 (**Stackelberg Equilibrium** [8])**.** *Let $\mathcal{C}^p$ and $\mathcal{C}^q$ be the capacity of the leader and the follower, respectively. The poisoning plan $\mathbf{X}^{p*} \in \mathcal{C}^p$ is a Stackelberg solution for the leader if $\forall \mathbf{X}^p \in \mathcal{C}^p$,*

$$\inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^{p*})} \mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^q) \geq \inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^p)} \mathcal{L}^p(\mathbf{X}^p, \mathbf{X}^q), \quad (17)$$

where $R_{\mathcal{C}^q}(\mathbf{X}^p) = \arg\max_{Y \in \mathcal{C}^q} \mathcal{L}^q(\mathbf{X}^p, Y)$ is the optimal solution set for the follower. More precisely,

$$(\mathbf{X}^{p*}, \mathbf{X}^{q*}), \forall \mathbf{X}^{q*} \in R_{\mathcal{C}^q}(\mathbf{X}^{p*}) \quad (18)$$

are the Stackelberg equilibriums.

Definition 7 (**Differential Stackelberg Equilibrium** [5])**.** *A pair of leader and follower poisons $(\mathbf{X}^{p*}, \mathbf{X}^{q*}) \in (\mathcal{C}^p, \mathcal{C}^q)$ is a Differential Stackelberg Equilibrium if*

$$\frac{d^2}{d{\mathbf{X}^p}^2}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \ \frac{\partial^2}{\partial {\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \quad (19)$$

$$\frac{d}{d\mathbf{X}^p}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = \frac{\partial}{\partial \mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0. \quad (20)$$

Notice that (20) concurs with the update rules of MSO, where the attacker p is updated by the *total derivative* $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ and the opponent q is updated by the *partial derivative* $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$. Intuitively, such update rules approach optimality individually. However, both must be carefully designed to arrive at optimality simultaneously to reach an equilibrium. An important criteria required by Theorem 3 below is that the step-size of the attacker be smaller than that of the opponent $\eta^p < \eta^q$. Thus, the requirement is adopted in MSO to reach an equilibrium.

Existence of equilibrium

We present the proof of the existence of equilibrium based on Definition 6. Moreover, we remove the strong assumption on RecSys model convergence.

Theorem 2. *Stackelberg equilibrium exists for MCA between one attacker (p) and one opponent (q).*

Proof. Following [9], we assume the GNN-based RecSys model has a finite graph G with the initial user $\mathbf{h}_u$ and item $\mathbf{h}_i$ embeddings satisfying the Gaussian distribution, which has been demonstrated to represent uncertainty [10]. According to Theorem 1 in [11], the convergence rate of Het-RecSys training is thereby $O(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}))$ by Polyak-Lojasiewicz inequality [12], namely that the squared norm of the gradient $\nabla\mathcal{L}_{train}$ lower bounds the loss value at any iterate $|\mathcal{L}_{train,t} - \mathcal{L}_{train,t-1}|$. ϵ is the error rate. Therefore, with a set of $\mathbf{X}^p$ and $\mathbf{X}^q$, $\mathcal{L}^p$ and $\mathcal{L}^q$ are deterministic due to the convergence of Het-RecSys. Since the candidate sets $\mathcal{C}^p$ and $\mathcal{C}^q$ are finite sets and every possible strategy pair $(\mathbf{X}^p, \mathbf{X}^q)$ is finite under a given budget, an optimal strategy, i.e., Stackelberg equilibrium, exists over the finite set [13]. □

Guarantee of convergence

We present a proof of convergence based on Definition 7.

Theorem 3. *MSOPDS approaches differential Stackelberg equilibrium in MCA.*

Proof. Consider a differential Stackelberg equilibrium $\mathbf{X}^* \equiv (\mathbf{X}^{p*}, \mathbf{X}^{q*})$ such that $\frac{d\mathcal{L}^p(\mathbf{X}^*)}{d\mathbf{X}^p} = \frac{\partial\mathcal{L}^q(\mathbf{X}^*)}{\partial\mathbf{X}^q} = 0$, $\text{spec}(\frac{d^2\mathcal{L}^p(\mathbf{X}^*)}{d{\mathbf{X}^p}^2}) \subset \mathbb{R}^\circ_+$ and $\text{spec}(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial{\mathbf{X}^q}^2}) \subset \mathbb{R}^\circ_+$. Since $\det(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial{\mathbf{X}^q}^2}) \neq 0$ and $\frac{\partial}{\partial\mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0$, by the implicit function theorem [14], there exists a neighborhood $\mathcal{N}_1$ of $\mathbf{X}^p$ and a unique function $f : \mathcal{N}_1 \to \mathbb{R}^{|\mathcal{C}^q|}$ such that $\frac{\partial}{\partial\mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^p, f(\mathbf{X}^p)) = 0 \ \forall \mathbf{X}^p \in \mathbb{R}^{|\mathcal{C}^p|}$ and $f(\mathbf{X}^{p*})) = \mathbf{X}^{q*}$.[1]

Due to the fact that eigenvalues vary continuously, there exists a neighborhood $\mathcal{N}_2$ of $\mathbf{X}^p$ where $\frac{d^2}{d{\mathbf{X}^p}^2}\mathcal{L}^p(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$ and $\frac{\partial^2}{\partial{\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$. Let $\mathcal{U}_1 \subseteq \mathcal{N}_1 \cap \mathcal{N}_2$ be the non-empty, open set whose closure is contained in $\mathcal{N}_1 \cap \mathcal{N}_2$. Since $\frac{\partial^2}{\partial{\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0$, there exists an open neighborhood $\mathcal{U}_2$ of $\mathbf{X}^{q*}$ such that $\frac{\partial^2}{\partial{\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^p, \mathbf{X}^q) > 0 \ \forall(\mathbf{X}^p, \mathbf{X}^q) \in \mathcal{U}_1 \times \mathcal{U}_2$. Since we have set the step-size of the attacker η^p and the subsequent opponent η^q to be $\eta^p < \eta^q$ in MSOPDS, according to (Lemma G.2 of [5]), the optimization process of MSOPDS on $(\mathbf{X}^p, \mathbf{X}^q)$ converges to $(\mathbf{X}^{p*}, \mathbf{X}^{q*})$ □

W3 : **Overall, even if the paper addresses a more complex technical setting than previous work, I wonder what is new to learn from it. There seems to be novelty from having a different setting, but perhaps not so much in terms of methodological insights, as in the end the approach consists in gradient optimization coupled with binarization (Algorithm 1), which should be considered standard in this setting.**

Reply: Thank you for providing valuable feedback on our paper. Our work addresses a more realistic attack scenario that considers 1) multiple adversaries and 2) a more advanced Heterogeneous RecSys that employs social network and item graph. It is important to consider multiple adversaries due to

[1] Empirically, the numerical values of the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ and the partial derivative $\frac{\partial\mathcal{L}^q}{\partial\mathbf{X}^q}$ are all within reasonable range throughout the MSOPDS iterations, implying that they are L_1 and L_2-Lipschitz, respectively.

the competition among multiple sellers [15] who are incentivized to manipulate the RecSys for financial gains [16]. In addition, real-world eCommerce and review platforms often host diverse information, such as social networks (e.g., Facebook Marketplace) and item connections (e.g., Amazon's "People also bought" feature), which are exploited in a Heterogeneous RecSys to improve the performance of recommendations.

As suggested, we first highlight the motivations to study the more realistic data poisoning attack scenarios, leading to our problem settings which are naturally more complex than previous work due to the considered scenarios. Moreover, we propose several acceleration techniques to further improve our method. Furthermore, we clarify technical contributions and methodological insights of our approach.

Motivations for the realistic yet complex setting.

As suggested, we revise Section I to highlight the motivation to study the realistic yet complex setting of the Multiplayer Comprehensive Attack problem, where multiple competing data poisoning attacks are carried out on a Heterogeneous RecSys (Het-RecSys). To our knowledge, this is not only the first effort to capture multiple competing attacks on a RecSys, but also the first attempt to consider a Het-RecSys, which is more advanced and realistic than the basic RecSys considered in previous work as it exploits user relations in social network as well as item relations in item graph.

We enhance the motivations and provide justification. Specifically, we give examples of 1) multiple adversaries attempting to manipulate RecSys for their financial gains, and 2) RecSys adopting heterogeneous information (including social network and item association graph) for better recommendation results, to reflect the real-world scenarios. We also point out that research exploration in these aspects are missing in the prior works [17]–[20] and thus worth more research effort.

1) **Multiple adversary.** In a real-world setting, especially for eCommerce platforms powered by recommender systems (RecSys), the use of false or dishonest materials (so called poisoning data) to manipulate recommendation results is a commonly found phenomenon that triggers significant research interests and motivates many studies. Attacks have been attempted by various actors, including merchants and vendors. These attacks are so prevalent that some companies even offer services to support them. For instance,
    a) Search engine optimization (SEO) companies offer services to help clients improve the ranking of their sites or products through a mixture of techniques, which may be acceptable or malicious [21].
    b) Malicious companies that offer to create fraudulent reviews also exist. As reported, Amazon filed lawsuits to block such companies from creating fraudulent positive reviews on its platform [22].

    Furthermore, several real-world incidents suggest that malicious attempts to manipulate recommendation systems are not rare. For example,
    a) Amazon removed notable gadget vendors Aukey, Mpow [23], RavPower [24], Vava, TaoTronics [25] and Choetech [26] from its site to protect integrity of its site and to ensure genuine reviews [23].
    b) Beyond the notable brands, Amazon also permanently banned as many as 600 Chinese brands for review frauds [27].
    c) Yelp, facing fraud reviews in its website, also filed lawsuits against a company for allegedly selling fake positive reviews to restaurants [28]. It has been estimated that nearly 16% of the restaurant reviews are fake in Yelp [29].
    d) Countless sellers and vendors attempt to manipulate recommendations with fake reviews. For example, The Washington Post reports that nearly 2% of the top-grossing apps in the Apple App store are frauds [30]. It states that some (low-quality) fake reviews may be done with bots. It also reports how merchants use Facebook to flood Amazon with fake reviews [31].
    e) Some well-known companies were accused of fake review marketing, including Huawei [32], which hired people to write fake reviews for an unreleased phone, Roomster [33], a room renting platform sued by the FTC for paying people for fake reviews and listings on their site, and Sony Pictures [16], [34], which made some specific movies widely recommended by falsifying movie reviews.

    The above real examples indicate the scope and prevalence of the phenomenon of manipulating RecSys, as well as the involvement of multiple actors. Therefore, it is important to examine the interactions among multiple actors attempting data poisoning attacks on the same recommender system. Prior research has also noticed that "many merchants in various systems are well motivated to take advantage of data poisoning attacks to promote their products or demote their competitors' ones" [16]. However, previous research has overlooked the realistic scenario of multiple adversaries, which we explore in this work.
2) **Heterogeneous information.** It is common for online recommendation systems to utilize heterogeneous information, including user social network and item relations, to improve recommendation results. Many online platforms host users' social network on their websites, facilitating their recommender systems to exploit the information [35], [36]. For instance,
    a) Facebook Marketplace[2], operating within the main Facebook app and website, leverages the social information in Facebook.
    b) Yelp[3] also records social links by allowing its users to connect with their friends [37]. In particular, Yelp is reported to be in competition with and

[2]https://www.facebook.com/marketplace
[3]https://www.yelp.com/

under pressure from Facebook with regards to its social recommendation features [38], [39]

c) Pinterest[4] is a social network and image sharing service where people discover and save images. It utilizes machine learning algorithms on its platform to build its recommender system [40].

On the other hand, multiple eCommerce platforms also enhance user experience with relevant recommendations based on item relations. For example,

a) Amazon[5] provides "Other Users Also-bought," "users who bought X also bought Y," and "users who viewed X also viewed Y" categories in a product page. Such information has also been collected as a dataset [41] by the academic community and is actively researched to improve recommender system designs [42]–[44].

b) Google maps[6] presents "User also searched" for alternative location suggestions on each location page. It also provides "search nearby" option for users to explore locations that are connected geologically. Such information is employed in its recommender system [45].

c) Netflix[7] displays "Because You Watched X" to recommend other movie items to users. Netflix's help center also explains that movie items may be recommended based on connections through user preference and tastes or information about the titles [46].

Furthermore, the use of heterogeneous information is important in the eCommerce industry. Specifically, social commerce [47], the combination of eCommerce and social media platforms [48], is a rapidly growing sector of business [49]. According to a recent survey, 87% of eCommerce shoppers believe that social media helps them make shopping decisions, and 40% of merchants use social media to generate sales [50]. As social commerce continues to grow and evolve, it is vital to study and understand how to effectively leverage these heterogeneous information of social connections through RecSys in order to maximize their profits.

To conclude, this paper formulate a realistic problem setting of data poisoning attacks against RecSys, called Multiplayer Comprehensive Attack (MCA). It is not only the first to capture the dynamics among multiple players conducting attacks on the same RecSys, but also the first attempt to consider the more realistic Heterogeneous RecSys. In this work, we propose MSOPDS to solve MCA. In the following, we present new techniques introduced in the revision to improve MSOPDS and then discuss methodological insights in the design of MSOPDS.

New improvements on the proposed MSOPDS.

[4]https://www.pinterest.com/
[5]https://www.amazon.com/
[6]https://maps.google.com/
[7]https://www.netflix.com/

As suggested, we reexamine the proposed method, MSOPDS. We observe that MSOPDS is more computationally intense than standard gradient optimization processes, since it involves multiple vector-Jacobian product calculations. This is because MSOPDS allows the attacker to anticipate subsequent opponent's actions by accounting for the attacker's *indirect influence* on his opponent. In particular, in the attacker's update rule in MSOPDS,

$$\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \left(\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial {\mathbf{X}^q}^2} \right)^{-1} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} \right),$$

the first update term ($\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$) can be efficiently computed with backpropagation, but the second *Stackelberg correction term* ($Q_2 \equiv \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial {\mathbf{X}^q}^2} \right)^{-1} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$) involves multiple vector-Jacobian product calculations. Therefore, we design three acceleration techniques, namely, *Competitor-aware Adaptive Gradient Reuse (reuse)*, *Stochastic Poisoning Action Omission (omis)*, and *Progressive Attack Influence Propagation (prop)*, to further improve MSOPDS. Details of these three techniques are presented below.

1) **Competitor-aware Adaptive Gradient Reuse (reuse).** In MSOPDS, we set a lower learning rate for the attacker's vector compared to the opponent's vector following the idea that the attacker's vector should be updated slower [5]. With this in mind, we update the Stackelberg correction term less frequently and allow it to evolve slower in order to reduce the computational cost. Therefore, inspired by gradient variance reduction techniques [51], we store and reuse the Stackelberg correction term Q_2 and update it every 2 iterations. Note that we observe storing over longer iteration leads to reduced effectiveness.

2) **Stochastic Poisoning Action Omission (omis).** MSOPDS iteratively updates both the attacker's vector $\mathbf{X}^p$ and the opponent's vector $\mathbf{X}^q$ to reach an equilibrium. The iterations are required because the poison objective of a player can be influenced by the other player's decisions. For example, among different poisoning actions of the attacker, some actions may always contribute towards his poison objective regardless of the opponent's decision, while other actions are heavily affected by the opponent's actions. Thus, during the iterative update in MSOPDS, some actions may remain selected (unselected), i.e., the corresponding element value in $\mathbf{X}^p$ remain higher (lower) than the rest, while other elements with values ranging in the center may undergo stochastic changes between selected and unselected from iterations to iterations. Based on the above observation, an idea is to only update the elements of the attacker's vector $\mathbf{X}^p$ with values ranked in the middle half after the initial iterations. Note that we resume full vector optimization in the final iterations. Besides, we do not apply the above procedure to the opponent's vector, because the opponent's responses need to remain optimal to theoretically achieve the

convergence of MSOPDS, and that the computation for updating $\mathbf{X}^q$ is not the bottleneck since it only involves standard backpropagation.

3) **Progressive Attack Influence Propagation (prop).** After reexamine the graph convolution process of Het-RecSys, we further improve the efficiency of MSOPDS by starting from a smaller subset of the heterogeneous graph $\mathcal{G}' \subset \mathcal{G}$, and then progressively update the next-hop neighbors. This is because a poisoning edge first directly affects the embeddings of the two nodes that are connected by it, and then indirectly influence other nodes since their embeddings are updated through the RecSys training process. Therefore, the **prop** method first considers the terminal nodes and their first-hop neighbors of all candidate poisoning edge on the social network $\mathcal{G}_U$ and the item graph $\mathcal{G}_I$. The terminal nodes of the candidate poisoning edges include the customer base $\mathcal{U}_{base} \subset \mathcal{U}$ and the fake users $\mathcal{U}_{fake}$ as well as the company products $\mathcal{I}_{product} \subset \mathcal{I}$ and the target item i_t. We then progressively include the next-hop neighbors on $\mathcal{G}_U$ and $\mathcal{G}_I$ into the MSOPDS process until reaching the full heterogeneous graph.

As shown in Figure 10, these three acceleration techniques provides varied tradeoffs between reaching equilibrium and efficiency for MSOPDS. In particular, as the acceleration techniques achieve suboptimal results but have lower running time, these techniques may be applicable to use cases that are required to operate with less computation resources. We include the acceleration techniques in a new subsection under Section IV (THE MSOPDS METHOD) and the results in a new subsection under Section VI (EXPERIMENTS).

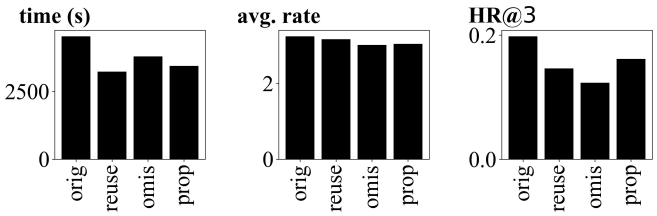


Fig. 10: Bar plot of the total run time (time), average rating prediction (avg. rate) and Hitrate@3 (HR@3) comparing the original method (orig) and the three acceleration methods (reuse, omis, and prop). We find that all three acceleration techniques trades-off reaching equilibrium for efficiency.

Methodological insights of MSOPDS

As suggested, we revise Section I and Section IV to highlight the three key methodological insights from MSOPDS as follows.

1) MSOPDS approaches an equilibrium amongst multiple objectives, instead of following the standard gradient optimization that directly optimizes on a single objective.
2) New update rules in MSOPDS are designed in accordance with a formal analysis on guaranteed convergence to equilibrium based on recent theoretical results.
3) MSOPDS presents Progressive Differentiable Surrogate, a novel GNN-based RecSys design, where poison edges and poison ratings are separately incorporated into the graph convolution process and the training loss, respectively.

Detailed explanations on these three insights are provided as follows.

1) MSOPDS differs from standard gradient optimization with binarization procedure in the data poisoning attack literature [2], [17], [20], owing to the challenging multi-player setting it faces. In the standard gradient optimization adopted by all previous methods [2], [17], [18], [20], [52], the goal is to directly optimize for a single objective $\mathcal{L}$ and update the variable $\mathbf{X}$ by partial derivatives $\frac{\partial \mathcal{L}}{\partial \mathbf{X}}$. In other words, even if the objective may depend on other factors, the main focus is on a single variable and how to attain optimality as evaluated by $\mathcal{L}$. On the other hand, MSOPDS takes into account the effects of multiple players' poisoning actions on the optimization process. Instead of directly updating the variable with the gradient direction, our approach aims to approach an equilibrium to evaluate the best response of the opponents (for their respective objectives) and the best poisoning plan for the attacker's objective. Specifically, we present Multilevel Stackelberg Optimization (MSO), which needs to simultaneously handle multiple sets of data poisoning for the attacker and the subsequent opponents. This is different from the standard bi-level optimization, where only a set of poisoning actions for a single attacker is optimized.
2) The new update rules in MSO are designed in this paper to ensure that both the attacker's and the opponent's plans approach an equilibrium state. In MSOPDS, we carefully model update rules for the attacker to incorporate its influence on the subsequent opponents and derive a correction term in (13) consisting of second-order derivatives (i.e., Jacobian matrices). The more complex update rules in our method play key roles in the formal analysis of convergence guarantees using recent theoretical results [5], which is not explored in standard data poisoning approaches.
3) Lastly, the formulation of the importance vector and the Progressive Differentiable Surrogate are developed specifically to address the challenges arising in the task of targeting Heterogeneous RecSys. The separate incorporation of poison edges into the graph convolution process and poison ratings into the training loss in a GNN-based Progressive Differentiable Surrogate is a novel design specifically proposed for our problem setting. In contrast, prior works target the basic RecSys that only processes the rating records and their methods involve using simple matrix factorization models as surrogates [2], [17], [20].

A0 : I wonder whether ICDE is a good fit for this type of paper. Venues for recommender systems would be obviously far more suitable. And more generic Machine Learning venues would also be more suitable. At least in my understanding of the call for papers, poisoning attacks are not what is usually meant by "data engineering", and poisoning is essentially related to the machine learning model, not to the management or the processing of data from the administrator or user view.

Reply: Thank you for sharing your concerns. The topic of data poisoning attack against recommender systems [19] is a relevant and important issue in previous ICDE [19], VLDB [53] and VLDBJ [54]. As business continues to generate and collect large amounts of data, it becomes increasingly important to have reliable recommender systems to enable efficient and dynamic data access and exploration [55]–[57]. However, data poisoning attacks against these systems can undermine the recommendations, leading to degraded user experience, financial losses, or even larger societal problems. For example, in the case of eCommerce platforms, poisoning attacks that aims to demote a product could result in a reduction of exposure and a decrease in sales [19]. On social media [58] or news platforms [59], such attacks could also contribute to the spread of misinformation or even the promotion of malicious content, which can have serious consequences for society. Thus, it is important to research data poisoning attacks against recommender systems to defend against these risks.

In particular, the connection between this paper and ICDE can be observed in the following aspects.

1) Our work investigates the topic of data poisoning attacks against recommender systems in the context of a multi-player game setting. This topic has also been explored in previous research published in ICDE, including studies on data poisoning attacks [19], [60]–[62], adversarial attacks and robustness [63]–[67], recommender systems (RecSys) [56], [68]–[72], Graph Neural Networks (GNN) [73]–[78], and game theoretical analysis [79]–[82]. In particular, Song et al. [19] (ICDE 2020) and Fan et al. [61] (ICDE 2021) also examine data poisoning attacks against recommender systems. However, they assume a single adversary setting and target basic recommender systems while our work considers the more realistic and challenging multiple adversaries scenarios and targets heterogeneous recommender systems. In addition to ICDE, related works can also be found in other data engineering venues, including in VLDBJ for defending data poisoning attacks [83], in VLDB for RecSys [55], [84], [85] and GNNs [86]–[88]; and in TKDE for data poisoning attacks [89]–[93], adversarial attacks [94]–[99], and GNN-based RecSys [100]–[111].
2) Our research contributes to the ICDE research area "Data Mining and Knowledge Discovery" by providing a novel approach for extracting useful and actionable information about data poisoning attacks against recommender systems in a multi-player game setting. Our proposed approach, MSOPDS, presents new multilevel optimization frameworks with gradient optimization and Graph Neural Network to process the RecSys data and form decisions based on insights that may not be immediately apparent to a human observer. Similar to us, prior works on data poisoning attacks in ICDE [19], [60]–[62] also utilize knowledge discovery processes assisted by machine learning algorithms. In particular, Song et al. [19] and Fan et al. [61] both utilize reinforcement learning algorithm that learns to construct a data poisoning plan targeting a Matrix Factorization-based RecSys [20] under the single adversary assumption. Banerjee et al. [60] propose deep reinforcement learning-based algorithm to generate poison against machine learning-based knowledge graph completion models, while Zhu et al. [62] utilize the bi-level optimization framework to construct data poisoning attack against graph anomaly detection models through gradient optimization.

In summary, the topic of data poisoning attack against RecSys is actively pursued in ICDE, and our work has the potential to advance the research and inspire more contributions from the ICDE community on this topic area. Following the concern, we revise our paper to strengthen the connection between this work and ICDE.

Response to Reviewer #3

O1 : **The paper is difficult to read. Authors need to describe intuitions for various algorithms and choices they make. You can also include examples at various places to take the reader along.**

Reply: Thanks for the valuable feedback on the readability of our paper. As suggested, we have made every effort to revise and improve the manuscript to make it easier to read. In particular, we highlight the motivation of our problem setting and the present the intuition of our proposed method. Specifically, the changes in responding to your comments above are as follows.

1) In INTRODUCTION (Section I), we revise the explanation of our problem setting to include intuitive descriptions and motivation along with better figure illustrations. We also include high-level illustrations of our proposed MSOPDS method to accompany the respective descriptions.
2) In PROBLEM FORMULATION (Section III), we revise the definition of Injection Attack (Definition 3), Comprehensive Attack (Definition 4), and Multiplayer Comprehensive Attack (Definition 5) to include detailed intuitive explanations of the objectives $\mathcal{L}$ and capacity sets $\mathcal{C}$.
3) In THE MSOPDS METHOD (Section IV) we include high-level descriptions on the formulation of the importance vector as well as the two components MSO and PDS. We provide intuitions for MSO and intuitive figure examples for the explanations PDS mechanism to aid understanding.

In the following, we detail the changes made in the revised manuscript. Tables I and II provide a notation table and an abbreviation table for reference.

TABLE I: List of Notations.

symbol	descriptions
u, i	An individual user or an item.
i_t^p	The target item of player p.
$\mathcal{U}, \mathcal{I}$	The (real) user set and item set.
$\mathcal{U}_{fake}^p$	The set of fake users created by player p. $\mathcal{U}_{fake}^p \not\subseteq \mathcal{U}$
$\mathcal{U}_{TA}^p$	The target audience of player p.
$\mathcal{U}_{base}^p$	The customer base of player p.
$\mathcal{I}_{product}^p$	The company products of player p.
$\mathcal{I}_{compete}^p$	The competing items of player p.
Ξ, $\varnothing$	The set of ratings values $\{1, 2, 3, 4, 5\}$ or missing value
$\mathbf{R}$	The rating matrix ($\mathbf{R} \in \Gamma^{\lvert\mathcal{U}\rvert \cdot \lvert\mathcal{I}\rvert}, \Gamma \equiv \Xi \cup \{\varnothing\}$).
$\mathcal{G}_U$, $\mathcal{G}_I$	The social network and item graph.
$\mathcal{G}$	The heterogeneous graph information ($\mathcal{G} \equiv \mathcal{G}_U \cup \mathcal{G}_I$).
$\mathcal{R}(\boldsymbol{\theta}, \mathcal{G})$	The rating predictions of a RecSys with parameter $\boldsymbol{\theta}$.
$\hat{\mathbf{R}}, \hat{\mathcal{G}}$	The poisoned ratings and the poisoned graph data.
$\mathcal{R}_{(u,i)}$	The ratings predictions (between user u and item i).
$\boldsymbol{h}_u, \boldsymbol{h}_i$	The initial embeddings of user u or item i.
$\boldsymbol{h}_u^{(f)}, \boldsymbol{h}_i^{(f)}$	The final embeddings of user u or item i.
$\mathcal{L}_{train}$	RecSys training loss, i.e., the Mean Square Error loss.
$\mathcal{L}^p$	The adversarial loss, i.e., the poisoning objective (of p).
$\mathcal{C}^p$	The capacity set, i.e., candidate poisoning actions (of p).
$\mathcal{X}^p$	The poisoning action plan of player p
($\hat{\mathbf{X}}^p$) $\mathbf{X}^p$	The (binarized) importance vector of player p.

TABLE II: List of Abbreviations.

acronym	full phrase and descriptions
Het-RecSys	Heterogeneous RecSys, a Recommender System that utilizes heterogeneous graph information $\mathcal{G}$. In contrast, basic RecSys only utilizes the rating records $\mathbf{R}$.
MCA	Multiplayer Comprehensive Attack. Our problem setting which concerns how to conduct data poisoning attack against Het-RecSys while anticipating for subsequent opponent poisonings.
CA	Comprehensive Attack. A poisoning attack setting that targets Het-RecSys but assumes single adversary.
IA	Injection Attack. Prior works' poisoning attack setting. Targets basic RecSys and assumes single adversary
MSOPDS	Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate, our proposed method.
MSO	Multilevel Stackelberg Optimization, provides update rules to approach equilibrium for an attacker facing opponents.
PDS	Progressive Differentiable Surrogate, provides surrogate model for gradient computation over RecSys training.
BOPDS	Bi-level Optimization over Progressive Differentiable Surrogate, an ablation of MSOPDS that replaces MSO with a bi-level framework.

Revisions in INTRODUCTION for the problem setting.

Our problem addresses data poisoning attacks against Heterogeneous RecSys in a multiplayer setting. *Data poisoning attack* involves manipulating a machine learning system's training data to achieve a specific (malicious) goal. Attacks against recommender systems (RecSys) are very serious because abusing the recommendations can result in a significant financial loss and customer dissatisfaction, particularly in eCommerce. However, previous works have only considered the case of a single adversary attempting a data poisoning attack and have focused on basic RecSys, instead of the more advanced Heterogeneous RecSys.

In INTRODUCTION, Fig. 1 presents a comparison of the problem setting between our work and prior works, and Fig. 2 illustrates our Multiplayer Comprehensive Attack (MCA) problem.

1) As shown in Fig. 1, our problem setting, MCA, differs from prior works in two ways:
   a) We consider multiple adversaries, while prior works only consider a single adversary.
   b) We target Heterogeneous RecSys, while prior works target basic RecSys.
2) Fig. 2 motivates the study of MCA.
   a) Because RecSys is open (may be poisoned) and public (data available to the public), multiple attackers may add poisoning edges to the system with different objectives. However, as the later-added poisoning edges may interfere with the effort made by the first attacker, it is important to consider the actions of subsequent adversaries from the perspective of the first attacker in this setting.
   b) The difference between a basic RecSys and a more advanced Heterogeneous RecSys (Het-RecSys) is that the Het-RecSys processes three types of information (rating records, social network, and the item

graph), while the basic RecSys only processes rating records. In other words, the factors considered in producing a recommendation for Het-RecSys are different, and it is thus essential to explore the vulnerabilities of additional types of poisoning actions.

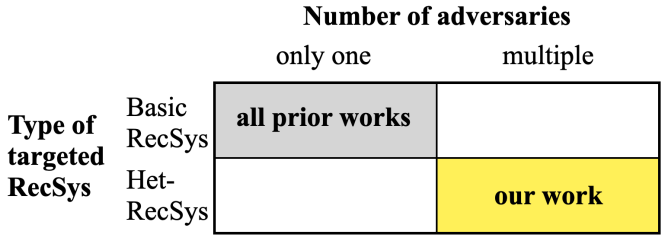


Fig. 1: Comparison of our problem setting with prior works. Our work studies data poisoning attack against Heterogeneous RecSys (Het-RecSys) under a multiple adversary setting.

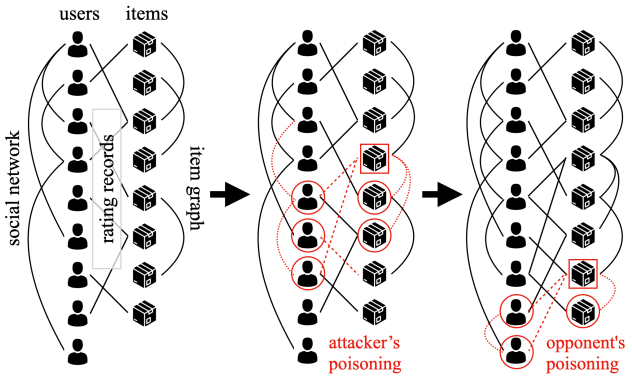


Fig. 2: Illustration of the Multiplayer Comprehensive Attack problem. Red lines indicate poison edges. After an adversary (attacker) poisons the RecSys for his objective, another adversary (opponent) may later poison the RecSys with a different goal. Thus, if the first attacker does not anticipate the *subsequent* opponent's poison actions, his poison effort may be voided by the following opponent's poisons.

Revisions in PROBLEM FORMULATION for the formal definition of MCA.

In PROBLEM FORMULATION, we rewrite the definitions of attack scenarios to streamline our explanation. Formally, we define all prior works as Injection Attack (IA) and our work as Multiplayer Comprehensive Attack (MCA). We also define Comprehensive Attack (CA) which targets Het-RecSys but does not consider the problem setting of multiple adversaries. A comparison of these three attack scenarios is presented in Table III.

In particular, both IA and CA can be solved using the *bi-level optimization framework* while MCA requires *multilevel optimization*. On the other hand, both CA and MCA targets Het-RecSys. Thus, they share the same capacity $\mathcal{C}$ and loss

TABLE III: Comparison of attack scenarios.

	IA	**CA**	**MCA**
targeted RecSys	basic	Het-RecSys	Het-RecSys
# of adversary	1	1	multiple

objective $\mathcal{L}$, which is different from IA. Our new revisions on the formal descriptions are presented as follows.

1) **Definition 2 (Bi-level formulation of data poisoning attack)** presents the optimization framework for the scenarios of *single attacker*, including Injection Attack (IA) and Comprehensive Attack (CA). We revise the manuscript to highlight that IA and CA ascribe to the single player scenario and is formulated by the Bi-level optimization framework by specifying the poisoning objective $\mathcal{L}$ and the capacity $\mathcal{C}$ as follows.

$$\begin{aligned} &\min_{\mathcal{X}\subseteq\mathcal{C}} \mathcal{L}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}), \end{aligned} \tag{1}$$

where the poisoning objective $\mathcal{L}$ is optimized by selecting a poisoning plan $\mathcal{X}$ from the capacity $\mathcal{C}$ in the first layer and the RecSys training is operated based on $\mathcal{L}_{train}$ in the second layer, with $\hat{\mathcal{G}}$ and $\hat{\mathbf{R}}$ denoting graph and rating records poisoned by $\mathcal{X}$. Intuitively, the attacker adjusts their poison data $\mathcal{X}$ (or poisoning action plan) to achieve their poisoning objective. However, to check the effect of the current poisoning, he needs to train the target RecSys with the poisoned data $\hat{\mathcal{G}}$ and rating records $\hat{\mathbf{R}}$. The training objective $\mathcal{L}_{train}$ describes how closely the RecSys predictions $\mathcal{R}$ match the poisoned records $\hat{\mathbf{R}}$, while the attacker's poisoning objective is derived on some aspect of the RecSys predictions results $\mathcal{L}(\mathcal{R})$. For instance, the objective may be to promote a particular item to all users through the recommendations of the RecSys. Finally, the attacker's poisoning plan is selected from a set of feasible actions $\mathcal{C}$ subject to a budget constraint.

2) **Definition 3 (Injection Attack (IA))**. IA subscribes to the above bi-level framework with the Injection Attack loss $\mathcal{L}_{IA}$ and the capacity set $\mathcal{C}_{IA}$. In particular, $\mathcal{L}_{IA}$, which describes the objective of promoting the target item i_t, to all users $u \in \mathcal{U}$, is written as follows.

$$\mathcal{L}_{IA} = -\frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \mathcal{R}_{(u,i_t)}. \tag{2}$$

On the other hand, Injection Attack only makes one type of poisoning action, i.e., manipulating ratings. Its capacity set $\mathcal{C}_{IA}$ can be written as follows.

$$\mathcal{C}_{IA} = \{(u, i, r) \mid u \in \mathcal{U}_{fake},\ i \in \mathcal{I}, r \in \Xi\}, \tag{3}$$

where a set of fake users $\mathcal{U}_{fake}$ is controlled to give fake rating r (selected from a rating set $\Xi \equiv [1, 2, 3, 4, 5]$) to any item $i \in \mathcal{I}$. The capacity set of IA allows the fake users ($\mathcal{U}_{fake}$) to give any fake rating (r out of 5 stars) to

any item ($\mathcal{I}$). The budget constraints limit the number of items that each fake user can rate.

3) **Definition 4 (Comprehensive Attack (CA))**. CA can also be formulated with the bi-level optimization framework. However, to align with realistic scenarios, we design CA with the Comprehensive Attack loss $\mathcal{L}_{CA}$ to focus on promoting the target item i_t to its target audience $\mathcal{U}_{TA}$ over competing products $\mathcal{I}_{compete}$ as follows.

$$\mathcal{L}_{CA} = \frac{1}{|\mathcal{U}_{TA}|} \sum_{u \in \mathcal{U}_{TA}} \sum_{i_c \in \mathcal{I}_{compete}} SELU(\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)}), \tag{4}$$

where the scaled exponential linear units (SELU) [1] are used to emphasize the individual terms where the target item i_t losing to competing items $i_c \in \mathcal{I}_{compete}$. Note that the capacity set of CA, $\mathcal{C}_{CA}$, includes three types of poisoning actions: i) hiring real users from the customer base ($\mathcal{U}base$) to rate the target item (i_t) with a preset rating of 5, ii) hiring users from the customer base to connect to fake accounts in the social network, and iii) selecting real items from a set of company products ($\mathcal{I}_{product}$) to connect to the target item in the item graph. Formally, $\mathcal{C}_{CA}$ is written as follows.

$$\begin{aligned} \mathcal{C}_{CA} = &\{\, (u, i_t, \hat{r}) \mid u \in \mathcal{U}_{base} \,\} \\ &\cup \{\, (u, u_f) \mid u \in \mathcal{U}_{base},\ u_f \in \mathcal{U}_{base} \,\} \\ &\cup \{\, (i, i_t) \mid i \in \mathcal{I}_{product} \,\}. \end{aligned} \tag{5}$$

The budget constraints for $\mathcal{C}_{CA}$ limit the respective number of users and items that can be hired or selected. Overall, the capacity set of IA has one type of poisoning action. In contrast, the capacity set of CA has three types of actions, corresponding to the three types of information used by a Het-RecSys.

4) **Definition 5 (Multiplayer Comprehensive Attack (MCA)).** Finally, in the multiplayer setting, we introduce Multiplayer Comprehensive Attack (MCA). MCA involves an attacker attempting to conduct a data poisoning attack on a Het-RecSys. Still, it faces other adversaries who also attempt poisoning attacks to manipulate Het-RecSys to achieve their respective objectives. As a result, MCA cannot be solved using the basic bi-level optimization framework, as in (1). Instead, we formulate the following multilevel optimization.

$$\begin{aligned} &\min_{\mathcal{X}^p \subseteq \mathcal{C}^p_{CA}} \mathcal{L}^p_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &\mathcal{X}^{q_1} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}^{q_1}_{CA}} \mathcal{L}^{q_1}_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &\qquad \vdots \\ &\mathcal{X}^{q_N} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}^{q_N}_{CA}} \mathcal{L}^{q_N}_{CA}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}) + \lambda \|\theta\|^2, \end{aligned} \tag{6}$$

where the opponents are denoted by $q_1 \dots q_N$. MCA utilizes the same objective $\mathcal{L}_{CA}$ and capacity $\mathcal{C}_{CA}$ as CA. However, MCA differs from CA in that it considers the perspective of an attacker facing multiple opponents (instead of a single attacker facing no other attackers). In Equation (6), the attacker adjusts his poison data, $\mathcal{X}^p$, to achieve his poisoning objective $\mathcal{L}^p$. To anticipate other adversaries' actions, he first simulates the response of each opponent q_i by consecutively optimizing his opponent's poison data $\mathcal{X}^{q_i}$ with his poison data already added to the system, and then trains the targeted RecSys with all players' poison data. Different levels of optimization in (6) represent the hierarchical sequence of actions, with the top level as the attacker selecting his poisoning actions from his capacity set, $\mathcal{X}^p \subseteq \mathcal{C}^p_{CA}$, and the next N levels as the opponents optimizing their respective plans from their capacities, $\mathcal{X}^{q_i} \subseteq \mathcal{C}^{q_i}_{CA}$. The bottom level represents the training of the targeted RecSys with all players' poison data.

The multiple levels in Equation (6) indicate the complexity of the multiplayer setting. From the perspective of the first attacker, p, whose objective is represented in the top layer of (6), we aim to find the optimal poisoning plan, $\mathcal{X}^{p*}$.

Revisions to INTRODUCTION for the proposed method.

In the INTRODUCTION section, the following high-level illustrations for the importance vector (Fig. 3, Fig. 4), Multilevel Stackelberg Optimization (MSO, Fig. 5), and Progressive Differentiable Surrogate (PDS, Fig. 6) are presented together as a full page figure with the respective captions combined. Details explanations to these figures are presented as follows.

To solve the Multiplayer Comprehensive Attack problem, we propose Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate (MSOPDS), which consists of two key components, i.e., Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), with *importance vectors* representing poisoning plans for attackers. In particular, the importance vector is utilized to record the *priority* of each poisoning action. The benefit of using importance vectors is straightforward: it is much easier to perform iterative optimizations on vectors than on sets of discrete actions.

Each element of the importance vector maps to a candidate poisoning action. Furthermore, heterogeneous types of actions can all be concatenated into the importance vector. As shown in Fig. 3, the green elements corresponds to adding poisoning edges in the social network, red elements corresponds to adding poisoning ratings, and blue elements corresponds to adding poisoning edges in the item connection graph.

However, due to the budget constraint, not all poisoning actions can be chosen. Therefore, we utilize a *binarizing step* to select a poisoning plan from the priorities recorded by the importance vector. As shown in Fig. 4, we copy the importance vector into a *binarized* version where the top values of each section (i.e., each action type) are set to 1 and the rest to 0. Then, we can obtain a poisoning plan that abides by the budget constraint by selecting the poisoning actions with 1. Therefore, if we iteratively optimize the value for each vector

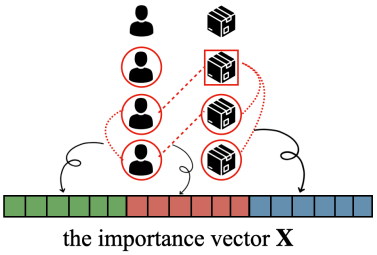


Fig. 3: Illustration of the importance vector, which collectively represents all candidate poisoning actions where each vector element corresponds to one candidate action.

element on the importance vector $\mathbf{X}$ we can effectively adjusts and optimizes the poisoning action plan $\mathcal{X}$.

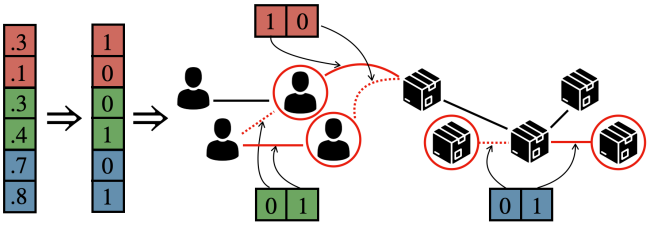
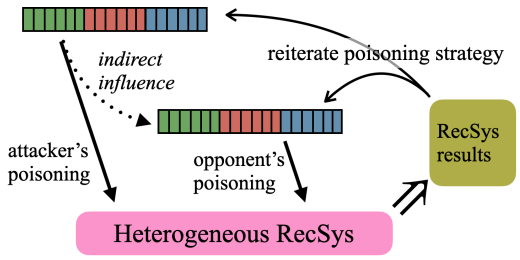


Fig. 4: Example of *binarizing* an importance vector $\mathbf{X}$. The binarizing step is the process we use to select a poisoning plan with the priorities recorded by the importance vector $\mathbf{X}$. The portrayed example shows the selection of a poisoning plan, including poisoning actions on the rating (dark red), the social network (green) and the item graph (blue). With a budget of selecting one out of two options for each of the three types of poisoning actions, the red solid lines indicate the selected actions while red dash lines indicate the candidate actions that are not selected.

Next, as shown in Fig. 5, MSO aims to iteratively approach the attacker's optimal solution (i.e., the optimal importance vector corresponding to an optimal poisoning plan) by *simultaneously* modeling both his and his opponent's actions to carefully examine the interactions between both players. In our problem setting, the attacker conducts his poisoning actions first, and then the opponent plans the subsequent poisoning action based upon the attacker's poison. Such a scenario forms a Stackelberg game. Thus, MSO aims to approach a Stackelberg equilibrium. By definition, under a game equilibrium, no player can change his plan to obtain a better gain for his objective. Intuitively, an equilibrium indicates that both players have obtained the optimal result and thus both follow the optimal poisoning plan.

As detailed in the following section, MSO designs update rules composed of the partial derivatives of the poisoning objectives w.r.t. the importance vectors. As shown in Fig. 6, PDS finds the partial derivatives by incorporating the importance vectors into the training process of a Het-RecSys. This is because the training process determines the RecSys



Multilevel Stackelberg Optimization (MSO)

Fig. 5: Illustration of MSO. In each iteration, the opponent's vector is updated based on how the poison directly influences the opponent's objective. Note that attacker's vector also considers his *indirect influence* on the opponent since the opponent will observe the attacker's poison when deciding his poisoning plan.

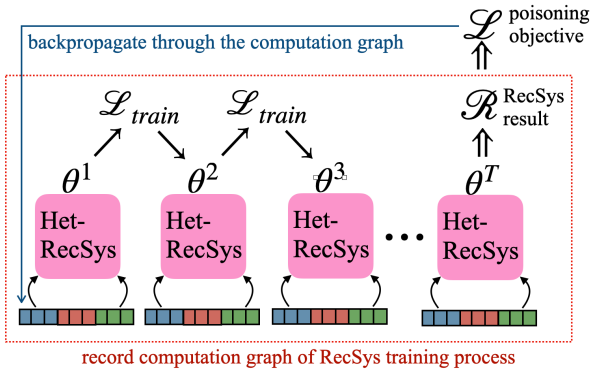


Fig. 6: Illustration of PDS. The PDS calculates the partial derivatives of the poisoning objectives w.r.t. importance vectors by incorporating it into the training process of a Het-RecSys. By recording the computation graph of the training process, the gradient is derived by backpropagation.

result $\mathcal{R}$, which in turn decides the poisoning objective as $\mathcal{L}(\mathcal{R})$. Specifically, to understand the process of training a RecSys and the impact of poisoning on its results, we record the computation graph of the training process. Then, by establishing how each element of the importance vector contributing to the training (poisoning) process of RecSys and to its final poisoned result $\mathcal{R}$, we can also understand how the importance vector affects the poisoning objective $\mathcal{L}$ via backpropagation. Intuitively, we consider the training process of the RecSys as a series of transitions from one iteration to the next, where the parameter obtained in each training iteration is represented by $\theta^1, \theta^2, \theta^3 \cdots \theta^T$ in Fig. 6 and the training loss denoted as $\mathcal{L}_{train}$. The RecSys result is a function of the final iteration parameter $\mathcal{R}(\theta^T)$. The poisoning objective, represented by $\mathcal{L}$, is a function of the RecSys result. Thus, by the chain rule, backpropagation through the entire process may yield the partial derivatives of the poisoning objectives w.r.t. the importance vectors.

Revisions in THE MSOPDS METHOD for the proposed method.

In THE MSOPDS METHOD (Section IV) we reorganize the subsections to introduce the importance vector in Section IV-A, MSO in Section IV-B, PDS in Section IV-C, and the full algorithm in Section IV-D. In each subsection, we present intuitions with figures and examples. Note that MSO and PDS are two components of MSOPDS, while the importance vector is used between them throughout our proposed method.

The Importance Vector.

In Section IV-A, we present the importance vector as follows. To iteratively adjusts intermediate plans by appropriate update rules, we formulate the *importance vector* $\mathbf{X} \in \mathbb{R}^{|\mathcal{C}|}$ where each element x indicates the *priority* of the corresponding candidate poisoning action in $\mathcal{C}$. Given the budget constraints for each type of poisoning actions, the importance vector $\mathbf{X}$ maps to a poisoning plan $\mathcal{X}$ consisting of the actions corresponding to the top-valued elements in $\mathbf{X}$.

Example. Following Fig. 4, suppose an attacker has the budget to hire 1 (out of 2) user to rate a product, 1 (out of 2) user to connect to another user account, and to form 1 (out of 2) item graph connections. His importance vector $\mathbf{X}$ would have 6 elements while his poisoning plan would consists of 3 poisoning actions including 1 poisoning rating, 1 poisoning edge on the social network, and 1 poisoning edge on the item graph. As shown in Fig. 4, the top elements with the top values are selected for each type of poisoning action.

Component 1: Multilevel Stackelberg Optimization.

In Section IV-B, we revise our explanation to clearly present the "update rules" and the intuition behind our design based on the following observation for an attacker facing each opponent.

1) MSO approaches the equilibrium between the attacker and the opponent by designing appropriate update rules for each player. Both players' poisoning objectives ($\mathcal{L}^p$ and $\mathcal{L}^q$) depend on the final RecSys trained with data poisoned by both players and are thus expressed as $\mathcal{L}^p(\mathbf{X}^p, \mathbf{X}^q)$ and $\mathcal{L}^q(\mathbf{X}^p, \mathbf{X}^q)$.
2) If the attacker's vector $\mathbf{X}^p$ is given, the opponent's vector $\mathbf{X}^q$ can be optimized using the standard gradient update [2]:
$$\mathbf{X}^q \leftarrow \mathbf{X}^q - \eta^q \frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}, \tag{7}$$
where η^q is the step size and $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ is the partial derivative of $\mathcal{L}^q$ with respect to $\mathbf{X}^q$. By using the partial derivative, the gradient update direction only considers the effect of the opponent's vector $\mathbf{X}^q$ and not $\mathbf{X}^p$.
3) After finding the optimal $\mathbf{X}^q$, a naive approach is to fix $\mathbf{X}^q$ and update $\mathbf{X}^p$ using the same approach. However, this naive approach does not take into account that the current $\mathbf{X}^q$ was solved according to the given $\mathbf{X}^p$. As a result, it would not be optimal for the opponent after $\mathbf{X}^p$ is updated. Intuitively, this would not lead to an equilibrium but rather a never-ending cycle where one player constantly tries to outmaneuver the other.
4) Instead, the attacker's update should anticipate the opponent's reaction and the impact the opponent's poison has on the target RecSys. This is because, as the first mover, the attacker indirectly influences the opponent's poisoning decisions by altering the context in which the opponent plans their poisoning actions. Specifically, the opponent attacks a target RecSys that the attacker has already poisoned. Therefore, we design the attacker's update rule using the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ (with η^p as step-size).
$$\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \frac{d\mathcal{L}^p}{d\mathbf{X}^p}. \tag{8}$$
This is because, by the chain rule, the total derivative can be broken down to two terms as follows.
$$\frac{d\mathcal{L}^p}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} + \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}, \tag{9}$$
where the first partial derivative term (i.e., $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$) considers how $\mathbf{X}^p$ directly affects $\mathcal{L}^p$ (ignoring $\mathbf{X}^q$), and the second term shows how the attacker's poison $\mathbf{X}^p$ influences the opponent's poison $\mathbf{X}^q$ (i.e., $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$) and how the latter in turn affects the attacker's objective $\mathcal{L}^p$ (i.e., $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}$).
5) To formulate a calculable formula, we further break down $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ according to the optimal opponent reaction. As illustrated in (Fig. 7, left), an optimality creates a correspondence between $\mathbf{X}^p$ and the optimal $\mathbf{X}^{q*}$. Accordingly, after each update on the attacker's vector $\mathbf{X}^p$, the opponents vector $\mathbf{X}^q$ is updated iteratively until it converges to the optimal solution $\mathbf{X}^{q*}$ according to the opponent's objective $\mathcal{L}^q$, i.e., $\partial \mathcal{L}^q / \partial \mathbf{X}^q = 0$. With this connection, we derive the total derivative used in the attacker's update rule. By the chain rule, we employ
$$\frac{d}{d\mathbf{X}^p}\left(\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}\right) = \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} + \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} = 0, \tag{10}$$
to obtain $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} = -\left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\right)^{-1}\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$. By applying $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ to the total derivative formula (9) above, we find
$$\frac{d\mathcal{L}^p}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}\right)^{-1}\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}. \tag{11}$$
With (11), we present the practical computation steps as follows. In (11), first-order partial derivatives are gradient vectors (for the scalar function objective $\mathcal{L}$s) that the standard backpropagation may acquire. On the other hand, second-order partial derivatives are the Jacobian matrix that represents how an importance vector $\mathbf{X}$ may influence a gradient vector. For example, $\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} \equiv \frac{\partial}{\partial \mathbf{X}^p}\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ represents how the gradient step for the opponent $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$, is influenced by the attacker's poisoning vector $\mathbf{X}^p$. With the gradient vector $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}$ in the second term, we compute the *vector-Jacobian product* (vjp) with automatic differentiation packages [3]. Specifically, vjp estimations are executed within the conjugate gradient method [4] for solving $\xi \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}} = \frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$, where

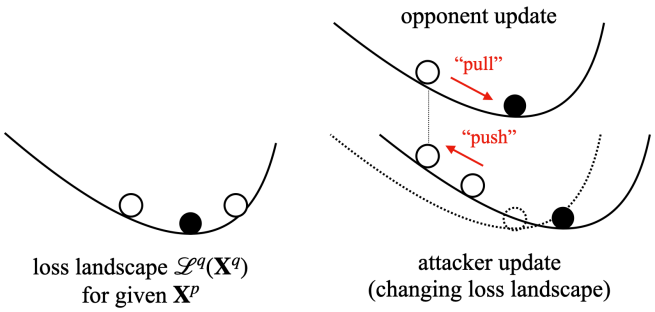


Fig. 7: Left: given the attacker's poison $\mathbf{X}^p$, the opponent's loss landscape $\mathcal{L}^q(\mathbf{X}^q)$ is fixed. Intuitively, there is a corresponding optimal opponent poison (solid circle) but arbitrarily many suboptimal points (outline circle). Right: An illustration of "push" vs. "pull". The opponent's update step takes it closer to the optimal, resembling a "pull" to the optimal position. However, the attacker's update step may shift the loss landscape such that the corresponding opponent's position is further away from optimal, resembling a "push".

the vjp computations $\xi \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}}$ are processed for each intermediate solution of ξ. After obtaining the solution, which is $\xi = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q2}} \right)^{-1}$, a final vjp computation of $\xi \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$ yields the second term of (11)

6) However, requiring the opponent to be updated until convergence after every attacker update is computationally expensive. Therefore, we follow the theoretical results of Fiez et al. [5], and change the procedure by updating both $\mathbf{X}^p$ and $\mathbf{X}^q$ *simultaneously* on the condition that *the step size for the attacker is smaller than that of the attacker* $\eta^p < \eta^q$. As shown in (Fig. 7, right), an intuitive explanation is as follows. Suppose the original procedure to update the opponent until convergence results in a pair of *optimal trajectory* where the opponent's vector is always at an optimal point. In the new procedure, the opponent update rule still nudges the opponent's vector toward the optimal at each stage. Nevertheless, the opponent's vector will slightly deviate from its optimal points in the relaxed setting because the attacker's vector update results in the landscape shifting in the opponent's loss. Thus, the step-size difference is the key ingredient that guarantees the "pull" of the opponent's update step to overcome the "push" of the attacker's update, such that the new trajectory of $\mathbf{X}^q$ still asymptotically converges to the original *optimal trajectory*.

Component 2: Progressive Differentiable Surrogate.

In Section IV-B, we revise our paper to provide an intuitive explanation for the PDS. In particular, following GNN-based RecSys [6], [7], PDS exploits the embeddings of a user $\boldsymbol{h}_u$ and item $\boldsymbol{h}_i$ for rating prediction between the user u and the item i. It first obtains the final user $\boldsymbol{h}_u^f$ and item $\boldsymbol{h}_u^f$ embeddings through graph convolution on the social network and item graph, respectively. Then, the dot product between the final user and item embeddings is provided as the rating prediction.

PDS incorporates the importance vector into the training process of a surrogate Het-RecSys. In particular, the binarized element values, i.e., the decision to select an action or not, are utilized in the graph convolution process to transform the initial embedding into the final embedding.

$$\begin{aligned} \boldsymbol{h}_u^f &= \mathbf{W}_U^\top \left(\boldsymbol{h}_u \oplus \sum_{n \in \mathcal{N}_U(u)} \mathbb{1}_{\mathcal{C}} \hat{x}_U(u,n) \frac{\boldsymbol{h}_n}{|\mathcal{N}_U(u)|} \right) \\ \boldsymbol{h}_i^f &= \mathbf{W}_I^\top \left(\boldsymbol{h}_i \oplus \sum_{m \in \mathcal{N}_I(i)} \mathbb{1}_{\mathcal{C}} \hat{x}_I(i,m) \frac{\boldsymbol{h}_m}{|\mathcal{N}_I(i)|} \right), \end{aligned} \tag{12}$$

where $\mathbf{W}_U$ and $\mathbf{W}_I$ are trainable projection matrices, $\oplus$ denotes concatenation, $\mathcal{N}_U(v)$ and $\mathcal{N}_I(i)$ represent the first-hop neighbors of u and i in $\mathcal{G}_U$ and $\mathcal{G}_I$, respectively. $\hat{x}_U(u,n)$ and $\hat{x}_I(i,m)$ are element values of $\hat{\mathbf{X}}$ that corresponds to the poisoning edge between u and n in the social network and between i and m in the item graph, respectively, whereas $\mathbb{1}_{\mathcal{C}}$ is a selection function that defaults to one, but adopts element value of the binarized importance vector ($\hat{x}_U(u,n)$ or $\hat{x}_I(i,m)$) if the link ((u,n) or (i,m) is in the candidate set $\mathcal{C}$. In other words, the selection function applies the selection decision presented in the binarized element values into the graph convolution process and corresponds to whether the surrogate model perceives that the corresponding edge exists.

On the other hand, poisoning actions on the rating records affect the training loss.

$$\begin{aligned} \mathcal{L}_{train} = &\sum_{(u,i,r) \in \mathbf{R}} \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - r \right)^2 \\ &+ \sum_{(u,i) \in \mathcal{C}_R} \hat{x}_R(u,i) \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - \hat{r} \right)^2, \end{aligned} \tag{13}$$

where $\mathbf{R}$ is the real rating records, $\mathcal{C}_R$ is the set of candidate poisoning actions that include adding item ratings (with real or fake users), $\hat{x}_R(u,i)$ are element values of $\hat{\mathbf{X}}$ that correspond to the poisoning action of adding a rating $r_{u,i}$ with user u on item i, and $\hat{r}$ is a preset target rating value.

Next, we present Fig. 8 to illustrate the detailed mechanisms of the above equations step by step.

1) The importance vector is binarized according to budget constraints to find the current state of the poisoning plan. Here, Fig. 8 assumes a budget of selecting 1 from each type of poisoning actions. For example, the element values corresponding to the two poisoning actions on the rating record are 0.3 and 0.1. The result after binarizing is thus 1 and 0, corresponding to the plan of selecting the first but not the second option.
2) As described by (12), binarized element values 1 and 0 for the two candidate poisoning action on each of the social network and item graph are utilized in the graph convolution process. In Fig. 8, the dotted red lines in the social network and item graph corresponds to a binarized value of 0, while the solid red lines corresponds to 1.

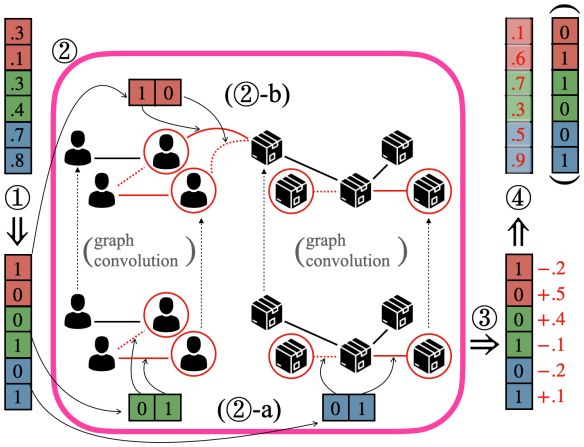


Fig. 8: Illustrative example of PDS. From the top left: ① the importance vector is binarized to select 1 of each type of poisoning action (budget constraint). ② the binary result is incorporated into the RecSys training process: (②-a) poison edges regulate the graph convolution on social network and item graph; (②-b) poison ratings modulate the training loss; ③ backpropagation through the recorded training process (in ②) computes the required derivatives to calculate the "gradient step" according to the update rule (9) or (10);④ finally, the importance vectors are updated by the gradient calculated from the update rule. Comparing the binarized result of before (bottom left) and after (top right) this iteration, we can see that the update results in some of the selections changed.

3) On the other hand, the binarized element values corresponding to poisoning action on the rating records are applied to each candidate poisoning terms in the training loss (13). For instance, in Fig. 8, the dotted red curve represents a candidate poisoning rating with binarized value 0 while the other solid red curve represents 1. Thus, in (13), their $\hat{x}_R$ is $\hat{x}_R = 1$ for the former and $\hat{x}_R = 0$ for the latter case.
4) Note that in the above two steps, all candidate poisoning actions on the graphs and the rating records have already taken place within PDS to allow all elements to appear in the computation graph (even as a 0 value). Thus, all element values participate in the training process and are present in the recorded computation graph.
5) After calculating the poisoning loss objective $\mathcal{L}$ from the poisoned PDS RecSys results $\mathcal{R}$, the gradient can be obtained by backpropagation from $\mathcal{L}$ to $\mathcal{R}$ and eventually to each and every element.
6) The calculated update is applied to the importance vector. Here, Fig. 8 shows that the update results in a different binarized vector in the next iteration since the update changes the priority of each poisoning action.

An overview of the full MSOPDS algorithm.

Finally, in Section IV-D, we present an overview of the full MSOPDS algorithm with an illustrative example in Fig. 9. Corresponding to Fig. 9, a revised explanation is given as follows.

1) The importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ are binarized into $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^p$. Here, we present an example of selecting 1 out of the two candidate actions from three action types represented in red (poison rating), green (poison social network), and blue (poison item graph)
2) The binarized vectors are fed into the Progressive Differentiable Surrogate to simulate the poisoned RecSys result and to evaluate the poisoning objectives of the attacker $\mathcal{L}^p$ and the opponent $\mathcal{L}^q$.
3) We operate Multilevel Stackelberg Optimization to calculate the *total derivative* of $\mathcal{L}^p$ w.r.t. $\mathbf{X}^p$ and the *partial derivative* of $\mathcal{L}^q$ w.r.t. $\mathbf{X}^q$. As shown in the large blue dotted box, the total derivative includes a partial derivative term $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$ and a second term $\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ that describes the attacker's *indirect influence* on the opponent.
4) Finally, the updates are applied to the importance vectors by adding the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ to the attacker's $\mathbf{X}^p$ and the partial derivative $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ to the opponent's $\mathbf{X}^q$.

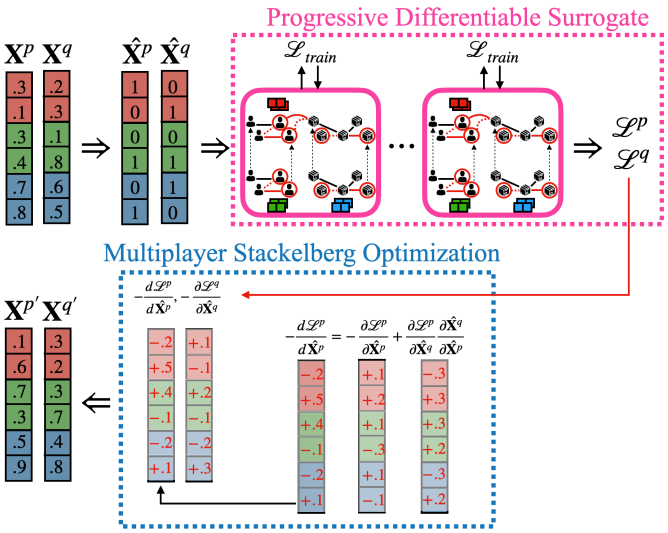


Fig. 9: An illustrative example of one iteration in the full MSOPDS algorithm. We highlight the two components of MSOPDS, namely, Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), by the blue box and the pink box, respectively. Note that for MSO, we also present the high-level ideas in Fig. 5 explaining that the update of the attacker's vector $\mathbf{X}^p$ considers its indirect influence on the opponent's vector $\mathbf{X}^q$, which is represented above as the two terms ($\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$ and $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}\frac{\partial \hat{\mathbf{X}}^q}{\partial \hat{\mathbf{X}}^q}$) adding up to become the total derivation $\frac{d\mathcal{L}^p}{d\hat{\mathbf{X}}^p}$. Furthermore, the high-level intuition for PDS is presented in Fig. 6 explaining that incorporating the vector into the training process and storing the resulting computation graph allows for the computation of the required partial derivative term. Besides, Fig. 8 presents an illustrative example for the detail mechanisms of how the PDS incorporates different sections of the importance vector corresponding to the three types of poisoning actions, i.e., poisoning the rating, poisoning the social network, and poisoning the item graph.

O2 : **Authors have not described the inputs clearly– for example, how the item-item graph is created, how to get the most effective user (#4 in section VIA), how the attacker ensures that real users give bad ratings to the opponent item, etc.**

Reply: Thanks. As suggested, we revise the descriptions and clarify some explanations as follows.

How the item-item graph is created.
We clarify how item-item graph is created in Section VI-A1 as follows.

> Following [7], the item graph $\mathcal{G}_I$ is created by connecting items that share over 50% of users that rated them in the rating record.

How to get the most effective user (#4 in section VIA).
We clarify that the opponent conducts Comprehensive Attack with BOPDS under the bi-level optimization to demote the attacker's target item among the competing item $\mathcal{I}_{compete}$. In particular, we have clarified in Section VI-A4 that the most effective user is selected from the opponent's customer base using BOPDS as follows.

> In particular, each opponent selects real users from his customer base by BOPDS to all give a 1-star rating to the attacker's target item i_t, such that the target item is demoted among the competing items $\mathcal{I}_{compete}$.

How the attacker ensures that real users give bad ratings to the opponent item.
We include more explanations for the behaviors of real users in Section I as follows.

1) Real users can be incentivized by financial gains [112], [113].
2) Real user accounts may also be bought [114], [115] or hacked [116], [117].

On the other hand, following [118], [119], we also extend the PDS training procedures so that each real user follows the attacker or opponent's commands with a compliance probability $\mathbf{p_{comply}}$, where the probability can be learned by Bayesian user modeling [120]. Then, in the poisoning loss objectives (the Injection Attack loss $\mathcal{L}_{IA}$ (3) and the Comprehensive Attack loss $\mathcal{L}_{CA}$ (3)), we add a regulation term $-\sum_{u\in\mathcal{X}} \mathbf{p}^{\mathbf{u}}_{\mathbf{comply}}$ to encourage MSOPDS to select the more compliant users u (with a larger compliance probability $\mathbf{p}^{\mathbf{u}}_{\mathbf{comply}}$) in the poisoning plan $\mathcal{X}$.

In particular, the following footnote is added in Section I.

> With the right incentives, eCommerce customers are often encouraged to leave reviews or ratings [112], [113] or connect with social media accounts [121]. Alternatively, real user accounts may also be purchased [114], [115] or hacked [116], [117], which gives manipulators direct control of those accounts.

In addition, we clarify that the attacker only selects real-users from his customer base $\mathcal{U}_{base}$ to give top ratings (5-star) to his own target item. On the other hand, the opponent gives bad-ratings to the attacker's target item.

Other clarifications on the input hyperparameter settings.
We revise Section VI-A7 to clarify the hyperparameter settings used in Algorithm 1 as follows.

> Corresponding to Algorithm 1, we set $\eta^p = 0.005$, $\eta^q = 0.05$, inner RecSys training loop number $L = 5$, and outer MSO loop number $K = 20$.

Algorithm 1: MSOPDS under the two player setting.

Input: Rating records $\mathbf{R}$, graph records $\mathcal{G}$; Comprehensive Attack losses $\mathcal{L}^p$ and $\mathcal{L}^q$; capacity sets $\mathcal{C}^p$ and $\mathcal{C}^q$ (with budget constraints represented as $\mathcal{B}^p$ and $\mathcal{B}^q$); step sizes η^p and η^q; outer and the inner loop max iteration K and L; PDS parameter θ.
Assert: $0 < \eta^p < \eta^q$
Output: Attacker's action plan $\mathcal{X}^p$.

Create importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ from $\mathcal{C}^p$ and $\mathcal{C}^q$.
Create $\mathbf{R}'$ and $\mathcal{G}'$ by inserting all poisoning actions in $\mathcal{C}^p$ and $\mathcal{C}^q$ into the rating record $\mathbf{R}$ and the heterogeneous graph $\mathcal{G}$;
for $k = 1$ **to** K **do**
  Binarize $\mathbf{X}^p$ and $\mathbf{X}^q$ into $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^q$ by $\mathcal{B}^p$ and $\mathcal{B}^q$.
  for $l = 1$ **to** L **do**
    Update $\theta^{(l)}$ by (1) with the training loss (16) and graph convolution (15) modulated by $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^q$.
    `// store the computation process.`
  Derive the Comprehensive Attack losses $\mathcal{L}^p$ and $\mathcal{L}^q$ for both players with the trained (poisoned) RecSys results $\mathcal{R}(\theta^{(L)}, \mathcal{G}'')$.
  Compute first-order partial derivatives $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$, $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}$, and $\frac{\partial \mathcal{L}^q}{\partial \hat{\mathbf{X}}^q}$ by backpropagation through the stored process.
  `// retain computation graph for second-order vector-Jacobian product.`
  Solve $\boldsymbol{\xi}$ for $\boldsymbol{\xi} \frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^{q\,2}} = \frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}$.
  `// Calculate vector-Jacobian product within the conjugate gradient method.`
  Update attacker $\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \left(\partial \mathcal{L}^p / \partial \hat{\mathbf{X}}^p - \boldsymbol{\xi} \frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^p \partial \hat{\mathbf{X}}^q} \right)$.
  `// Calculate vector-Jacobian product.`
  Update opponent $\mathbf{X}^q \leftarrow \mathbf{X}^q - \eta^q \partial \mathcal{L}^q / \partial \hat{\mathbf{X}}^q$.

O3 : **I didn't find any result in abstract. Its always a good practice to give important result in abstract rather than making a generic statement like "show the effectiveness of proposed method"**

Reply: Thanks for the valuable suggestion. As suggested, we revise the Abstract to clearly provide the quantitative results as follows.

> Experiments on Heterogeneous RecSys trained with public datasets show that MSOPDS outperforms all prior works by up to 10.6% in average predicted ratings and up to 11.4% in HitRate@3 for an item targeted by an attacker facing one opponent.

O4 : **Definition 3 is not clear at all. Mapping better various inputs and Equation 4 can be made more clear.**

Reply: Thanks for the valuable comments. As suggested, we improve our presentation of Definition 4 (i.e., the former Definition 3) for Comprehensive Attack (CA) along with the contained Equation (5) (i.e., the former Equation 4) for the poisoning objective of CA. Because CA is solved by the bi-level framework, we revise Definition 2 for the bi-level formulation of data poisoning attack to clarify the input of Comprehensive Attack and how the input maps to the poisoning objective. In particular, since CA targets Het-RecSys, $\mathcal{C}_{CA}$ defines the *capacity set* of Comprehensive Attack, which is the possible poisoning actions that an attacker may input into the targeted Heterogeneous RecSys. On the other hand, $\mathcal{L}_{CA}$ describes the poisoning objective of the Comprehensive Attack.

1) The inputs of data poisoning attack under the bi-level formulation consists of selecting a poisoning plan $\mathcal{X}$ from a capacity set $\mathcal{C}$. Definition 2 describes the bi-level formulation of data poisoning attack against RecSys that assumes a single attacker. Specifically, the top level of Equation (2) depicts optimizing the objective $\mathcal{L}$ over the poisoning plan selected from the capacity set $\mathcal{X} \subset \mathcal{C}$. Furthermore, the objective is influenced by the RecSys training result $\mathcal{R}$, where the training process of the RecSys parameter θ (to minimize the training loss $\mathcal{L}_{train}$) is given at the bottom level of Equation (2).
2) The inputs of Comprehensive Attack consists of poisoning actions on the training record, social network and item graph. The inputs can be formally described with the capacity set for CA $\mathcal{C}_{CA}$ as.

$$\begin{aligned}\mathcal{C}_{CA} = &\{ (u, i_t, \hat{r}) \mid u \in \mathcal{U}_{base} \} \\ &\cup \{ (u, u_f) \mid u \in \mathcal{U}_{base},\ u_f \in \mathcal{U}_{fake} \} \\ &\cup \{ (i, i_t) \mid i \in \mathcal{I}_{product} \},\end{aligned}$$

where the poisoning ratings $(u, i_t, \hat{r})$ are given by customer base users $u \in \mathcal{U}_{base}$ for the target item i_t using a preset value $\hat{r} = 5$, the poison edge in the social network (u, u_f) are connected between customer base users $u \in \mathcal{U}_{base}$ and fake users $u_f \in \mathcal{U}_{fake}$, and the poison edge in the item graph (i, i_t) are connected between the company products $i \in \mathcal{I}_{product}$ and the target item i_t. Note that not all actions may be taken, but a poison plan $\mathcal{X} \subseteq \mathcal{C}_{CA}$ must be selected from the capacity due to the budget constraint. Specifically, Comprehensive Attack is defined over the bi-level framework by setting the corresponding objective, i.e., the Comprehensive Attack loss $\mathcal{L}_{CA}$, and the capability set $\mathcal{C}_{CA}$. To align with real-world scenarios, we define $\mathcal{L}_{CA}$ to focus on achieving better ratings for the target item i_t over the competing items $i_c \in \mathcal{I}_{compete}$ for a set of target audience users $\mathcal{U}_{TA}$. Furthermore, since CA targets the Het-RecSys, its capacity set $\mathcal{C}_{CA}$ consists of three types of poisoning actions, namely, poisoning the rating records, poisoning the social network, and poisoning the item graph. Specifically, to align with real-world scenarios, we assign a set of customer base users $\mathcal{U}_{base}$ to poison the rating records (i.e., by giving false ratings) and social network (i.e., by connecting with fake users), and a set of company products $\mathcal{I}_{product}$ to poison the item graph by connecting with the target item i_t.

In the following, we present the revised definitions. For reference, Table IV presents the notations.

Definition 2 (**Bi-level formulation of data poisoning attack**). *Data poisoning attack (by single adversary) can be formulated as an bi-level optimization [2], where an adversarial loss (i.e., the poisoning objective) $\mathcal{L}$ describing the attacker's*

TABLE IV: Table of notations.

symbol	descriptions
u, i	An individual user or an item.
i_t	The target item.
$\hat{r} = 5$	The target rating.
$\mathcal{U}, \mathcal{I}$	The set of real users and the set of items.
$\mathcal{U}_{fake}$	The set of fake users for data poisoning.
$\mathcal{U}_{TA}$	The target audience of the attacker.
$\mathcal{U}_{base}$	The customer base of the attacker.
$\mathcal{I}_{product}$	The company products of the attacker.
$\mathcal{I}_{compete}$	The competing items
$\hat{\mathbf{R}}, \hat{\mathcal{G}}$	The poisoned ratings and the poisoned graph data.
$\mathcal{R}(\boldsymbol{\theta}, \mathcal{G})$	The rating predictions of a RecSys with parameter $\boldsymbol{\theta}$.
$\mathcal{R}_{(u,i)}$	Individual ratings predictions (between user u and item i).
$\mathcal{L}_{train}$	RecSys training loss, i.e., the Mean Square Error loss.
$\mathcal{L}_{CA}$	The Comprehensive Attack loss, i.e., the poisoning objective.
$\mathcal{C}_{CA}$	The capacity set of CA, i.e., the candidate poisoning actions.
$\mathcal{X}$	The poisoning action plan of the attacker.

objective is optimized in the upper-level, and the training of RecSys over the poisoned data is described in the lower-level.

$$\begin{aligned} &\min_{\mathcal{X} \subseteq \mathcal{C}} \mathcal{L}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}), \end{aligned} \tag{2}$$

where $\mathcal{X}$ is the poisoning plan selected (under relevant budget constraints) from the attacker's capacity set $\mathcal{C}$, $\mathcal{R}$ is the RecSys results, while $\hat{\mathbf{R}}$ and $\hat{\mathcal{G}}$ represents the poisoned rating records and the poisoned graph information, respectively.

Definition 4 (**Comprehensive Attack (CA)**). *To align with real-world scenarios, we formulate Comprehensive Attack to focus on promoting the target item i_t to its target audience $\mathcal{U}_{TA}$ over competing products $\mathcal{I}_{compete}$ with the* Comprehensive Attack loss *$\mathcal{L}_{CA}$ defined as follows.*

$$\mathcal{L}_{CA} = \frac{1}{|\mathcal{U}_{TA}|} \sum_{u \in \mathcal{U}_{TA}} \sum_{i_c \in \mathcal{I}_{compete}} SELU(\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)}), \tag{5}$$

where the scaled exponential linear units (SELU) [1] are used to emphasize the individual terms with the target item i_t losing to competing items, i.e., $\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)} \geq 0$.

Furthermore, according to common practice, we formulate Comprehensive Attack to leverage a set of customer base *users $\mathcal{U}_{base} \subset \mathcal{U}$ besides the fake users $\mathcal{U}_{fake}$, and a set of* company products *items $\mathcal{I}_{product} \subset \mathcal{I}$ to assist the* comprehensive *poisoning actions on both the rating and the heterogeneous graph information.*

Specifically, the capacity set $\mathcal{C}_{CA}$ consists of hiring users from the customer base *$\mathcal{U}_{base}$ to rate the target item with a preset rating $\hat{r} = 5$, hiring users from the* customer base *to connect to each fake account (on $\mathcal{G}_U$) and selecting items from his* company products *$\mathcal{I}_{product}$ to connect to the target item (on $\mathcal{G}_I$). The budget constraints restricts how many users (items) may be hired (selected).*

$$\begin{aligned} \mathcal{C}_{CA} = &\{ (u, i_t, \hat{r}) \mid u \in \mathcal{U}_{base} \} \\ &\cup \{ (u, u_f) \mid u \in \mathcal{U}_{base},\ u_f \in \mathcal{U}_{fake} \} \\ &\cup \{ (i, i_t) \mid i \in \mathcal{I}_{product} \} \end{aligned} \tag{6}$$

O5 : **Typo in the first line of Section III.**

Reply: Thanks for the valuable comment. We fix the typo in the first line of Section III and revise the manuscript thoroughly to correct others.

Response to Reviewer #4

O1 : **I hope the author could provide more experimental results about time complexity analysis, if possible.**

Reply: Thanks for the valuable suggestion. As suggested, we conduct experiments to verify Theorem 1, which states that the time complexity of our proposed MSOPDS is $O(|\theta| + |\mathbf{X}^p||\mathbf{X}^q|)$. In the time complexity formula, $|\theta|$ represents the number of trainable parameters in PDS, while $|\mathbf{X}^p|$ and $|\mathbf{X}^q|$ represent the sizes of the attacker's and opponent's importance vectors, respectively.

To verify the theorem, we measure the time it takes to perform one iteration of MSOPDS under different values of $|\theta|$, $|\mathbf{X}^p|$, and $|\mathbf{X}^q|$. We vary the size of $|\theta|$ by changing the dimension of the user and item embeddings. In addition, we vary the sizes of $\mathbf{X}^p$ and $\mathbf{X}^q$ by changing the sizes of the customer base and company products. While varying one of $|\theta|$, $|\mathbf{X}^p|$, and $|\mathbf{X}^q|$, we fix the values of the other two variables to small constants to better highlight the effect of the experimented variable on the time complexity of MSOPDS.

As depicted in Fig. 10, despite some fluctuations that may be caused by built-in accelerations in the GPU system, we observe that the time complexity in all three cases generally follows a linear trend, adhering to Theorem 1.

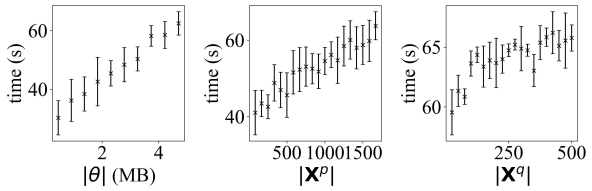


Fig. 1: Running time measurements on $|\theta|$, $|\mathbf{X}^p|$, and $|\mathbf{X}^q|$.

O2 : **Could the author provide more results on other datasets? It can be more convincing.**

Reply: Thanks for the valuable suggestion. Following [122], [123], we conduct additional experiments on the **LibraryThing** dataset [124]. We append the experiment results of LibraryThing dataset in Table IV for the performance comparison of each method facing a single opponent, in Fig. 6 for the impact of the number of opponent, and in Fig. 7 for the impact of the capacity of opponent.

In all three experiments, MSOPDS out-performs the compared baseline methods consistently on the LibraryThing dataset since the baselines do not anticipate opponent's poisoning actions. Moreover, we find that prior observations in experiments on the Epinions and the Ciao datasets are confirmed by the experiments on LibraryThing.

1) **Single opponent evaluation.**
   As shown in Table IV, the proposed MSOPDS under MCA yields the best performance in all budget settings and all three datasets, outperforming baseline methods by a significant margin. In particular, MSOPDS surpasses the second best method by up to 10.6% in terms of the average predicted rating. It also consistently outperforms the second best method in terms of the hit rate by up to 11.4%. We observe that Epinions and LibraryThing present larger difference between MSOPDS and the baselines than Ciao, since Ciao has a sparser social network for the poisoning effect to propagate.
   Apart from MSOPDS, all compared methods have weaker performance since they do not anticipate the poisoning attacks by opponents. Among the baselines, *PGA, S-attack, Revisit Attack*, and *Trial Attack* all plan the data poisoning attack through a bilevel optimization. However, since the above baseline and ablation methods are not designed to face the negative impact of the poisoning attacks by the opponent who acts after the attacker, these approaches suffer from bad performances and may even fall behind heuristic methods (Random Attack and Popular Attack). In contrast, the proposed MSOPDS integrates a Multilevel Stackelberg Optimization framework with Progressive Differentiable Surrogate to anticipate and evaluate the potential effects of hostile actions by the opponent. The potential opponent's actions evaluated within the Stackelberg framework play an essential role in the effective enhancement the robustness of MSOPDS.
2) **Impact by number of opponents.**
   Fig. 6 compares the performance of different methods under various numbers of opponents. The opponents are assumed to sequentially conduct their respective data poisoning attacks using Comprehensive Attack, where each opponent selects real users from their customer base by BOPDS to give a 1-star rating to the target items. While all baselines do not change their attacks as the number of opponents increases (identical to the single-opponent setting in Table IV), MSOPDS carefully plans the optimal poisoning attack by anticipating the opponents' actions via Multilevel Stackelberg Optimization. As a result, the performance of our proposed MSOPDS method under MCA consistently outperforms all compared baseline methods. Although the performance of baseline methods decreases drastically as the number of opponents increases, MSOPDS achieves smaller reductions in both evaluation metrics. In the Epinions dataset, MSOPDS maintains positive HitRate@3 scores, while other methods all drop to 0, manifesting the importance of modeling a multiplayer game for data poisoning attacks against RecSys.
3) **Impact of opponent capacity.**
   Fig. 7 compares the performance of an attacker facing a single opponent with varied budgets. While increasing the opponent's budget *reduces* the attacker's performance, MSOPDS under MCA outperforms all baselines. MSOPDS suffers from a smaller degradation in performance, since it anticipates the opponent's poisoning action plans and adjusts the attacker's poisoning action plan accordingly. Besides, the performances of the attacker's poisoning attacks in Epinions and LibraryThing are also more sensitive to the change of the opponent

TABLE III: Attacker's target item average predicted rating ($\bar{r}$) and Hit Rate@3 (HR@3) on ConsisRec facing a single opponent.

Dataset	Type	Method	$b=2$		$b=3$		$b=4$		$b=5$	
			$\bar{r}$	HR@3	$\bar{r}$	HR@3	$\bar{r}$	HR@3	$\bar{r}$	HR@3
Ciao	IA	None	2.1282	0.0154	2.1282	0.0154	2.1282	0.0154	2.1282	0.0154
		Random	2.6881	0.0692	2.6554	0.0538	2.8411	0.1154	2.8222	0.1000
		Popular	2.5508	0.0769	2.6948	0.0923	2.9072	0.1231	2.9403	0.1154
		PGA	2.5395	0.0615	2.5424	0.0538	2.7103	0.1000	2.8631	0.1154
		S-attack	2.6180	0.0846	2.8557	0.1231	2.7874	0.0923	2.9313	0.1154
		RevAdv	2.5250	0.0692	2.7005	0.0923	2.6618	0.0769	2.9862	0.1231
		Trial	2.6576	0.0769	2.5311	0.0538	2.8694	0.1000	3.0627	0.1462
	MCA	MSODS	**3.2100**	**0.1615**	**3.2177**	**0.1846**	**3.2022**	**0.1692**	**3.3953**	**0.2154**
Epinions	IA	None	1.2938	0.0000	1.2938	0.0000	1.2938	0.0000	1.2938	0.0000
		Random	1.9492	0.0000	2.6070	0.0104	2.4007	0.0104	2.5166	0.0208
		Popular	1.7655	0.0000	2.2960	0.0000	2.5757	0.0208	2.6094	0.0312
		PGA	1.9493	0.0000	2.3111	0.0000	2.3387	0.0000	2.4283	0.0000
		S-attack	1.8089	0.0000	2.2274	0.0000	2.3190	0.0208	2.5093	0.0312
		RevAdv	1.7721	0.0000	2.3980	0.0104	2.4116	0.0000	2.3946	0.0000
		Trial	1.8090	0.0000	2.3227	0.0000	2.3234	0.0000	2.4525	0.0104
	MCA	MSODS	**2.9252**	**0.1875**	**3.1662**	**0.2083**	**3.3865**	**0.2292**	**3.2414**	**0.1979**
library	IA	None	1.7001	0.0000	1.7001	0.0000	1.7001	0.0000	1.7001	0.0000
		Random	1.9865	0.0213	2.0482	0.0106	2.1590	0.0106	2.4537	0.0638
		Popular	2.0421	0.0213	2.1894	0.0426	2.2663	0.0426	2.3879	0.0426
		PGA	2.0316	0.0213	2.1229	0.0319	2.3192	0.0426	2.3629	0.0426
		S-attack	2.0486	0.0213	2.2473	0.0638	2.3050	0.0426	2.3725	0.0319
		RevAdv	1.9276	0.0213	2.0768	0.0213	2.3096	0.0319	2.4007	0.0532
		Trial	1.9687	0.0213	2.0220	0.0319	2.3881	0.0532	2.2885	0.0426
	MCA	MSODS	**2.7410**	**0.2234**	**3.0174**	**0.3085**	**3.0541**	**0.3830**	**3.0420**	**0.3085**

capacity than Ciao, since these two datasets have a sparser rating record than Ciao.

O3 : **Some references are missing, e.g., [1-3]**
[1] Adversarial attack on graph structured data. ICML 2018.
[2] Adversarial attacks on node embeddings via graph poisoning. ICML 2019.
[3] Adversarial attack on community detection by hiding individuals. WWW 2020.

Reply: Thanks for the valuable comments. As suggested, we extend the related works on adversarial attacks or poisoning attacks against Graph Neural Networks in Section II as follows.

> On the other hand, there are also other prior studies on test-time attack [64], [125]–[127] and train-time poisoning [60], [62], [128], [129] against different types of GNNs. For example, Dai et al. [126] formulate a reinforcement-learning based attack method against node classification and graph classification GNN models. Bojchevski and Günnemann [128] poisons the graph topology to manipulate node embeddings learned by graph methods. Besides, Li et al. [127] present an black-box attack on graph learning based community detection models using a novel graph generation neural network. However, while the above works targets different GNNs under a single attacker setting, our work is the first to investigate the GNN-based Het-RecSys under a multiplayer setting.

Specifically, the three mentioned papers are cited as [126], [128] and [127], respectively, in the revised manuscript.

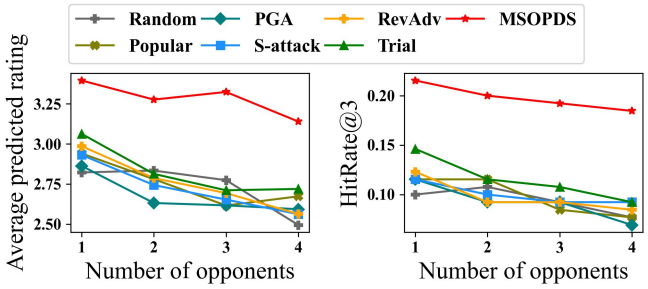
O4 : **Theorem 2-3 look like claims rather than theorem. Moreover, how these theorems serve your methods needs to be clarified.**

Reply: Thanks for the valuable comments. As suggested, we revise our theoretical analysis in Section V carefully to better connect with our proposed methods, MSOPDS.

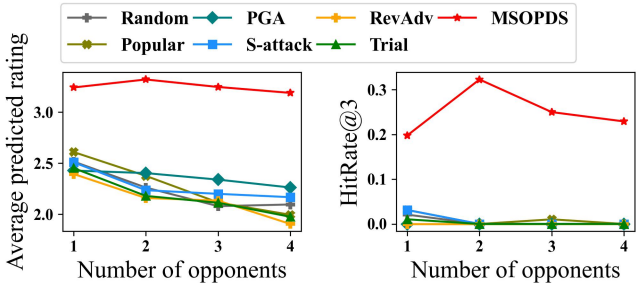
1) We emphasize the connection between the differential Stackelberg equilibrium with the update rules utilized in MSOPDS.
2) We add a new Theorem 2 with the proof on the existence of equilibrium.
3) We add a new Theorem 3 to present the proof on the convergence of MSOPDS to equilibrium.

In the following, we first introduce the definition of Stackelberg equilibrium [8] and differential Stackelberg equilibrium [5]. Then, we present the theorems on existence of equilibrium and on the guaranteed convergence. Note that the theoretical analysis are connected to MSOPDS in two ways.

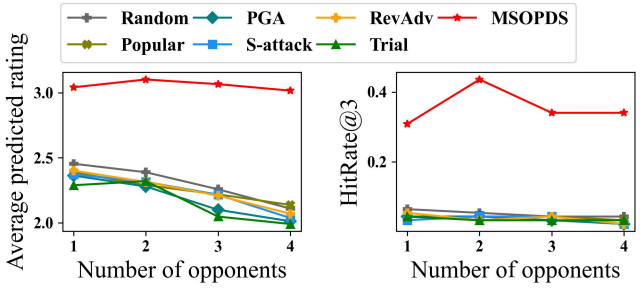
1) The update rules utilized in MSOPDS are aligned with the criterion utilized in the definition of differential Stackelberg equilibrium (Definition 7).
2) The learning rate of the attacker is set to be lower than that of the opponent in MSOPDS to meet the requirement for Theorem 3.



(a) Results on the Ciao dataset.

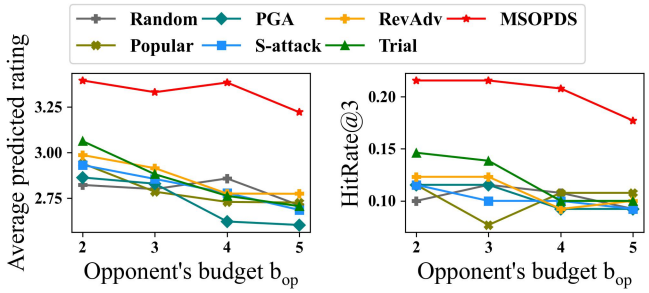


(b) Results on the Epinions dataset.

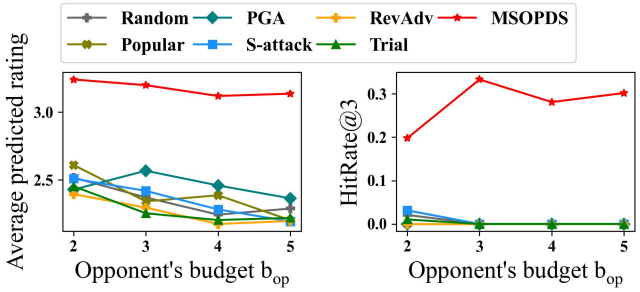


(c) Results on the LibraryThing dataset.

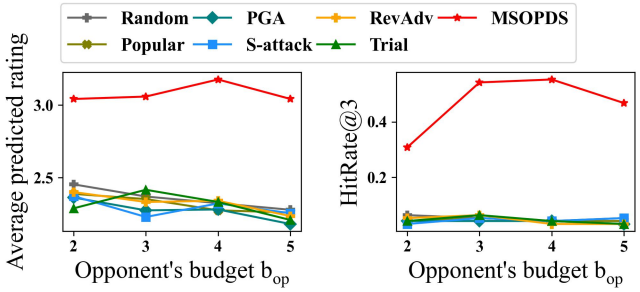
Fig. 5: Impact of the number of opponents.



(a) Results on the Ciao dataset.



(b) Results on the Epinions dataset.



(c) Results on the LibraryThing dataset.

Fig. 6: Impact of the opponent's capacity.

Definition 6 (**Stackelberg Equilibrium** [8]). *Let $\mathcal{C}^p$ and $\mathcal{C}^q$ be the capacity of the leader and the follower, respectively. The poisoning plan $\mathbf{X}^{p*} \in \mathcal{C}^p$ is a Stackelberg solution for the leader if $\forall \mathbf{X}^p \in \mathcal{C}^p$,*

$$\inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^{p*})} \mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^q) \geq \inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^p)} \mathcal{L}^p(\mathbf{X}^p, \mathbf{X}^q), \quad (17)$$

where $R_{\mathcal{C}^q}(\mathbf{X}^p) = \arg\max_{Y \in \mathcal{C}^q} \mathcal{L}^q(\mathbf{X}^p, Y)$ is the optimal solution set for the follower. More precisely,

$$(\mathbf{X}^{p*}, \mathbf{X}^{q*}), \forall \mathbf{X}^{q*} \in R_{\mathcal{C}^q}(\mathbf{X}^{p*}) \quad (18)$$

are the Stackelberg equilibriums.

Definition 7 (**Differential Stackelberg Equilibrium** [5]). *A pair of leader and follower poisons $(\mathbf{X}^{p*}, \mathbf{X}^{q*}) \in (\mathcal{C}^p, \mathcal{C}^q)$ is a Differential Stackelberg Equilibrium if*

$$\frac{d^2}{d{\mathbf{X}^p}^2}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \ \frac{\partial^2}{\partial {\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \quad (19)$$

$$\frac{d}{d\mathbf{X}^p}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = \frac{\partial}{\partial \mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0. \quad (20)$$

Notice that (20) concurs with the update rules of MSO, where the attacker p is updated by the *total derivative* $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ and the opponent q is updated by the *partial derivative* $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$. Intuitively, such update rules approach optimality individually. However, both must be carefully designed to arrive at optimality simultaneously to reach an equilibrium. An important criteria required by Theorem 3 below is that the step-size of the attacker be smaller than that of the opponent $\eta^p < \eta^q$. Thus, the requirement is adopted in MSO to reach an equilibrium.

Existence of equilibrium

We present the proof of the existence of equilibrium based on Definition 6. Moreover, we remove the strong assumption on RecSys model convergence.

Theorem 2. *Stackelberg equilibrium exists for MCA between one attacker (p) and one opponent (q).*

Proof. Following [9], we assume the GNN-based RecSys model has a finite graph G with the initial user $\mathbf{h}_u$ and item $\mathbf{h}_i$ embeddings satisfying the Gaussian distribution, which has been demonstrated to represent uncertainty [10]. According to Theorem 1 in [11], the convergence rate of Het-RecSys training is thereby $O(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}))$ by Polyak-Lojasiewicz inequality [12], namely that the squared norm of the gradient

$\nabla\mathcal{L}_{train}$ lower bounds the loss value at any iterate $|\mathcal{L}_{train,t} - \mathcal{L}_{train,t-1}|$. ϵ is the error rate. Therefore, with a set of $\mathbf{X}^p$ and $\mathbf{X}^q$, $\mathcal{L}^p$ and $\mathcal{L}^q$ are deterministic due to the convergence of Het-RecSys. Since the candidate sets $\mathcal{C}^p$ and $\mathcal{C}^q$ are finite sets and every possible strategy pair $(\mathbf{X}^p, \mathbf{X}^q)$ is finite under a given budget, an optimal strategy, i.e., Stackelberg equilibrium, exists over the finite set [13]. □

Guarantee of convergence

We present a proof of convergence based on Definition 7.

Theorem 3. *MSOPDS approaches differential Stackelberg equilibrium in MCA.*

Proof. Consider a differential Stackelberg equilibrium $\mathbf{X}^* \equiv (\mathbf{X}^{p*}, \mathbf{X}^{q*})$ such that $\frac{d\mathcal{L}^p(\mathbf{X}^*)}{d\mathbf{X}^p} = \frac{\partial\mathcal{L}^q(\mathbf{X}^*)}{\partial\mathbf{X}^q} = 0$, $\text{spec}(\frac{d^2\mathcal{L}^p(\mathbf{X}^*)}{d\mathbf{X}^{p\,2}}) \subset \mathbb{R}^\circ_+$ and $\text{spec}(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial\mathbf{X}^{q\,2}}) \subset \mathbb{R}^\circ_+$. Since $\det(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial\mathbf{X}^{q\,2}}) \neq 0$ and $\frac{\partial}{\partial\mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0$, by the implicit function theorem [14], there exists a neighborhood $\mathcal{N}_1$ of $\mathbf{X}^p$ and a unique function $f : \mathcal{N}_1 \to \mathbb{R}^{|\mathcal{C}^q|}$ such that $\frac{\partial}{\partial\mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^p, f(\mathbf{X}^p)) = 0\ \forall\mathbf{X}^p \in \mathbb{R}^{|\mathcal{C}^p|}$ and $f(\mathbf{X}^{p*})) = \mathbf{X}^{q*}$.[8]

Due to the fact that eigenvalues vary continuously, there exists a neighborhood $\mathcal{N}_2$ of $\mathbf{X}^p$ where $\frac{d^2}{d\mathbf{X}^{p\,2}}\mathcal{L}^p(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$ and $\frac{\partial^2}{\partial\mathbf{X}^{q\,2}}\mathcal{L}^q(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$. Let $\mathcal{U}_1 \subseteq \mathcal{N}_1 \cap \mathcal{N}_2$ be the non-empty, open set whose closure is contained in $\mathcal{N}_1 \cap \mathcal{N}_2$. Since $\frac{\partial^2}{\partial\mathbf{X}^{q\,2}}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0$, there exists an open neighborhood $\mathcal{U}_2$ of $\mathbf{X}^{q*}$ such that $\frac{\partial^2}{\partial\mathbf{X}^{q\,2}}\mathcal{L}^q(\mathbf{X}^p, \mathbf{X}^q) > 0\ \forall(\mathbf{X}^p, \mathbf{X}^q) \in \mathcal{U}_1 \times \mathcal{U}_2$. Since we have set the step-size of the attacker η^p and the subsequent opponent η^q to be $\eta^p < \eta^q$ in MSOPDS, according to (Lemma G.2 of [5]), the optimization process of MSOPDS on $(\mathbf{X}^p, \mathbf{X}^q)$ converges to $(\mathbf{X}^{p*}, \mathbf{X}^{q*})$ □

[8]Empirically, the numerical values of the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ and the partial derivative $\frac{\partial\mathcal{L}^q}{\partial\mathbf{X}^q}$ are all within reasonable range throughout the MSOPDS iterations, implying that they are L_1 and L_2-Lipschitz, respectively.

Planning Data Poisoning Attacks on Heterogeneous Recommender Systems in a Multiplayer Setting

Chin-Yuan Yeh[1,3], Hsi-Wen Chen[2], De-Nian Yang[3,4], Wang-Chien Lee[5], Philip S. Yu[6], Ming-Syan Chen[1,2]
[1]*Graduate Institute of Communication Engineering, National Taiwan University, Taiwan*
[2]*Department of Electrical Engineering, National Taiwan University, Taiwan*
[3]*Institute of Information Science, Academia Sinica, Taiwan*
[4]*Research Center for Information Technology Innovation, Academia Sinica, Taiwan*
[5]*Department of Computer Science and Engineering, The Pennsylvania State University, USA*
[6]*Department of Computer Science, University of Illinois Chicago, USA*
{cyyeh, hwchen}@arbor.ee.ntu.edu.tw, dnyang@iis.sinica.edu.tw
wlee@cse.psu.edu, psyu@uic.edu, mschen@ntu.edu.tw

***Abstract*—Data poisoning attacks against recommender systems (RecSys) often assume a single seller as the adversary. However, in reality, there are usually multiple sellers attempting to promote their items through RecSys manipulation. To obtain the best data poisoning plan, it is important for an attacker to anticipate and withstand the actions of his opponents. This work studies the problem of Multiplayer Comprehensive Attack (MCA) from the perspective of the attacker, considering the subsequent attacks by his opponents. In MCA, we target the Heterogeneous RecSys, where user-item interaction records, user social network, and item correlation graph are used for recommendations. To tackle MCA, we present the Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate (MSOPDS). The Multilevel Stackelberg Optimization (MSO) method is used to form the optimum strategies by solving the Stackelberg game equilibrium between the attacker and his opponents, while the Progressive Differentiable Surrogate (PDS) addresses technical challenges in deriving gradients for candidate poisoning actions. Experiments on Heterogeneous RecSys trained with public datasets show that MSOPDS outperforms all prior works by up to 10.6% in average predicted ratings and up to 11.4% in HitRate@3 for an item targeted by an attacker facing one opponent.**

***Index Terms*—data poisoning attack, recommender system, Stackelberg game, graph neural network**

I. Introduction

Recommender Systems (RecSys) are used in eCommerce to help users explore and discover items that may be of interest to them. The ranking of items in a RecSys can thus have a significant impact on their visibility and sales [130]. However, since RecSys use past user data to improve their recommendations, users can also collectively affect the behavior of the RecSys through their actions [131]. This opens the door to malicious actors for *data poisoning attacks*, which involve introducing fake or biased data into the system to manipulate its recommendations.

Data poisoning attacks against RecSys have been observed in both research [2] and real-world scenarios. For instance, fake reviews are prevalent in Yelp [132] and Amazon [133]. In the context of eCommerce, multiple sellers compete for a shared market, resulting in a complex multiplayer game. Unethical competitive actions have been observed on eCommerce, e.g., spamming fake comments to jeopardize other businesses [134]–[136]. Nevertheless, the scenario of multiple adversaries has been largely ignored in previous research. Additionally, prior research has mostly focused on basic RecSys that operate only on user-item interactions [137]. To address these gaps, we introduce a novel attack scenario called *Multiplayer Comprehensive Attack (MCA)*, where an attacker aims to influence the RecSys while anticipating subsequent *poisoning actions* taken by opponents in a Heterogeneous RecSys environment.

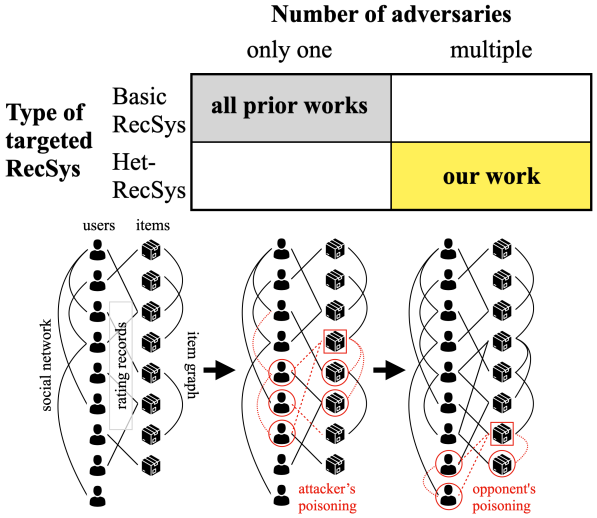


Fig. 1: Top: comparison of our problem setting with prior works. Our work studies data poisoning attack against Heterogeneous RecSys (Het-RecSys) under a multiple adversary setting. Bottom: illustration of the Multiplayer Comprehensive Attack problem. After an adversary (attacker) poisons the RecSys for his objective, another adversary (opponent) may later poison the RecSys with a different goal. Thus, if the first attacker does not anticipate the *subsequent* opponent's poison actions, his poison effort may be voided by the following opponent's poisons. Red lines indicate poison edges.

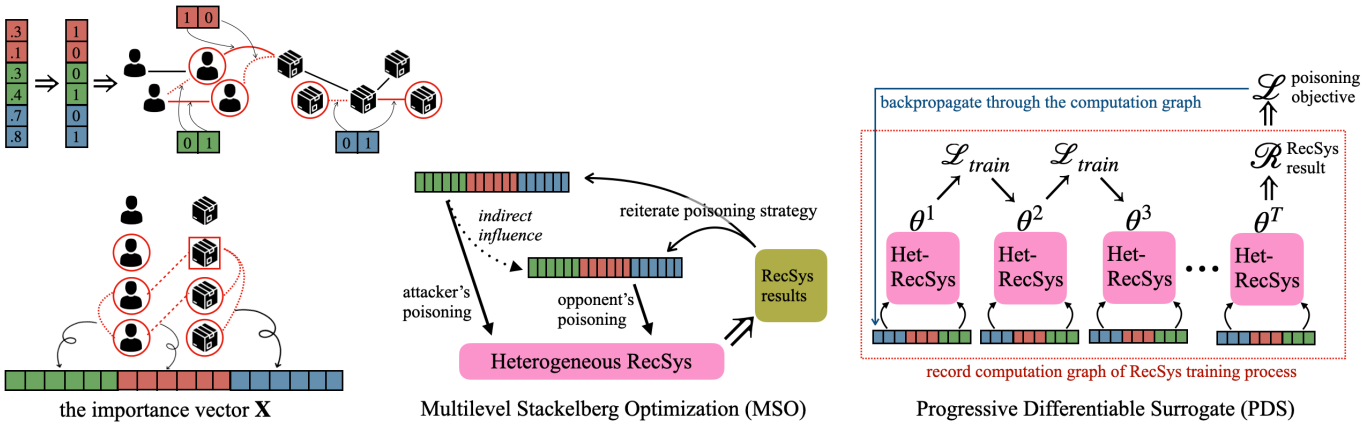


Fig. 2: A comprehensive high-level presentation of the different parts of MSOPDS, which consists of MSO and PDS, and utilizes the importance vector throughout the procedure. **Bottom left:** each element of the importance vector maps to a candidate poisoning action that may be of different type, e.g., adding poisoning edges in the social network (green) and item graph (blue), or adding poisoning ratings (red). **Upper left:** the element values of the importance vector represents the priority in action selection under the budget constraint. We obtain a poisoning plan from an importance vector by creating a *binarized* copy of the importance vector where 1 indicates the top-valued actions (red solid lines) that are selected under budget constraint, and 0 indicates not selected (red dashed lines). **Middle:** Illustration of MSO. In each iteration, the opponent's vector is updated based on how the poison directly influences the opponent's objective. Note that attacker's vector also considers his *indirect influence* on the opponent since the opponent will observe the attacker's poison when deciding his poisoning plan. **Right:** Illustration of PDS. The PDS calculates the partial derivatives of a poisoning objective w.r.t. importance vectors by incorporating it into the training process of a Het-RecSys. By recording the computation graph of the training process, the gradient is derived by backpropagation.

As shown in Top of Fig. 1, MCA is novel in the following ways: **1) Multiplayer Game.** Existing studies oversimplify the problem by considering only a single attacker [138]. In contrast, we consider a multiplayer game setting where an attacker who generates a batch of poison data may face opponents who also inject data into the RecSys for contradicting objectives, resulting in unexpected or counteractive effects. As shown in bottom of Fig. 1, since the later-added poisoning edges may interfere with the effort made by the first attacker to influence the RecSys, it is important to consider the actions of subsequent adversaries from the perspective of the first attacker in our setting. **2) Heterogeneous RecSys (Het-RecSys).** To align with real-world eCommerce platforms, we investigate the vulnerability of Het-RecSys [7] The difference between a basic RecSys and a more advanced Heterogeneous RecSys (Het-RecSys) is that the Het-RecSys processes three types of information (i.e., rating records, social network, and the item graph), while the basic RecSys only processes rating records. In other words, the factors considered in producing a recommendation for Het-RecSys are different, and it is thus essential to explore the vulnerabilities of additional types of poisoning actions. This is in contrast to prior works which only considered basic RecSys [2], [17], [20]. In our work, since Het-RecSys exploit the additional information, an attacker also conduct additional calculations for the optimal planning of poisoning actions beyond fake user-item interactions to include hiring real users as well as manipulations on the social network and item graph.[1] Finally, based on common marketing priorities in eCommerce platforms, we focus on promoting a specific item to a targeted audience [139] over a set of *competing items* [140]. Thus, our considerations are overall more "comprehensive" than previous research on data poisoning attacks against RecSys.

To solve the Multiplayer Comprehensive Attack problem, we propose Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate (MSOPDS), which consists of two key components, i.e., Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), with *importance vectors* representing poisoning plans for attackers. In particular, the importance vector is utilized to record the *priority* of each poisoning action. The benefit of using importance vectors is straightforward: it is much easier to perform iterative optimizations on vectors than on sets of discrete actions. Each element of the importance vector maps to a candidate poisoning action. Furthermore, heterogeneous types of actions can all be concatenated into the importance vector. As shown in the bottom left of Fig. 2, the green elements corresponds to adding poisoning edges in the social network, red elements corresponds to adding poisoning ratings, and blue elements corresponds to adding

[1] With the right incentives, eCommerce customers are often encouraged to leave reviews or ratings [112], [113] or connect with social media accounts [121]. Alternatively, real user accounts may also be purchased [114], [115] or hacked [116], [117], which gives manipulators direct control of those accounts.

poisoning edges in the item connection graph. However, due to the budget constraint, not all poisoning actions can be chosen. Therefore, we utilize a *binarizing step* to select a poisoning plan from the priorities recorded by the importance vector. As shown in the upper left of Fig. 2, we copy the importance vector into a *binarized* version where the top values of each section (i.e., each action type) are set to 1 and the rest to 0. Then, we can obtain a poisoning plan that abides by the budget constraint by selecting the poisoning actions with 1.

With the formulation of importance vectors, *Multilevel Stackelberg Optimization (MSO)* aims to iteratively approach the attacker's optimal solution (i.e., the optimal importance vector corresponding to an optimal poisoning plan) by *simultaneously* modeling both his and his opponent's actions to carefully examine the interactions between both players. In the interaction, the attacker conducts his poisoning actions first, and then the opponent plans the subsequent poisoning action based upon the attacker's poison. Such a scenario forms a Stackelberg game, and the update rules of MSO is designed based on the Stackelberg game framework to approach a Stackelberg equilibrium. The Stackelberg game framework is widely used in eCommerce analyses to models hierarchical relations between multiple players and to effectively exploit the first-mover (leader) advantages over the subsequent opponents (followers) [141]. Examples include marketing [142], real-time bidding [143], logistics coordination [144], and pricing strategies [145]. In the MSO framework, the attacker is the *leader* and the subsequent opponents are the *followers*. As shown in the middle of Fig. 2, MSO assists the attacker to anticipate the actions of subsequent opponents and approaches the Stackelberg game equilibrium by accounting for this indirect influence of the attacker on the opponent. By definition, under a game equilibrium, no player can change his plan to obtain a better gain for his objective. Intuitively, an equilibrium indicates that both players have obtained the optimal result and thus both follow the optimal poisoning plan.

Nevertheless, the partial derivatives required for the update rules of MSO cannot be directly computed due to the nature of Het-RecSys as a Graph Neural Network (GNN) [125]. Specifically, iterating fractional update steps on the poisoning action plans would lead to fractional edge values (esp. in the social network and the item graph) that could not be processed by the Het-RecSys. Prior works [2], [17] do not face such a challenge since they plan their data poisoning attacks against basic RecSys which do not utilize graph information.

To circumvent the difficulty, we propose the *Progressive Differentiable Surrogate (PDS)*. As shown at the right of Fig. 2, PDS finds the partial derivatives by incorporating the importance vectors into the training process of a Het-RecSys. This is because the training process determines the RecSys result $\mathcal{R}$, which in turn decides the poisoning objective as $\mathcal{L}(\mathcal{R})$. Specifically, to understand the process of training a RecSys and the impact of poisoning on its results, we record the computation graph of the training process. Then, by establishing how each element of the importance vector contributing to the training (poisoning) process of RecSys and to its final poisoned result $\mathcal{R}$, we can also understand how the importance vector affects the poisoning objective $\mathcal{L}$ via backpropagation. Intuitively, we consider the training process of the RecSys as a series of transitions from one iteration to the next, where the parameter obtained in each training iteration is represented by $\theta^1, \theta^2, \theta^3 \cdots \theta^T$ in Fig. 2 and the training loss denoted as $\mathcal{L}_{train}$. The RecSys result is a function of the final iteration parameter $\mathcal{R}(\theta^T)$. The poisoning objective, represented by $\mathcal{L}$, is a function of the RecSys result. Thus, by the chain rule, backpropagation through the entire process may yield the partial derivatives.

The contributions of this paper are as follows.

- This paper provides a more realistic modeling of data poisoning attacks against RecSys. It is the first to capture the dynamics between multiple players conducting attacks on the same RecSys. As for the targeted model, it is also the first to consider a more realistic RecSys, referred to as Heterogeneous RecSys, which exploits user relations via social networks as well as item relations.
- We formulate this pioneering problem as the *Multiplayer Comprehensive Attack (MCA)*, which captures real-world competitions among multiple sellers attacking a Heterogeneous RecSys.
- MSOPDS approaches an equilibrium amongst multiple objectives, instead of following the standard gradient optimization that directly optimizes on a single objective.
- New update rules in MSOPDS are designed in accordance with a formal analysis on guaranteed convergence to equilibrium based on recent theoretical results.
- MSOPDS presents Progressive Differentiable Surrogate, a novel GNN-based RecSys design, where poison edges and poison ratings are separately incorporated into the graph convolution process and the training loss, respectively.
- Extensive experiments on Heterogeneous RecSys trained with real-world datasets demonstrate the effectiveness of our work over the state-of-the-art methods when facing single or multiple opponents.[2]

II. Related Work

The vulnerability of RecSys against data poisoning attacks is well recognized in prior research [2], [17], [138]. In a data poisoning attack [146], an attacker *poisons* a machine learning model by injecting adversarial (fake) samples into the training data [147]. While earlier works on poisoning RecSys focus on heuristic approaches (e.g., popular attack or shilling attack [148]), recent works utilize optimization-based methods [2], [16], [17], [19], [20], [52], [149]–[152] to generate poison data that fool the RecSys models under specific requirements. For example, Li et al. [17] apply the Stochastic Gradient Langevin Dynamics to determine the poison data while adhering to the distribution of the rating records. Tang et al. [2] provided an in-depth analysis on the bilevel optimization for data poisoning attack on RecSys. In

[2]Source code: https://github.com/jimmy-academia/ICDE-extra-material

addition, Zhang et al. [20] tackle the problem of data poisoning attacks using modified and perturbed user-item interaction data. Besides, Song et al. [19] propose to utilize reinforcement learning to decide a poisoning strategy under limited information. Nevertheless, all the aforementioned works target only one single attacker attempting to manipulate the RecSys through data poisoning. In contrast, we formulate the problem of data poisoning attacks as a multiplayer game by considering the effects of the subsequent opponents' poisoning actions.

Furthermore, these works only focus on conducting the poisoning actions with fake user accounts [2], [17] for basic RecSys that only operates on the rating records (e.g., implicit clicks or explicit ratings) [20]. In contrast, we investigate the vulnerability of Graph Neural Network (GNN)-based Het-RecSys [153], [154] in a more realistic scenario with various poisoning actions involving real and fake users and allowing modifications on the rating records, social networks, and item graphs.

On the other hand, there are also other prior studies on test-time attack [64], [125]–[127] and train-time poisoning [60], [62], [128], [129] against different types of GNNs. For example, Dai et al. [126] formulate a reinforcement-learning based attack method against node classification and graph classification GNN models. Bojchevski and Günnemann [128] poisons the graph topology to manipulate node embeddings learned by graph methods. Besides, Li et al. [127] present an black-box attack on graph learning based community detection models using a novel graph generation neural network. However, while the above works targets different GNNs under a single attacker setting, our work is the first to investigate the GNN-based Het-RecSys under a multiplayer setting.

III. Problem Formulation

Tables I and II provide a notation table and an abbreviation table for reference.

A. Heterogeneous RecSys (Het-RecSys)

In general, RecSys provides a personalized ranking of all items for a given user based on the similarity between their embedding representations learned from past rating records [155]. *Het-RecSys* [7], [156] extends the basic RecSys by exploiting additional information. Besides, the rating records, Het-RecSys utilizes the social network $\mathcal{G}_U$ and the item graph $\mathcal{G}_I$ to model the social influences among users and associations among items and improve the recommendations.

Definition 1 (Het-RecSys). *Given the user set $\mathcal{U}$, item set $\mathcal{I}$, rating matrix $\mathbf{R} \in \Gamma^{|\mathcal{U}|\cdot|\mathcal{I}|}$ with $\Gamma \equiv \Xi \cup \{\varnothing\}$ being a rating out of 5 stars $\Xi \equiv \{1,2,3,4,5\}$ or a missing value $\varnothing$, social network $\mathcal{G}_U = \{\mathcal{U}, \mathcal{E}_U\}$, and item graph $\mathcal{G}_I = \{\mathcal{I}, \mathcal{E}_I\}$, a Het-RecSys predicts the missing rating values in $\mathbf{R}$.*

Utilizing the heterogeneous graph data, Graph Neural Network (GNN)-based Het-RecSys [7] are powerful because it encodes the entity features along with the graph topology through graph convolution processes [157]. Formally, with the Het-RecSys prediction result $\mathcal{R}(\theta, \mathcal{G})$ with $\mathcal{G} \equiv \mathcal{G}_U \cup \mathcal{G}_I$ and θ denoting the learnable parameters, the training process aims to minimizes the training loss $\mathcal{L}_{train}$,

$$\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \mathcal{G}), \mathbf{R}) + \lambda \|\theta\|_2^2, \tag{1}$$

where $\mathcal{L}_{train}$ is formulated as the Mean Square Error [158] between the predicted ratings $\mathcal{R}$ and the provided ratings $\mathbf{R}$ [7], and λ controls the strength of L_2 regularization [6].

B. Data poisoning attack scenarios

In this work, we investigate the effects of multiple adversaries conducting data poisoning attacks [19] on the same RecSys. Since eCommerce are often open and public [20],

TABLE I: List of Notations.

symbol	descriptions
u, i	An individual user or an item.
i_t^p	The target item of player p.
$\mathcal{U}, \mathcal{I}$	The (real) user set and item set.
$\mathcal{U}_{fake}^p$	The set of fake users created by player p. $\mathcal{U}_{fake}^p \not\subseteq \mathcal{U}$
$\mathcal{U}_{TA}^p$	The target audience of player p.
$\mathcal{U}_{base}^p$	The customer base of player p.
$\mathcal{I}_{product}^p$	The company products of player p.
$\mathcal{I}_{compete}^p$	The competing items of player p.
Ξ, $\varnothing$	The set of ratings values $\{1,2,3,4,5\}$ or missing value
$\mathbf{R}$	The rating matrix ($\mathbf{R} \in \Gamma^{\lvert\mathcal{U}\rvert\cdot\lvert\mathcal{I}\rvert}, \Gamma \equiv \Xi \cup \{\varnothing\}$).
$\mathcal{G}_U$, $\mathcal{G}_I$	The social network and item graph.
$\mathcal{G}$	The heterogeneous graph information ($\mathcal{G} \equiv \mathcal{G}_U \cup \mathcal{G}_I$).
$\mathcal{R}(\boldsymbol{\theta}, \mathcal{G})$	The rating predictions of a RecSys with parameter $\boldsymbol{\theta}$.
$\hat{\mathbf{R}}, \hat{\mathcal{G}}$	The poisoned ratings and the poisoned graph data.
$\mathcal{R}_{(u,i)}$	The ratings predictions (between user u and item i).
$\boldsymbol{h}_u, \boldsymbol{h}_i$	The initial embeddings of user u or item i.
$\boldsymbol{h}_u^{(f)}, \boldsymbol{h}_i^{(f)}$	The final embeddings of user u or item i.
$\mathcal{L}_{train}$	RecSys training loss, i.e., the Mean Square Error loss.
$\mathcal{L}^p$	The adversarial loss, i.e., the poisoning objective (of p).
$\mathcal{C}^p$	The capacity set, i.e., candidate poisoning actions (of p).
$\mathcal{X}^p$	The poisoning action plan of player p
$(\hat{\mathbf{X}}^p)$ $\mathbf{X}^p$	The (binarized) importance vector of player p.

TABLE II: List of Abbreviations.

acronym	full phrase and descriptions
Het-RecSys	Heterogeneous RecSys, a Recommender System that utilizes heterogeneous graph information $\mathcal{G}$. In contrast, basic RecSys only utilizes the rating records $\mathbf{R}$.
MCA	Multiplayer Comprehensive Attack. Our problem setting which concerns how to conduct data poisoning attack against Het-RecSys while anticipating for subsequent opponent poisonings.
CA	Comprehensive Attack. A poisoning attack setting that targets Het-RecSys but assumes single adversary.
IA	Injection Attack. Prior works' poisoning attack setting. Targets basic RecSys and assumes single adversary
MSOPDS	Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate, our proposed method.
MSO	Multilevel Stackelberg Optimization, provides update rules to approach equilibrium for an attacker facing opponents.
PDS	Progressive Differentiable Surrogate, provides surrogate model for gradient computation over RecSys training.
BOPDS	Bi-level Optimization over Progressive Differentiable Surrogate, an ablation of MSOPDS that replaces MSO with a bi-level framework.

each adversary may have access to the full set of records, including any poison data that was injected *prior to his action* by other adversaries. Our goal is thus to understand the potential impact of *subsequent* poisoning attacks on an attacker's data poisoning plan to manipulate a Het-RecSys.

To present a clear distinction between the scenario assumptions, we formally define three different scenarios of attacks, namely, Injection Attack (IA), Comprehensive Attack (CA), and Multiplayer Comprehensive Attack (MCA). As compared in Table III, while IA targets basic RecSys and CA targets Het-RecSys, they both assume single adversary and is oblivious to the subsequent opponents. On the other hand, MCA targets Het-RecSys and aims to prepare a *resilient* poisoning plan in anticipation of the subsequent opponent actions. Notice that all prior works on data poisoning attack against RecSys [2], [16], [17], [151] can be categorized as IA.

TABLE III: Comparison of attack scenarios.

	IA	CA	MCA
targeted RecSys	basic	Het-RecSys	Het-RecSys
# of adversary	1	1	multiple

Definition 2 (**Bi-level formulation of data poisoning attack**). *Data poisoning attack (by single adversary) can be formulated as an bi-level optimization [2], where an adversarial loss (i.e., the poisoning objective) $\mathcal{L}$ describing the attacker's objective is optimized in the upper-level, and the training of RecSys over the poisoned data is described in the lower-level.*

$$\begin{aligned} &\min_{\mathcal{X}\subseteq\mathcal{C}} \mathcal{L}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\ &given\ \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}), \end{aligned} \tag{2}$$

where $\mathcal{X}$ is the poisoning plan selected (under relevant budget constraints) from the attacker's capacity set $\mathcal{C}$, $\mathcal{R}$ is the RecSys results, while $\hat{\mathbf{R}}$ and $\hat{\mathcal{G}}$ represents the poisoned rating records and the poisoned graph information, respectively.

IA and CA ascribe to the single player assumption and may be formulated following the Bi-level optimization framework by specifying the poisoning objective $\mathcal{L}$ and the capacity $\mathcal{C}$.

Definition 3 (**Injection Attack (IA)**). *An injection attack aims to maximize the rating of a target item i_t to all real users $u \in \mathcal{U}$ by creating fake ratings with a set of fake users $\mathcal{U}_{fake}$. The* Injection Attack loss *$\mathcal{L}_{IA}$ is defined as*

$$\mathcal{L}_{IA} = -\frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \mathcal{R}_{(u,i_t)}, \tag{3}$$

where subscripts are used in $\mathcal{R}_{(u,i_t)}$ to denote the individual rating predictions between user u and the target item i_t. Following prior works [2], [17], [19], [20], Injection Attack exploits a set of fake users $\mathcal{U}_{fake} \not\subseteq \mathcal{U}$ that are injected into the RecSys. Its capacity set $\mathcal{C}_{IA}$ consists of each fake user giving fake ratings to any items, where a budget constraints restricts how many items each fake user may rate.

$$\mathcal{C}_{IA} = \{(u, i, r) \mid u \in \mathcal{U}_{fake},\ i \in \mathcal{I}, r \in \Xi\} \tag{4}$$

While IA has been proven effective, it targets basic RecSys and may not be aligned with real-world scenarios. Thus, we formulate Comprehensive Attack as follows.

Definition 4 (**Comprehensive Attack (CA)**). *To align with real-world scenarios, we formulate Comprehensive Attack to focus on promoting the target item i_t to its target audience $\mathcal{U}_{TA}$ over competing products $\mathcal{I}_{compete}$ with the* Comprehensive Attack loss *$\mathcal{L}_{CA}$ defined as follows.*

$$\mathcal{L}_{CA} = \frac{1}{|\mathcal{U}_{TA}|} \sum_{u\in\mathcal{U}_{TA}} \sum_{i_c\in\mathcal{I}_{compete}} SELU(\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)}), \tag{5}$$

where the scaled exponential linear units (SELU) [1] are used to emphasize the individual terms with the target item i_t losing to competing items, i.e., $\mathcal{R}_{(u,i_c)} - \mathcal{R}_{(u,i_t)} \geq 0$.

Furthermore, according to common practice, we formulate Comprehensive Attack to leverage a set of customer base *users $\mathcal{U}_{base} \subset \mathcal{U}$ besides the fake users $\mathcal{U}_{fake}$, and a set of* company products *items $\mathcal{I}_{product} \subset \mathcal{I}$ to assist the* comprehensive *poisoning actions on both the rating and the heterogeneous graph information.*

Specifically, the capacity set $\mathcal{C}_{CA}$ consists of hiring users from the customer base *$\mathcal{U}_{base}$ to rate the target item with a preset rating $\hat{r} = 5$, hiring users from the* customer base *to connect to each fake account (on $\mathcal{G}_U$) and selecting items from his* company products *$\mathcal{I}_{product}$ to connect to the target item (on $\mathcal{G}_I$). The budget constraints restricts how many users (items) may be hired (selected).*

$$\begin{aligned} \mathcal{C}_{CA} = &\{(u, i_t, \hat{r}) \mid u \in \mathcal{U}_{base}\} \\ &\cup \{(u, u_f) \mid u \in \mathcal{U}_{base},\ u_f \in \mathcal{U}_{fake}\} \\ &\cup \{(i, i_t) \mid i \in \mathcal{I}_{product}\} \end{aligned} \tag{6}$$

For comparison, note that CA does not spam fake ratings with fake users like IA but explores vulnerabilities in the heterogeneous graph information.

Ultimately, we aim to investigate the scenario where an attacker faces one or many opponents. To account for the opponents' actions, we formulate Multiplayer Comprehensive Attack (MCA), where an attacker can optimize his poisoning plan by considering the expected (future) poisoning plans of his opponents. Notice that MCA inherites the same loss $\mathcal{L}_{CA}$ and the capacity (and budget constraint) $\mathcal{C}_{CA}$ from CA. However, while CA ignores other opponents, MCA anticipates the subsequent poisoning actions by opponents (who are conducting CAs) based on their respective objectives and capacities. MCA cannot be solved by the bi-level optimization (Definition 2), but must be formulated the following *multilevel optimization*.

Definition 5 (**Multiplayer Comprehensive Attack (MCA)**). *In general, an attacker may face N individual opponents, resulting in an $(N+1)$-player game, and thereby an $(N+2)$-level optimization problem. Given the attacker p and his opponents $q_i, i \in [1, N]$. Let $\mathcal{L}_{CA}^{p}$ and $\mathcal{L}_{CA}^{q_i}$ be the* Comprehensive

Attack loss *and* $\mathcal{C}_{CA}^{p}$ *and* $\mathcal{C}_{CA}^{q_i}$ *be the capacity of the attacker (leader)* p *and the opponent (follower)* q_i*, respectively. The attacker* p *aims to solve:*

$$\begin{aligned}
&\min_{\mathcal{X}^p \subseteq \mathcal{C}_{CA}^{p}} \mathcal{L}_{CA}^{p}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\mathcal{X}^{q_1} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}_{CA}^{q_1}} \mathcal{L}_{CA}^{q_1}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\qquad \vdots \\
&\mathcal{X}^{q_N} \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}_{CA}^{q_N}} \mathcal{L}_{CA}^{q_N}(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\textit{given } \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{R}(\theta, \hat{\mathcal{G}}), \hat{\mathbf{R}}) + \lambda \|\theta\|^2,
\end{aligned} \tag{7}$$

where similar to (2) in Definition 2, $\mathcal{R}$ *is the RecSys result parameterized by* θ*,* $\hat{\mathbf{R}}$ *and* $\hat{\mathcal{G}}$ *are the rating records and the heterogeneous graph poisoned by all* $N+1$ *players, while* $\mathcal{X}^p$ *and* $\mathcal{X}^{q_i}$ *are the sets of poisoning actions selected by* p *and* q_i*, respectively.*

IV. The MSOPDS Method

Here, we introduce the *Multilevel Stackelberg Optimization over Progressive Differentiable Surrogate (MSOPDS)*, a framework that allows an attacker to plan his poisoning plan against subsequent opponents' poisoning plans to solve MCA. We first introduce the formulation of the *importance vector* to transform the original discrete set selection problem into continuous vector optimization. Then, we introduce the MSO framework which updates the attacker's poisoning plan while simulating the corresponding opponents' poisoning plans and estimating the total poisoning effect on the Het-RecSys, with all player's plans instantiated by *importance vector*s. We then present PDS, which facilitates computations of the required derivatives for the update rules calculated in MSO for each player's poisoning plan. Finally, we detail the algorithm for MSOPDS and present the pseudo-code in Algorithm 1.

A. The importance vector

To facilitate a gradient-based approach that iteratively adjusts intermediate plans by appropriate update rules, we formulate the *importance vector* $\mathbf{X} \in \mathbb{R}^{|\mathcal{C}|}$ where each element x indicates the *priority* of the corresponding candidate poisoning action in $\mathcal{C}$. Given the budget constraints for each type of poisoning actions, the importance vector $\mathbf{X}$ maps to a poisoning plan $\mathcal{X}$ consisting of the actions corresponding to the top-valued elements in $\mathbf{X}$. We elucidate how an attacker would optimize $\mathbf{X}$ in the following with MSO and PDS.

B. Multilevel Stackelberg Optimization (MSO) framework

1) two-player game: We first explore the analytical solution to MCA in the simplified two-player setting between the attacker and a single opponent, then extend to the full solution of N opponents. Under the two-player setting, (7) becomes

$$\begin{aligned}
&\min_{\mathcal{X}^p \subseteq \mathcal{C}^p} \mathcal{L}^p(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\mathcal{X}^q \in \arg\min_{\mathcal{X} \subseteq \mathcal{C}^q} \mathcal{L}^q(\mathcal{R}(\theta^*, \hat{\mathcal{G}})) \\
&\textit{given } \theta^* \in \arg\min_{\theta} \mathcal{L}_{train}(\theta, \mathcal{R}(\theta^*, \hat{\mathcal{G}}), \hat{\mathbf{R}}) + \lambda \|\theta\|^2,
\end{aligned} \tag{8}$$

where subscripts $_{CA}$ are removed from $\mathcal{C}$ and $\mathcal{L}$ for simplicity.

As displayed in (8), the attacker optimize his poisoning plan $\mathcal{X}^p$ (top level of (8)) *over* a bi-level optimization process of his opponent q solving for $\mathcal{X}^q$ (middle and lower level of (8)). Accordingly, if the attacker decides on a specific poisoning plan $\mathcal{X}^p$, he may simulate the final RecSys result $\mathcal{R}$ by solving his opponent's poisoning plan with the rating records $\mathbf{R}$ and heterogeneous graph $\mathcal{G}$ already poisoned with $\mathcal{X}^p$. Following prior works [2] and representing the poisoning plans $\mathcal{X}^p$ and $\mathcal{X}^q$ with the importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$, we present the basic update rule for the opponent as

$$\mathbf{X}^q \leftarrow \mathbf{X}^q - \eta^q \frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}, \tag{9}$$

where η^q is the step size of the opponent and the partial derivative $\partial \mathcal{L}^q / \partial \mathbf{X}^q$ represents how a change in $\mathbf{X}^q$ directly affects $\mathcal{L}^q$.

After solving for $\mathbf{X}^q$, a possible idea is to then update $\mathbf{X}^p$ with update rules similar to (9) with $\mathbf{R}$ and $\mathcal{G}$ poisoned by $\mathcal{X}^q$. However, such an approach fail to consider the current $\mathbf{X}^q$ was solved for the already given $\mathbf{X}^p$, and would change again after $\mathbf{X}^p$ is updated. Intuitively, such an approach would not lead to an equilibrium but rather a never-ending cycle where one player constantly tries to outmaneuver the other.

We thus present the following update rule for the attacker

$$\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \frac{d \mathcal{L}^p}{d \mathbf{X}^p}, \tag{10}$$

where

$$\frac{d \mathcal{L}^p}{d \mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} \tag{11}$$

is the *total derivative* of $\mathcal{L}^p$ *w.r.t.* $\mathbf{X}^p$, representing how the attacker's actions $\mathbf{X}^p$ influence his objective $\mathcal{L}^p$ directly in the first partial derivative term, and indirectly due to the subsequent opponent reactions in the second term. By utilizing the *total derivative*, the attacker is able to incorporate and adapt to the anticipated opponent's actions. However, as the term $\frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p}$ cannot be easily computed, we aim to expand (11) into a formula that is *computable*.

We first consider a temporary construction where after each attacker update step, the opponent updates by (9) until convergence, i.e., $\partial \mathcal{L}^q / \partial \mathbf{X}^q = 0$. As shown in left of Fig. 3, such an optimality condition implicitly defines $\mathbf{X}^q$ as a function of $\mathbf{X}^p$ through the dependence of $\mathcal{L}^q$ on $\mathbf{X}^p$ by the implicit function theorem [14]. We find

$$\frac{d}{d \mathbf{X}^p} \left(\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q} \right) = \frac{\partial^2 \mathcal{L}^q}{\partial {\mathbf{X}^q}^2} \frac{\partial \mathbf{X}^q}{\partial \mathbf{X}^p} + \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} = 0 \tag{12}$$

and thus

$$\frac{d \mathcal{L}^p}{d \mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial {\mathbf{X}^q}^2} \right)^{-1} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} \tag{13}$$

by rearranging the terms in (12) and applying the resulting formulation for $\partial \mathbf{X}^q / \partial \mathbf{X}^p$ to (10).

Equation (13) is attained under the strict assumption that the opponent's plan $\mathbf{X}^q$ is updated to convergence after every attacker update. Nevertheless, such a requirement will

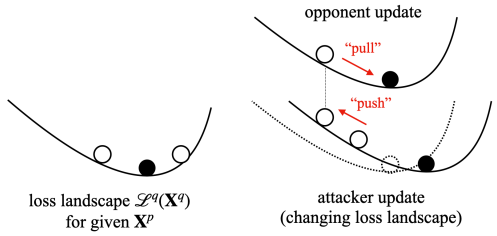


Fig. 3: Left: given the attacker's poison $\mathbf{X}^p$, the opponent's loss landscape $\mathcal{L}^q(\mathbf{X}^q)$ is fixed. Intuitively, there is a corresponding optimal opponent poison (solid circle) but arbitrarily many suboptimal points (outline circle). Right: An illustration of "push" vs. "pull". The opponent's update step takes it closer to the optimal, resembling a "pull" to the optimal position. However, the attacker's update step may shift the loss landscape such that the corresponding opponent's position is further away from optimal, resembling a "push".

inevitably cause the entire algorithm to be very slow. Thus, using recent results of Fiez et al. [5], we proceed relax the requirement and construct the procedure where both $\mathbf{X}^p$ and $\mathbf{X}^q$ are *updated simultaneously* by (13) and (9), respectively, on the condition that $\eta^p < \eta^q$. With formal theoretical analysis detailed in Section V, a high-level explanation is presented as follows. Under the former strict requirement, the opponent (i.e., the follower) is always playing a local best response strategy at each iteration, forming an *optimal trajectory*. As illustrated in the right of Fig. 3, in the relaxed case, (9) still pulls $\mathbf{X}^q$ towards while the attacker update changes the loss landscape of $\mathcal{L}^q$ and pushes $\mathbf{X}^q$ away from the *optimal trajectory*. Intuitively speaking, $\eta^p < \eta^q$ provides the key ingredient that guarantees the former overcomes the latter such that the new trajectory of $\mathbf{X}^q$ asymptotically converges to the *optimal trajectory*.

2) General multiplayer game: To efficiently solve the general multiplayer MCA game with N opponents, we focus on the interaction between the attacker and each opponent. In particular, since each opponent conducts Comprehensive Attack and is oblivious to the other players, we may reapply the attacker update rule (13) with minor adjustments. Following [159], we simplify (7) by flattening the lower-level objectives of opponents $q_1 \cdots q_N$ into a single-level optimization.[3] With $\mathbf{X}$s denoting the importance vectors as above, we can rewrite (13) into a multi-opponent version as follows.

$$\frac{d\mathcal{L}^p}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \sum_{i=1}^{N} \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^{q_i}} \left(\frac{\partial^2 \mathcal{L}^{q_i}}{\partial \mathbf{X}^{q_i 2}} \right)^{-1} \frac{\partial^2 \mathcal{L}^{q_i}}{\partial \mathbf{X}^p \partial \mathbf{X}^{q_i}}, \tag{14}$$

where the second term in (13) is extended into the summation that covers the indirect effects of all opponents. Thus, MSO solves (7) in the general setting by iteratively updating $\mathbf{X}^p$ by (10) with the *total derivative* $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ defined by (14) and while each simulated opponent poisoning plan $\mathbf{X}^{q_i}$ is still updated by $\frac{\partial \mathcal{L}^{q_i}}{\partial \mathbf{X}^{q_i}}$ as in (9).

[3]In other words, all opponents are considered to be in a subgame-perfect equilibrium [160]

C. Progressive Differentiable Surrogate

To compute the required derivatives in the update rules (13) and (9), we introduce the Progressive Differentiable Surrogate (PDS). Following GNN-based RecSys design [6], [7], PDS utilize a set of user $\boldsymbol{h}_u$ and item $\boldsymbol{h}_i$ embeddings, obtains the final user $\boldsymbol{h}_u^f$ and item $\boldsymbol{h}_u^f$ embeddings through graph convolution on the respective graphs $\mathcal{G}_U$ and $\mathcal{G}_I$, and is trained by (1) with the RecSys ratings prediction $\mathcal{R}_{(u,i)} \equiv \boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f$ obtained through dot products of between pairs of final embeddings.

Intuitively, we must *incorporate* the importance vectors $\mathbf{X}$ throughout the training process of PDS for $\mathbf{X}$ to influences the RecSys results and eventually the loss objectives. PDS first obtains the *binarized* importance vector $\hat{\mathbf{X}}$ (representing a tentative attack plan $\mathcal{X}$) by setting the elements of $\mathbf{X}$ with the largest values to 1 and others to 0, according to budget. Then, PDS utilize $\hat{\mathbf{X}}$ in the graph convolutional processes for the final embeddings as follows.

$$\begin{aligned} \boldsymbol{h}_u^f &= \mathbf{W}_U^\top \left(\boldsymbol{h}_u \oplus \sum_{n \in \mathcal{N}_U(u)} \mathbb{1}_{\mathcal{C}} \hat{x}_U(u,n) \frac{\boldsymbol{h}_n}{|\mathcal{N}_U(u)|} \right) \\ \boldsymbol{h}_i^f &= \mathbf{W}_I^\top \left(\boldsymbol{h}_i \oplus \sum_{m \in \mathcal{N}_I(i)} \mathbb{1}_{\mathcal{C}} \hat{x}_I(i,m) \frac{\boldsymbol{h}_m}{|\mathcal{N}_I(i)|} \right), \end{aligned} \tag{15}$$

where $\mathbf{W}_U$ and $\mathbf{W}_I$ are trainable projection matrices, $\oplus$ denotes concatenation, $\mathcal{N}_U(v)$ and $\mathcal{N}_I(i)$ represent the first-hop neighbors of u and i in $\mathcal{G}_U$ and $\mathcal{G}_I$, respectively. $\hat{x}_U(u,n)$ and $\hat{x}_I(i,m)$ are element values of $\hat{\mathbf{X}}$ that corresponds to the poisoning edge between u and n in the social network and between i and m in the item graph, respectively, whereas $\mathbb{1}_{\mathcal{C}}$ is a selection function that defaults to one, but adopts element value of the binarized importance vector ($\hat{x}_U(u,n)$ or $\hat{x}_I(i,m)$) if the link ((u,n) or (i,m) is in the candidate set $\mathcal{C}$. In other words, the selection function applies the selection decision presented in the binarized element values into the graph convolution process and corresponds to whether the surrogate model perceives that the corresponding edge exists.

Furthermore, PDS replaces $\mathcal{L}_{train}$ in (1) with a $\hat{\mathbf{X}}$-modulated loss $\mathcal{L}_{train-PDS}$ as follows.

$$\begin{aligned} \mathcal{L}_{train-PDS} = &\sum_{(u,i,r) \in \mathbf{R}} \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - r \right)^2 \\ &+ \sum_{(u,i) \in \mathcal{C}_R} \hat{x}_R(u,i) \left(\boldsymbol{h}_u^f \cdot \boldsymbol{h}_i^f - \hat{r} \right)^2, \end{aligned} \tag{16}$$

where $\mathbf{R}$ is the real ratings records, $\mathcal{C}_R$ is the set of candidate poisoning actions that include adding item ratings (with real or fake users), $\hat{x}_R(u,i)$ are element values of $\hat{\mathbf{X}}$ that correspond to the poisoning action of adding a rating $r_{u,i}$ with user u on item i, and $\hat{r}$ is a predefined rating value. Note that $\hat{r}$ can be set to a high value to influence PDS to predict higher ratings (i.e., promotion), or vice versa.

With (15) and (16), the binarized importance vector $\hat{\mathbf{X}}$ corresponding to poisoning actions on the social network $\mathcal{G}_U$

and item graph $\mathcal{G}_I$ affects the final user and item embeddings through (15), while those corresponding to poisoning actions on the rating records $\mathbf{R}$ affects the training process through the training loss (16). Since *all* element of $\mathbf{X}$ will be utilized in either (15) or (16), no matter the element value being 0 or 1, derivatives with respect to the whole vector may be computed.

we present Fig. 4 to illustrate the detailed mechanisms of the above equations step by step within one iteration of MSOPDS.

1) The importance vector is binarized according to budget constraints to find the current state of the poisoning plan. Here, Fig. 4 assumes a budget of selecting 1 from each type of poisoning actions. For example, the element values corresponding to the two poisoning actions on the rating record are 0.3 and 0.1. The result after binarizing is thus 1 and 0, corresponding to the plan of selecting the first but not the second option.
2) As described by (15), binarized element values 1 and 0 for the two candidate poisoning action on each of the social network and item graph are utilized in the graph convolution process. In Fig. 4, the dotted red lines in the social network and item graph corresponds to a binarized value of 0, while the solid red lines corresponds to 1.
3) On the other hand, the binarized element values corresponding to poisoning action on the rating records are applied to each candidate poisoning terms in the training loss (16). For instance, in Fig. 4, the dotted red curve represents a candidate poisoning rating with binarized value 0 while the other solid red curve represents 1. Thus, in (16), their $\hat{x}_R$ is $\hat{x}_R = 1$ for the former and $\hat{x}_R = 0$ for the latter case.
4) Note that in the above two steps, all candidate poisoning actions on the graphs and the rating records have already taken place within PDS to allow all elements to appear in the computation graph (even as a 0 value). Thus, all element values participate in the training process and are present in the recorded computation graph.
5) After calculating the poisoning loss objective $\mathcal{L}$ from the poisoned PDS RecSys results $\mathcal{R}$, the gradient can be obtained by backpropagation from $\mathcal{L}$ to $\mathcal{R}$ and eventually to each and every element.
6) The calculated update is applied to the importance vector. Here, Fig. 4 shows that the update results in a different binarized vector in the next iteration since the update changes the priority of each poisoning action.

D. The complete MSOPDS framework

Combining MSO and PDS, we present the complete MSOPDS framework. Fig. 5 presents an illustrative example of one iteration of MSOPDS for the two-player case. As shown in Fig. 5, the importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ are binarized into $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^p$, then fed into the PDS to simulate the poisoned RecSys result and to evaluate the poisoning objectives of the attacker $\mathcal{L}^p$ and the opponent $\mathcal{L}^q$. Finally, by the update rules of MSO, we update $\mathbf{X}^p$ by computing the *total derivative* of $\mathcal{L}^p$ w.r.t. $\hat{\mathbf{X}}^p$ and the *partial derivative* of $\mathcal{L}^q$ w.r.t. $\hat{\mathbf{X}}^q$.

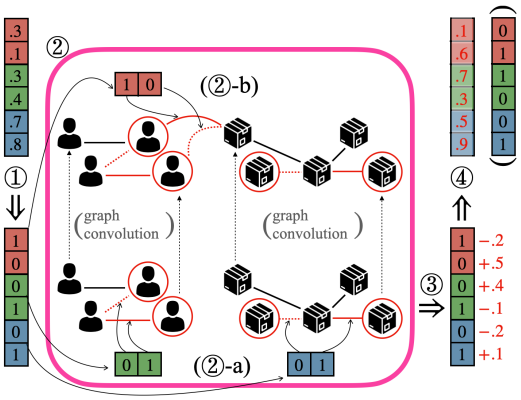


Fig. 4: Illustrative example of PDS. From the top left: ① the importance vector is binarized to select 1 of each type of poisoning action (budget constraint). ② the binary result is incorporated into the RecSys training process: (②-a) poison edges regulate the graph convolution on social network and item graph; (②-b) poison ratings modulate the training loss; ③ backpropagation through the recorded training process (in ②) computes the required derivatives to calculate the "gradient step" according to the update rule (9) or (10); ④ finally, the importance vectors are updated by the gradient calculated from the update rule. Comparing the binarized result of before (bottom left) and after (top right) this iteration, we can see that the update results in some of the selections changed.

Finally, Algorithm 1 presents the pseudo-code for the two-player case. As presented in Algorithm 1, an inner loop of PDS training is embedded within an outer loop of K iterations of updates for the importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ following the MSO update rules. Notice that in step 2, *all* candidate poisoning is inserted into the fully poisoned rating records $\mathbf{R}'$ and heterogeneous graph $\mathcal{G}'$, which is later regulated by the element values of the binarized importance vectors during the PDS training process. Therefore, by storing the inner loop RecSys training computations (step 6),[4] first-order derivatives of the losses w.r.t. the binarized importance vectors ($\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$, $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}$, and $\frac{\partial \mathcal{L}^q}{\partial \hat{\mathbf{X}}^q}$) can be obtained via backpropagation through the stored computation graph in step 8.

To reduce the computation cost of deriving the full Jacobian matrices,[5] we defer the realization of the Jacobian matrices and integrate the calculation of vector-Jacobian products into each iteration when solving $\boldsymbol{\xi}$ for the linear equation $\boldsymbol{\xi} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q\,2}} = \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q}$ by the conjugate gradient method [4] (step 9).[6]

Finally, following the update rules ((10) for the attacker and (9) for the opponent) as well as the calculations of (13) in MSO, step 10 and 11 updates the attacker's importance vector $\mathbf{X}^p$ and the opponent's importance vector $\mathbf{X}^q$, respectively.

[4]Achieved with the "higher" package [161].

[5]Note that $\frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^{q\,2}}$ is the Jacobian of $\frac{\partial \mathcal{L}^q}{\partial \hat{\mathbf{X}}^q}$ *w.r.t.* $\hat{\mathbf{X}}^q$, and $\frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^p \partial \mathbf{X}^q}$ is the Jacobian of $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ *w.r.t.* $\hat{\mathbf{X}}^p$.

[6]The process is achieved with Pytorch Autograd [3] and SciPy [162].

Algorithm 1: MSOPDS under the two player setting.

Input: Rating records $\mathbf{R}$, graph records $\mathcal{G}$; Comprehensive Attack losses $\mathcal{L}^p$ and $\mathcal{L}^q$; capacity sets $\mathcal{C}^p$ and $\mathcal{C}^q$ (with budget constraints represented as $\mathcal{B}^p$ and $\mathcal{B}^q$); step sizes η^p and η^q; outer and the inner loop max iteration K and L; PDS parameter θ.

Assert: $0 < \eta^p < \eta^q$

Output: Attacker's action plan $\mathcal{X}^p$.

1: Create importance vectors $\mathbf{X}^p$ and $\mathbf{X}^q$ from $\mathcal{C}^p$ and $\mathcal{C}^q$.
2: Create $\mathbf{R}'$ and $\mathcal{G}'$ by inserting all poisoning actions in $\mathcal{C}^p$ and $\mathcal{C}^q$ into the rating record $\mathbf{R}$ and the heterogeneous graph $\mathcal{G}$;
3: **for** $k = 1$ **to** K **do**
4: Binarize $\mathbf{X}^p$ and $\mathbf{X}^q$ into $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^q$ by $\mathcal{B}^p$ and $\mathcal{B}^q$.
5: **for** $l = 1$ **to** L **do**
6: Update $\theta^{(l)}$ by (1) with the training loss (16) and graph convolution (15) modulated by $\hat{\mathbf{X}}^p$ and $\hat{\mathbf{X}}^q$.
`// store the computation process.`
7: Derive the Comprehensive Attack losses $\mathcal{L}^p$ and $\mathcal{L}^q$ for both players with the trained (poisoned) RecSys results $\mathcal{R}(\theta^{(L)}, \mathcal{G}'')$.
8: Compute first-order partial derivatives $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$, $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}$, and $\frac{\partial \mathcal{L}^q}{\partial \hat{\mathbf{X}}^q}$ by backpropagation through the stored process.
`// retain computation graph for second-order vector-Jacobian product.`
9: Solve $\boldsymbol{\xi}$ for $\boldsymbol{\xi} \frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^{q\,2}} = \frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^q}$.
`// Calculate vector-Jacobian product within the conjugate gradient method.`
10: Update attacker $\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \left(\partial \mathcal{L}^p / \partial \hat{\mathbf{X}}^p - \boldsymbol{\xi} \frac{\partial^2 \mathcal{L}^q}{\partial \hat{\mathbf{X}}^p \partial \hat{\mathbf{X}}^q} \right)$.
`// Calculate vector-Jacobian product.`
11: Update opponent $\mathbf{X}^q \leftarrow \mathbf{X}^q - \eta^q \partial \mathcal{L}^q / \partial \hat{\mathbf{X}}^q$.

Remark. To perform MSOPDS for multiple opponents, we modify Algorithm 1 according to (14). Besides applying the same η^q for all opponents, the input as well as steps 1, 2, 4, 6 and 7 are straightforwardly adjusted to *include all opponent* q_i. Step 8 includes all first-order partial derivative described in (14) for the respective $\hat{\mathbf{X}}$s. Step 9 and 11 are repeated for each q_i while step 10 is adjusted according to (14).

Besides, we also present an ablation method named *Bi-level Optimization over Progressive Differentiable Surrogate (BOPDS)* by combining the bi-level optimization (Definition 2) with PDS to conduct Comprehensive Attack or Injection Attack, later used as a experiment comparison in Section VI. Similarly, we modify Algorithm 1 to perform the single-player BOPDS by removing the opponent q from the input as well as steps 1, 2, 4, 6 and 7, replacing the final steps $8-11$ by only computing $\frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$ and updating the attacker's importance vector by $\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \frac{\partial \mathcal{L}^p}{\partial \hat{\mathbf{X}}^p}$.

E. Acceleration techniques

We observe that MSOPDS is more computationally intense than standard gradient optimization processes, since it involves multiple vector-Jacobian product calculations. This is because MSOPDS allows the attacker to anticipate subsequent opponent's actions by accounting for the attacker's *indirect influence* on his opponent. In particular, in the attacker's update

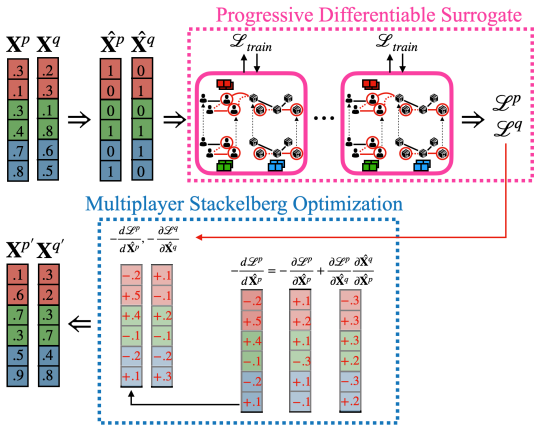


Fig. 5: Illustration of an iteration of MSOPDS. We highlight the two components of MSOPDS, namely, Multilevel Stackelberg Optimization (MSO) and Progressive Differentiable Surrogate (PDS), by blue and pink box, respectively.

rule in MSOPDS,

$$\mathbf{X}^p \leftarrow \mathbf{X}^p - \eta^p \left(\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p} - \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q\,2}} \right)^{-1} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q} \right),$$

the first update term ($\frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^p}$) can be efficiently computed with backpropagation, but the second *Stackelberg correction term* ($Q_2 \equiv \frac{\partial \mathcal{L}^p}{\partial \mathbf{X}^q} \left(\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^{q\,2}} \right)^{-1} \frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$) involves multiple vector-Jacobian product calculations. Therefore, we design three acceleration techniques, namely, *Competitor-aware Adaptive Gradient Reuse (reuse)*, *Stochastic Poisoning Action Omission (omis)*, and *Progressive Attack Influence Propagation (prop)*, to further improve MSOPDS. Details of these three techniques are presented below.

1) *Competitor-aware Adaptive Gradient Reuse (reuse).* In MSOPDS, we set a lower learning rate for the attacker's vector compared to the opponent's vector, following the idea that the attacker's vector should be updated slower [5]. With this in mind, we update the Stackelberg correction term less frequently and allow it to evolve slower in order to reduce the computational cost. Therefore, inspired by gradient variance reduction techniques [51], we store and reuse the Stackelberg correction term Q_2 and update it every 2 iterations. Note that we observe storing over longer iteration leads to reduced effectiveness.
2) *Stochastic Poisoning Action Omission (omis).* MSOPDS iteratively updates both the attacker's vector $\mathbf{X}^p$ and the opponent's vector $\mathbf{X}^q$ to reach an equilibrium. The iterations are required because the poison objective of each player can be influenced by the other player's decisions. For example, among different poisoning actions of the attacker, some actions may always contribute towards his poison objective regardless of the opponent's decision, while others are heavily affected by the opponent's actions. Thus, during the iterative update

in MSOPDS, some actions may remain selected (i.e., the corresponding element value in $\mathbf{X}^p$ is higher than the rest), while other elements may undergo stochastic changes from iterations to iterations. Based on the above observation, the idea is to only update the elements of the attacker's vector $\mathbf{X}^p$ with values ranked in the middle half after the initial iterations. Note that we resume full vector optimization in the final iterations. Besides, we do not apply the above procedure to the opponent's vector, because the opponent's responses need to remain optimal to theoretically achieve the convergence of MSOPDS. Note that, the computation for updating $\mathbf{X}^q$ is not the bottleneck because it only involves standard backpropagation.

3) *Progressive Attack Influence Propagation (prop).* After reexamine the graph convolution process of Het-RecSys, we further improve the efficiency of MSOPDS by starting from a smaller subset of the heterogeneous graph $\mathcal{G}' \subset \mathcal{G}$, and then progressively update the next-hop neighbors. This is because a poisoning edge first directly affects the embeddings of the two nodes that are connected by it, and then indirectly influence other nodes since their embeddings are updated through the RecSys training process. Therefore, the **prop** method first considers the terminal nodes and their first-hop neighbors of all candidate poisoning edge on the social network $\mathcal{G}_U$ and the item graph $\mathcal{G}_I$. The terminal nodes of the candidate poisoning edges include the customer base $\mathcal{U}_{base} \subset \mathcal{U}$ and the fake users $\mathcal{U}_{fake}$ as well as the company products $\mathcal{I}_{product} \subset \mathcal{I}$ and the target item i_t. It then progressively include the next-hop neighbors on $\mathcal{G}_U$ and $\mathcal{G}_I$ into the MSOPDS process until reaching the full heterogeneous graph.

V. Theoretical Analysis

Here, we present the theoretical analysis. We first provide the time complexity of MSOPDS as follows.

Theorem 1. *With MSOPDS, the time complexity are reduced from* $\Omega(|\theta|^{|\mathbf{X}^p||\mathbf{X}^q|})$ *to* $O(|\theta|+|\mathbf{X}^p||\mathbf{X}^q|)$*, where* $|\cdot|$ *represents the dimension of the model parameters or importance vectors.*

Proof. Since we avoid enumerating all possible combinations, the computational complexities of MSOPDS are reduced from the brute-force method. In particular, $O(|\theta|)$ is the complexity of calculating $\frac{\partial \mathcal{L}}{\partial X}$ with the reverse-mode algorithmic differentiation, which is proportional to the surrogate model size $|\theta|$ [163]. $O(|\mathbf{X}^p||\mathbf{X}^q|)$ is caused by the conjugate gradient method, which is proportional to the size of the Jacobian matrix $\frac{\partial^2 \mathcal{L}^q}{\partial \mathbf{X}^p \partial \mathbf{X}^q}$ [4]. □

Next, we prove the existence of Stackelberg game equilibrium and the convergence to such equilibrium in terms of player strategy optimization. For simplicity, we focus on the two player Stackelberg game between an attacker (leader) and one opponent (follower).

Definition 6 (**Stackelberg Equilibrium** [8])**.** *Let* $\mathcal{C}^p$ *and* $\mathcal{C}^q$ *be the capacity of the leader and the follower, respectively. The poisoning plan* $\mathbf{X}^{p*} \in \mathcal{C}^p$ *is a Stackelberg solution for the leader if* $\forall \mathbf{X}^p \in \mathcal{C}^p$*,*

$$\inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^{p*})} \mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^q) \geq \inf_{\mathbf{X}^q \in R_{\mathcal{C}^q}(\mathbf{X}^p)} \mathcal{L}^p(\mathbf{X}^p, \mathbf{X}^q), \quad (17)$$

where $R_{\mathcal{C}^q}(\mathbf{X}^p) = \arg\max_{Y \in \mathcal{C}^q} \mathcal{L}^q(\mathbf{X}^p, Y)$ *is the optimal solution set for the follower. More precisely,*

$$(\mathbf{X}^{p*}, \mathbf{X}^{q*}), \forall \mathbf{X}^{q*} \in R_{\mathcal{C}^q}(\mathbf{X}^{p*}) \quad (18)$$

are the Stackelberg equilibriums.

Following [5], we utilize the differential Stackelberg equilibrium as follows.

Definition 7 (**Differential Stackelberg Equilibrium** [5])**.** *A pair of leader and follower poisons* $(\mathbf{X}^{p*}, \mathbf{X}^{q*}) \in (\mathcal{C}^p, \mathcal{C}^q)$ *is a Differential Stackelberg Equilibrium if*

$$\frac{d^2}{d{\mathbf{X}^p}^2}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \ \frac{\partial^2}{\partial {\mathbf{X}^q}^2}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0, \quad (19)$$

$$\frac{d}{d\mathbf{X}^p}\mathcal{L}^p(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = \frac{\partial}{\partial \mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0. \quad (20)$$

For reasonably smooth loss functions $\mathcal{L}^p, \mathcal{L}^q$, (19) provides the necessary conditions for the implicit mapping utilized in (13), while the partial differentiation in (20) recreates the condition $R_{\mathcal{C}^q}$ in Definition 6, where the total differentiation corresponds to the optimality of X^p in (17).

Theorem 2. *Stackelberg equilibrium exists for MCA between one attacker (p) and one opponent (q).*

Proof. Following [9], we assume the GNN-based RecSys model has a finite graph G with the initial user $\mathbf{h}_u$ and item $\mathbf{h}_i$ embeddings satisfying the Gaussian distribution, which has been demonstrated to represent uncertainty [10]. According to Theorem 1 in [11], the convergence rate of Het-RecSys training is thereby $O(\frac{1}{\epsilon^2}\log(\frac{1}{\epsilon}))$ by Polyak-Lojasiewicz inequality [12], namely that the squared norm of the gradient $\nabla \mathcal{L}_{train}$ lower bounds the loss value at any iterate $|\mathcal{L}_{train,t} - \mathcal{L}_{train,t-1}|$. ϵ is the error rate. Therefore, with a set of $\mathbf{X}^p$ and $\mathbf{X}^q$, $\mathcal{L}^p$ and $\mathcal{L}^q$ are deterministic due to the convergence of Het-RecSys. Since the candidate sets $\mathcal{C}^p$ and $\mathcal{C}^q$ are finite sets and every possible strategy pair $(\mathbf{X}^p, \mathbf{X}^q)$ is finite under a given budget, an optimal strategy, i.e., Stackelberg equilibrium, exists over the finite set [13]. □

We continue to assert the convergence of MSOPDS towards differential Stackelberg equilibrium of MCA.

Theorem 3. *MSOPDS approaches differential Stackelberg equilibrium in MCA.*

Proof. Consider a differential Stackelberg equilibrium $\mathbf{X}^* \equiv (\mathbf{X}^{p*}, \mathbf{X}^{q*})$ such that $\frac{d\mathcal{L}^p(\mathbf{X}^*)}{d\mathbf{X}^p} = \frac{\partial \mathcal{L}^q(\mathbf{X}^*)}{\partial \mathbf{X}^q} = 0$, $\text{spec}(\frac{d^2\mathcal{L}^p(\mathbf{X}^*)}{d{\mathbf{X}^p}^2}) \subset \mathbb{R}^\circ_+$ and $\text{spec}(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial {\mathbf{X}^q}^2}) \subset \mathbb{R}^\circ_+$. Since $\det(\frac{\partial^2\mathcal{L}^q(\mathbf{X}^*)}{\partial {\mathbf{X}^q}^2}) \neq 0$ and $\frac{\partial}{\partial \mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) = 0$, by the implicit function theorem [14], there exists a neighborhood

$\mathcal{N}_1$ of $\mathbf{X}^p$ and a unique function $f : \mathcal{N}_1 \to \mathbb{R}^{|\mathcal{C}^q|}$ such that $\frac{\partial}{\partial \mathbf{X}^q}\mathcal{L}^q(\mathbf{X}^p, f(\mathbf{X}^p)) = 0 \ \forall \mathbf{X}^p \in \mathbb{R}^{|\mathcal{C}^p|}$ and $f(\mathbf{X}^{p*})) = \mathbf{X}^{q*}$.[7]

Due to the fact that eigenvalues vary continuously, there exists a neighborhood $\mathcal{N}_2$ of $\mathbf{X}^p$ where $\frac{d^2}{d\mathbf{X}^{p2}}\mathcal{L}^p(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$ and $\frac{\partial^2}{\partial \mathbf{X}^{q2}}\mathcal{L}^q(\mathbf{X}^{p*}, f(\mathbf{X}^{p*})) > 0$. Let $\mathcal{U}_1 \subseteq \mathcal{N}_1 \cap \mathcal{N}_2$ be the non-empty, open set whose closure is contained in $\mathcal{N}_1 \cap \mathcal{N}_2$. Since $\frac{\partial^2}{\partial \mathbf{X}^{q2}}\mathcal{L}^q(\mathbf{X}^{p*}, \mathbf{X}^{q*}) > 0$, there exists an open neighborhood $\mathcal{U}_2$ of $\mathbf{X}^{q*}$ such that $\frac{\partial^2}{\partial \mathbf{X}^{q2}}\mathcal{L}^q(\mathbf{X}^p, \mathbf{X}^q) > 0 \ \forall (\mathbf{X}^p, \mathbf{X}^q) \in \mathcal{U}_1 \times \mathcal{U}_2$. Since we have set the step-size of the attacker η^p and the subsequent opponent η^q to be $\eta^p < \eta^q$ in MSOPDS, according to (Lemma G.2 of [5]), the optimization process of MSOPDS on $(\mathbf{X}^p, \mathbf{X}^q)$ converges to $(\mathbf{X}^{p*}, \mathbf{X}^{q*})$ □

VI. Experiments

A. Experiment settings

1) Datasets and Threat Het-RecSys: We evaluate three real-world datasets, including **Ciao** [164] (2,611 users, 3,823 items, 44,453 ratings, and 49,953 links), **Epinion** [165] (1,929 users, 9,962 items, 12,612 ratings, and 41,270 links), and **LibraryThing** [124] (1,108 users, 8,583 items, 19,615 ratings, and 14508 links).[8] The links denote the number of edges in the social network $\mathcal{G}_U$. Following [7], the item graph $\mathcal{G}_I$ is created by connecting items that share over 50% of users that rated them in the rating record.

We experiment on a representative Het-RecSys, **ConsisRec** [7], which learns an embedding vector for each user and item node. To predict the rating of a user u on an item i, ConsisRec selectively aggregates the neighbor nodes on the heterogeneous graphs, i.e., the social network $\mathcal{G}_U$ and item graph $\mathcal{G}_I$, with an attention mechanism. Graph convolution operations are then performed over the aggregated nodes to obtain the final user and item embeddings. It then utilizes the dot product of the final embeddings for prediction and employs the Mean Square Error as the training loss in (1).

2) Demographic settings: We randomly select 5% of the user set $\mathcal{U}$ as the target audience $\mathcal{U}_{TA}$ and randomly sample 100 users from the user set $\mathcal{I}$ as the *customer base* $\mathcal{U}_{base}$ for each attacker and his opponents. We randomly select 50 items as the *competing items* $\mathcal{I}_{compete}$ and extract the item with the lowest average rating as the attacker's target item. Lastly, we randomly sample 100 items from non-competing items $\mathcal{I} \setminus \mathcal{I}_{compete}$ as the *company products* $\mathcal{I}_{product}$ for each attacker and his opponents.

3) Attacker's capability: We utilize a common budget parameter b, default to $b = 5$, to control attacker's capability under IA, CA, and MCA. In all attack scenarios, the attacker creates fake users that amount to b‰ of the user set $|\mathcal{U}|$, with each fake user giving a top rating (5-star) [166] to promote the target items i_t. In addition to the fake user ratings, each attack scenario employs a different set of poisoning actions as follows.

- Under **Injection Attack (IA)**, with capacity $\mathcal{C}_{IA}$, the attacker rates 100 filler items with each fake user.
- Under **Multiplayer Comprehensive Attack (MCA)**, with capacity $\mathcal{C}_{CA}$, the attacker hires N real users from his *customer base* $\mathcal{U}_{base}$ to give top ratings (5-star) to i_t, connects each fake account to N real users of his *customer base* and selects N items of his *company products* $\mathcal{I}_{product}$ to connect to i_t on the item graph $\mathcal{G}_I$, where $N = b \times 5‰|\mathcal{U}|$

Compared with IA, CA and MCA reduce the number of fake ratings and replace them with new social and item links. For instance, with $b = 3$ in the Ciao dataset, IA injects 900 additional ratings by fake accounts to filler items, while MCA create 15 ratings from real users, 35 social links, and 15 item connections. As shown later, MCA achieves better results than IA by employing multiple categories of poisoning actions and properly anticipating the opponents' actions.

4) Opponent goal and capability: We assume the opponents conduct a simplified Comprehensive Attack with budget $b_{op} = 2$.[9] In particular, each opponent selects real users from his customer base by BOPDS to all give a 1-star rating to the attacker's target item i_t, such that the target item is demoted among the competing items $\mathcal{I}_{compete}$.

5) Compared baseline methods: We compare the proposed method with several baselines under the parameters set according to the original papers. Note that **MSOPDS** is utilized under MCA, which anticipates subsequent attacks by opponents. On the other hand, the compared baseline attacks are all operated under the IA setting.[10]

- **None**: The attacker conducts no attack.
- **Random**: The attacker selects proxy items randomly.
- **Popular** [52], [168]: The attacker selects 90% random and 10% popular items as the proxy items.
- **Projected gradient ascent attack (PGA)** [17]: The attacker optimizes the poisoning attack over matrix factorization-based recommendation models.
- ***S*-attack** [151]: *S*-attack formulates the selection of proxy items as an optimization problem and targets on a graph-based recommender model. Each proxy item is rated with a normal distribution that matches the real user ratings.
- **Revisit Attack (RevAdv)** [2]: The state-of-the-art bilevel optimization approach with the gradient calculation conducted over the process of RecSys training over the generated poison data.
- **Trial Attack** [16]: The attack utilizes triple adversarial learning by extending the traditional GAN [169] into a generator that creates poison data similar to the normal ratings, a discriminator differentiating real and fake ratings, and an influence module appraising the effectiveness of the poison data (for the attacker's objective).

[7] Empirically, the numerical values of the total derivative $\frac{d\mathcal{L}^p}{d\mathbf{X}^p}$ and the partial derivative $\frac{\partial \mathcal{L}^q}{\partial \mathbf{X}^q}$ are all within reasonable range throughout the MSOPDS iterations, implying that they are L_1 and L_2-Lipschitz, respectively.

[8] We employ the standard preprocessing step to include only the users with at least 15 friends and at least 1 item rating.

[9] We further evaluate the effect to opponent capability b_{op} in Section VI-D.

[10] For methods designed for implicit ratings, we transform rating above three stars as positive [167]. If rating for filler items are not specified, we follow [52] to produce ratings from normal distribution fitted to the true ratings.

TABLE IV: Attacker's target item average predicted rating ($\bar{r}$) and Hit Rate@3 (HR@3) on ConsisRec facing a single opponent.

Dataset	Type	Method	$b=2$		$b=3$		$b=4$		$b=5$	
			$\bar{r}$	HR@3	$\bar{r}$	HR@3	$\bar{r}$	HR@3	$\bar{r}$	HR@3
Ciao	IA	None	2.1282	0.0154	2.1282	0.0154	2.1282	0.0154	2.1282	0.0154
		Random	2.6881	0.0692	2.6554	0.0538	2.8411	0.1154	2.8222	0.1000
		Popular	2.5508	0.0769	2.6948	0.0923	2.9072	0.1231	2.9403	0.1154
		PGA	2.5395	0.0615	2.5424	0.0538	2.7103	0.1000	2.8631	0.1154
		S-attack	2.6180	0.0846	2.8557	0.1231	2.7874	0.0923	2.9313	0.1154
		RevAdv	2.5250	0.0692	2.7005	0.0923	2.6618	0.0769	2.9862	0.1231
		Trial	2.6576	0.0769	2.5311	0.0538	2.8694	0.1000	3.0627	0.1462
	MCA	MSODS	**3.2100**	**0.1615**	**3.2177**	**0.1846**	**3.2022**	**0.1692**	**3.3953**	**0.2154**
Epinions	IA	None	1.2938	0.0000	1.2938	0.0000	1.2938	0.0000	1.2938	0.0000
		Random	1.9492	0.0000	2.6070	0.0104	2.4007	0.0104	2.5166	0.0208
		Popular	1.7655	0.0000	2.2960	0.0000	2.5757	0.0208	2.6094	0.0312
		PGA	1.9493	0.0000	2.3111	0.0000	2.3387	0.0000	2.4283	0.0000
		S-attack	1.8089	0.0000	2.2274	0.0000	2.3190	0.0208	2.5093	0.0312
		RevAdv	1.7721	0.0000	2.3980	0.0104	2.4116	0.0000	2.3946	0.0000
		Trial	1.8090	0.0000	2.3227	0.0000	2.3234	0.0000	2.4525	0.0104
	MCA	MSODS	**2.9252**	**0.1875**	**3.1662**	**0.2083**	**3.3865**	**0.2292**	**3.2414**	**0.1979**
library	IA	None	1.7001	0.0000	1.7001	0.0000	1.7001	0.0000	1.7001	0.0000
		Random	1.9865	0.0213	2.0482	0.0106	2.1590	0.0106	2.4537	0.0638
		Popular	2.0421	0.0213	2.1894	0.0426	2.2663	0.0426	2.3879	0.0426
		PGA	2.0316	0.0213	2.1229	0.0319	2.3192	0.0426	2.3629	0.0426
		S-attack	2.0486	0.0213	2.2473	0.0638	2.3050	0.0426	2.3725	0.0319
		RevAdv	1.9276	0.0213	2.0768	0.0213	2.3096	0.0319	2.4007	0.0532
		Trial	1.9687	0.0213	2.0220	0.0319	2.3881	0.0532	2.2885	0.0426
	MCA	MSODS	**2.7410**	**0.2234**	**3.0174**	**0.3085**	**3.0541**	**0.3830**	**3.0420**	**0.3085**

6) Evaluation Metrics: We inspect the following metrics.

- **Average predicted rating** ($\bar{r}$): We average the final ratings of the attacker's target item predicted by the poisoned Het-RecSys for each target user $\mathcal{U}_{TA}$.
- **HitRate at** 3 (**HR@3**): The ratio of target users $\mathcal{U}_{TA}$ for whom the attacker's target item is ranked by the poisoned Het-RecSys in the top 3 positions among the 50 competing items $\mathcal{I}_{compete}$ [16].

7) Hyperparameter setting: Corresponding to Algorithm 1, we set $\eta^p = 0.005$, $\eta^q = 0.05$, inner RecSys training loop number $L = 5$, and outer MSO loop number $K = 20$.

B. Result against a single opponent

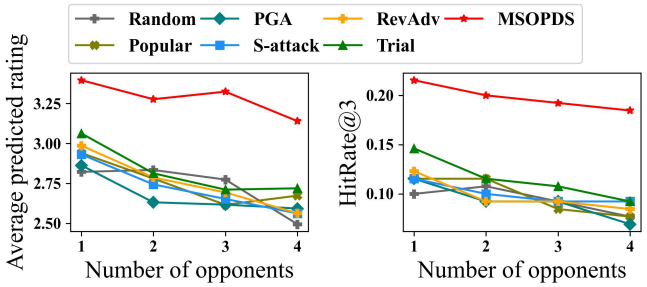
First, we evaluate MSOPDS with one attacker (leader) facing one opponent (follower), where the attacker first poisons the Het-RecSys given the original real ratings and ancillary data. Then, the opponent determines his poisoning actions (by planning a Comprehensive Attack) given both the real data and the attacker's actions (i.e., poisoning modifications on the graphs). The final prediction result is determined by the Het-RecSys trained with the original data and poison data of the attacker and opponent.

Table IV summarizes the results of different poisoning methods against ConsisRec [7] under different budget constraints $b \in [2, 5]$. As shown in Table IV, the proposed MSOPDS under MCA yields the best performance in all budget settings and all three datasets, outperforming baseline methods by a significant margin. In particular, MSOPDS surpasses the second best method by up to 10.6% in terms of the average predicted rating. It also consistently outperforms the second best method in terms of the hit rate by up to 11.4%. We observe that Epinions and LibraryThing present larger difference between MSOPDS and the baselines than Ciao, since Ciao has a sparser social network for the poisoning effect to propagate.

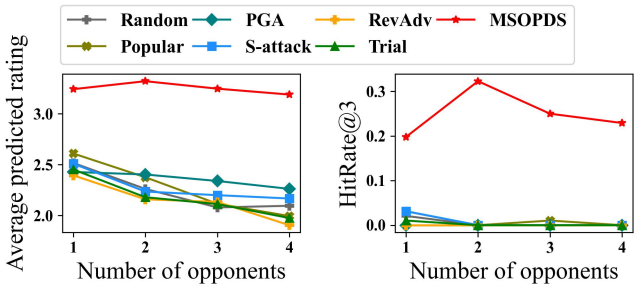
Apart from MSOPDS, all compared methods have weaker performance since they do not anticipate the poisoning attacks by opponents. Among the baselines, *PGA, S-attack, Revisit Attack*, and *Trial Attack* all plan the data poisoning attack through a bilevel optimization. However, since the above baseline and ablation methods are not designed to face the negative impact of the poisoning attacks by the opponent who acts after the attacker, these approaches suffer from bad performances and may even fall behind heuristic methods (Random Attack and Popular Attack). In contrast, the proposed MSOPDS integrates a Multilevel Stackelberg Optimization framework with Progressive Differentiable Surrogate to anticipate and evaluate the potential effects of hostile actions by the opponent. The potential opponent's actions evaluated within the Stackelberg framework play an essential role in the effective enhancement the robustness of MSOPDS.

C. Impact of the number of opponents.

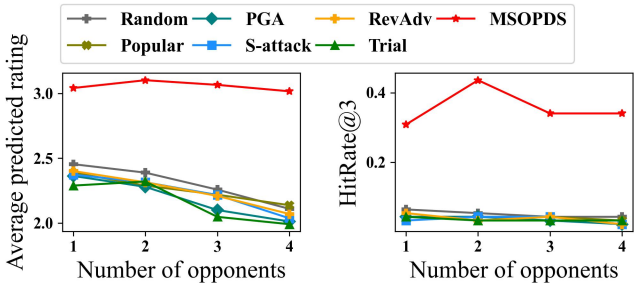
Fig. 6 compares the performance of different methods under various numbers of opponents. The opponents are assumed to sequentially conduct their respective data poisoning attacks using Comprehensive Attack, where each opponent selects real users from their customer base by BOPDS to give a 1-star rat-



(a) Results on the Ciao dataset.

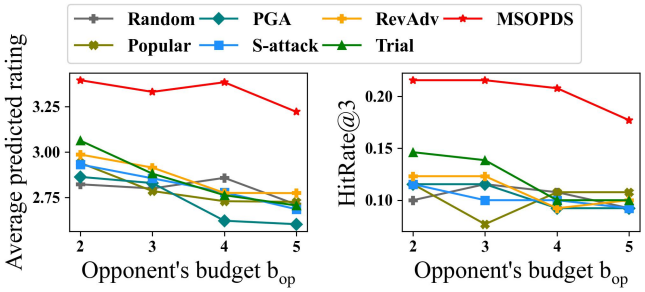


(b) Results on the Epinions dataset.

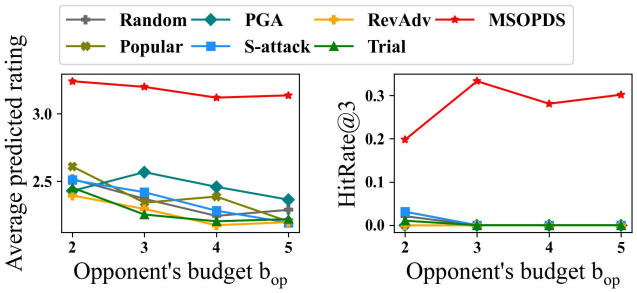


(c) Results on the LibraryThing dataset.

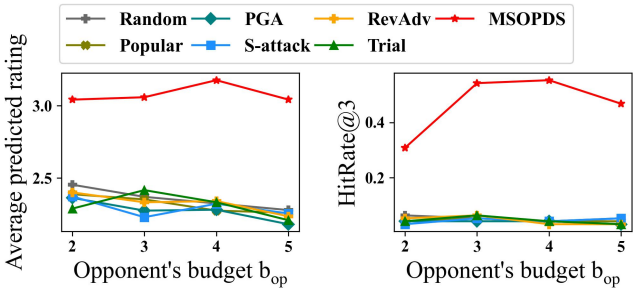
Fig. 6: Impact of the number of opponents.



(a) Results on the Ciao dataset.



(b) Results on the Epinions dataset.



(c) Results on the LibraryThing dataset.

Fig. 7: Impact of the opponent's capacity.

ing to the target items. While all baselines do not change their attacks as the number of opponents increases (identical to the single-opponent setting in Table IV), MSOPDS carefully plans the optimal poisoning attack by anticipating the opponents' actions via Multilevel Stackelberg Optimization. As a result, the performance of our proposed MSOPDS method under MCA consistently outperforms all compared baseline methods. Although the performance of baseline methods decreases drastically as the number of opponents increases, MSOPDS achieves smaller reductions in both evaluation metrics. In the Epinions dataset, MSOPDS maintains positive HitRate@3 scores, while other methods all drop to 0, manifesting the importance of modeling a multiplayer game for data poisoning attacks against RecSys.

D. Impact of the opponent's capacity

Fig. 7 compares the performance of an attacker facing a single opponent with varied budgets. While increasing the opponent's budget *reduces* the attacker's performance, MSOPDS under MCA outperforms all baselines. MSOPDS suffers from a smaller degradation in performance, since it anticipates the opponent's poisoning action plans and adjusts the attacker's poisoning action plan accordingly. Besides, the performances of the attacker's poisoning attacks in Epinions and LibraryThing are also more sensitive to the change of the opponent capacity than Ciao, since these two datasets have a sparser rating record than Ciao.

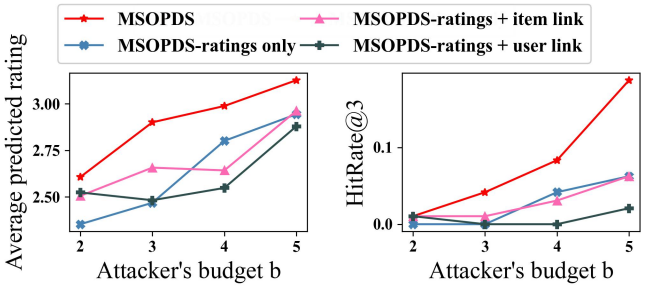


Fig. 8: Effect of different categories of poisoning actions in the Epinions dataset.

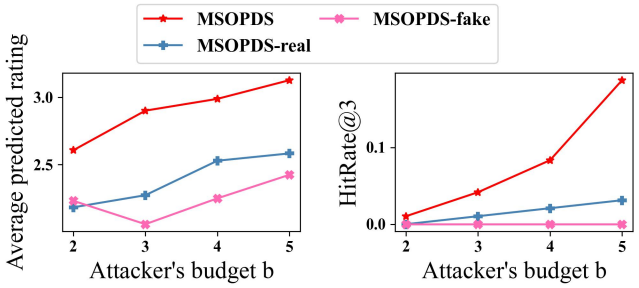


Fig. 9: Effect of creating ratings with real users and injecting ratings with fake users in the Epinions dataset.

E. Effect of different categories of poisoning actions

Fig. 8 compares the effectiveness of different categories of poisoning actions, with three ablation methods, including **1) MSOPDS-ratings only**, where the attacker only hires real users for ratings and does not poison the social network and the item graph, **2) MSOPDS-ratings+item link**, which poisons the rating records and item graph, and **3) MSOPDS-ratings+user link**, which poisons the rating records and social network. Here, we first evaluate the Epinions dataset since Epinions is more sensitive to the changes in the attack procedure and settings. As shown in Fig. 8, MSOPDS obtains the best results by carefully examining all poisoning actions. In contrast, the ablation methods ratings+item and ratings+social both suffer a performance degradation. This is because HetRecSys utilizes the dot product between the final user and item embeddings to determine the rating predictions [7]. Thus, poisoning only the user or only the item can merely influence the result partially. While the final user and item embedding involve a graph convolution process along the social network and the item graph, the convolution process dilutes the poisoning effects of modifying edges in the corresponding graph, because the operator normalizes the neighborhood embeddings by the node degree in (15). Thus, jointly poisoning the social and item graph obtains a compound effect in the proposed MSOPDS. Lastly, attacking the item graph is more effective than attacking the social network. The reason is that item graph poisoning directly influences the target item's final embedding, while poisoning the social network indirectly affects the target users through creating social links between the users from the customer base $\mathcal{U}_{base}$ and the injected fake accounts.

F. Effect of poisoning by real users and fake accounts

Fig. 9 compares the effectiveness of data poisoning by real users and fake accounts with two ablation methods, including **1) MSOPDS-real**, where the attacker only hires real users, and **2) MSOPDS-fake**, where the attacker only injects the fake accounts. For a fair comparison, all methods exclude the poisoning actions against the item graph. MSOPDS obtains the best results since it carefully examines all types of poisoning actions. Besides, MSOPDS-real performs *better* than MSOPDS-fake since real users have more substantial connections in the social network. In contrast, fake users are only connected to selected real users from the customer base and thus have limited social influence. As website moderators usually detect and remove fake user accounts [170], conducting poisoning actions via real users may work better.

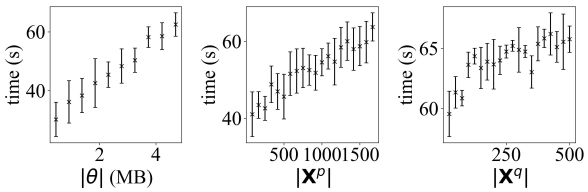


Fig. 10: Running time measurements on $|\theta|$, $|\mathbf{X}^p|$, and $|\mathbf{X}^q|$.

G. Running time measurements

We validate Theorem 1 by measuring the time to perform 1-iteration of MSOPDS (i.e., 1- iteration of the outer loop in Algorithm 1) for varying size of the PDS model parameter $|\theta|$, attacker importance vector $|\mathbf{X}^p|$, and opponent importance vector $|\mathbf{X}^q|$. In particular, the size of $|\theta|$ is varied by changing the dimension d of the user and item embeddings $\mathbf{h_u}, \mathbf{h_i} \in \mathbb{R}^{\mathbf{d}}$, while the sizes of $\mathbf{X}^p$ and $\mathbf{X}^q$ are varied by changing the sizes of the $\mathcal{U}_{base}$ and company products $\mathcal{U}_{base}$. As shown in Fig. 10, we observe that the relation follows a linear trend in all three cases, verifying Theorem 1

H. Acceleration techniques

Here, we report the evaluation results and the total running time measurements of the three acceleration techniques presented in Section IV-E. The three techniques include **Competitor-aware Adaptive Gradient Reuse (reuse)** which reuses the Stackelberg correction term every two iterations, **Stochastic Poisoning Action Omission (omis)**, which narrows the updated elements in $\mathbf{X}^p$ to those with values ranked in the middle half after the initial iterations, and **Progressive Attack Influence Propagation (prop)**, which starts from the first hop neighbors of the the terminal nodes of the poisoning actions, than progressively include the next-hop neighbors on $\mathcal{G}_U$ and $\mathcal{G}_I$ into the MSOPDS process until reaching the full heterogeneous graph. As shown in Figure 11, we observe that all three acceleration techniques trades effectiveness for efficiency in MSOPDS. In particular, as the acceleration techniques achieve suboptimal results but have lower running time, these techniques may be applicable to use cases that are required to operate with less computation resources.

VII. Conclusion

In this paper, we investigate the problem of planning a data poisoning attack against recommender systems formulated as a multiplayer Stackelberg game, where an attacker anticipates opponents' attack plans to secure an attack plan with good data poisoning results. Particularly, we target on the sophisticated Heterogeneous RecSys, which utilizes rich contextual data, including rating records, social networks, and item graphs.

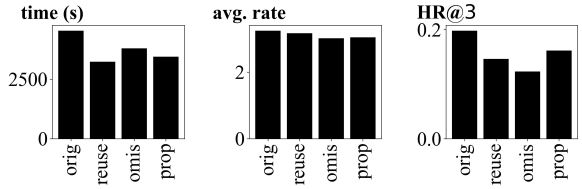


Fig. 11: Run time (time), average rating prediction (avg. rate) and Hitrate@3 (HR@3) comparing the original method (orig) and the three acceleration methods (reuse, omis, and prop). We find that all three acceleration techniques trades effectiveness for efficiency.

We present the MSOPDS, consisting of a Multilevel Stackelberg Optimization framework and a Progressive Differentiable Surrogate model, to find the equilibrium of a multiplayer Stackelberg game and obtain the optimal attack plan for three categories of poisoning actions. Extensive experiments demonstrate the effectiveness of MSOPDS in planning a Multiplayer Comprehensive Attack under both single-opponent and multiple-opponent settings.

REFERENCES

[1] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Tang, H. Wen, and K. Wang, "Revisiting adversarially learned injection attacks against recommender systems," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 318–327.

[3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[4] J. R. Shewchuk *et al.*, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[5] T. Fiez, B. Chasnov, and L. Ratliff, "Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study," in *ICML*. PMLR, 2020, pp. 3133–3144.

[6] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *ACM SIGIR*, 2020, pp. 639–648.

[7] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, "Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation," in *ACM SIGIR*, 2021, pp. 2141–2145.

[8] H. Von Stackelberg, *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[9] Y. Zhang, S. Gao, J. Pei, and H. Huang, "Improving social network embedding via new second-order continuous graph neural networks," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2515–2523.

[10] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 623–632.

[11] P. Awasthi, A. Das, and S. Gollapudi, "A convergence analysis of gradient descent on graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 385–20 397, 2021.

[12] B. T. Polyak, "Gradient methods for solving equations and inequalities," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 6, pp. 17–32, 1964.

[13] B. Bošanskỳ, S. Brânzei, K. A. Hansen, T. B. Lund, and P. B. Miltersen, "Computation of stackelberg equilibria of finite sequential games," *ACM Transactions on Economics and Computation (TEAC)*, vol. 5, no. 4, pp. 1–24, 2017.

[14] R. Abraham, J. E. Marsden, and T. Ratiu, *Manifolds, tensor analysis, and applications*. Springer Science & Business Media, 2012, vol. 75.

[15] S. W. Schmitz and M. Latzer, "Competition in b2c e-commerce: analytical issues and empirical evidence," *Electronic Markets*, vol. 12, no. 3, pp. 163–174, 2002.

[16] C. Wu, D. Lian, Y. Ge, Z. Zhu, and E. Chen, "Triple adversarial learning for influence based poisoning attack in recommender systems," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1830–1840.

[17] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," *Advances in neural information processing systems*, vol. 29, pp. 1885–1893, 2016.

[18] L. Haoyang, S. Di, and L. Chen, "Revisiting injective attacks on recommender systems," in *Advances in Neural Information Processing Systems*.

[19] J. Song, Z. Li, Z. Hu, Y. Wu, Z. Li, J. Li, and J. Gao, "Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 157–168.

[20] H. Zhang, C. Tian, Y. Li, L. Su, N. Yang, W. X. Zhao, and J. Gao, "Data poisoning attack against recommender system using incomplete and perturbed data," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2154–2164.

[21] M. Egele, C. Kolbitsch, and C. Platzer, "Removing web spam links from search engine results," *Journal in Computer Virology*, vol. 7, no. 1, pp. 51–62, 2011.

[22] A. Kieler. (2015, apr) Amazon files first lawsuit to block companies from selling fraudulent positive reviews. [Online]. Available: https://consumerist.com/2015/04/09/amazon-files-first-lawsuit-to-block-companies-from-selling-fraudulent-positive-reviews/

[23] J. Peters. (2021, may) Amazon-first gadget brands aukey and mpow are suddenly, suspiciously disappearing. [Online]. Available: https://www.theverge.com/2021/5/10/22428858/amazon-aukey-mpow-listings-disappearing

[24] S. Hollister. (2021, jun) Amazon confirms it removed ravpower, a popular phone battery and charger brand. [Online]. Available: https://www.theverge.com/2021/6/16/22536976/amazon-ravpower-battery-charger-removed-amazon

[25] ——. (2021, jun) Not just ravpower - amazon has yanked vava and taotronics, too. [Online]. Available: https://www.theverge.com/2021/6/17/22538779/amazon-sunvalley-tek-ravpower-vava-taotronics-pulled-store

[26] ——. (2021, jul) Another amazon-first gadget brand has suspiciously vanished: Choetech. [Online]. Available: https://www.theverge.com/2021/7/7/22567530/amazon-choetech-removed-pulled-aukey-mpow

[27] ——. (2021, sep) Amazon says it's permanently banned 600 chinese brands for review fraud. [Online]. Available: https://www.theverge.com/2021/9/17/22680269/amazon-ban-chinese-brands-review-abuse-fraud-policy

[28] L. Plaugic. (2015, feb) Yelp is suing a company for allegedly selling fake positive reviews to restaurants. [Online]. Available: https://www.theverge.com/2015/2/20/8078147/yelp-suing-company-for-fake-reviews

[29] H. D. Canavan. (2013, sep) 16% of yelp restaurant reviews are fake, study says. [Online]. Available: https://www.eater.com/2013/9/26/6364287/16-of-yelp-restaurant-reviews-are-fake-study-says

[30] R. Albergotti and C. Alcantara. (2021, jun) Apple's tightly controlled app store is teeming with scams. [Online]. Available: https://www.washingtonpost.com/technology/2021/06/06/apple-app-store-scams-fraud/

[31] E. Dwoskin and C. Timberg. (2018, apr) How merchants use facebook to flood amazon with fake reviews. [Online]. Available: https://www.washingtonpost.com/business/economy/how-merchants-secretly-use-facebook-to-flood-amazon-with-fake-reviews/2018/04/23/5dad1e30-4392-11e8-8569-26fda6b404c7_story.html

[32] D. Deahl. (2018, feb) Huawei got people to write fake reviews for an unreleased phone. [Online]. Available: https://www.theverge.com/2018/2/12/17005798/huawei-fake-reviews-best-buy-mate-10-pro-phone

[33] J. Hicks. (2022, aug) Roomster sued by the ftc and six state ags for fake listings and fake app store reviews. [Online]. Available: https://www.theverge.com/2022/8/31/23330808/roomster-ftc-lawsuit-app-store-google-play-fake-reviews

[34] B. News. (2022, jun) Sony admits using fake reviewer. [Online]. Available: http://news.bbc.co.uk/2/hi/entertainment/1368666.stm

[35] Y. Lu, R. Xie, C. Shi, Y. Fang, W. Wang, X. Zhang, and L. Lin, "Social influence attentive neural network for friend-enhanced recom-

mendation," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2021, pp. 3–18.

[36] H. Liu, C. Zheng, D. Li, Z. Zhang, K. Lin, X. Shen, N. N. Xiong, and J. Wang, "Multi-perspective social recommendation method with graph representation learning," *Neurocomputing*, vol. 468, pp. 469–481, 2022.

[37] Y. S. Centor. (2022, jan) How do i add friends on yelp? [Online]. Available: https://www.yelp-support.com/article/How-do-I-add-friends-on-Yelp?l=en_US

[38] N. Lee. (2016, oct) Facebook's friend-based recommendations take on yelp. [Online]. Available: https://www.engadget.com/2016-10-19-facebook-recommendations.html

[39] I. Paul. (2016, oct) Facebook wants to replace yelp with your friends' recommendations. [Online]. Available: https://www.pcworld.com/article/410737/facebook-wants-to-replace-yelp-with-your-friends-recommendations.html

[40] G. Zaratti. (2021, apr) How pinterest uses machine learning to provide tailored recommendations to million of users worldwide. [Online]. Available: https://d3.harvard.edu/platform-digit/submission/how-pinterest-uses-machine-learning-to-provide-tailored-recommendations-to-million-of-users-worldwide/

[41] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *ACM SIGIR*, 2015, pp. 43–52.

[42] Y. Zhang, H. Lu, W. Niu, and J. Caverlee, "Quality-aware neural complementary item recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 77–85.

[43] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 785–794.

[44] W.-C. Kang, M. Wan, and J. McAuley, "Recommendation through mixtures of heterogeneous item relationships," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1143–1152.

[45] F. Lardinois. (2018, jun) The new google maps with personalized recommendations is now live. [Online]. Available: https://www.yelp-support.com/article/How-do-I-add-friends-on-Yelp?l=en_US

[46] N. H. Center. (2023, jan) How netflix's recommendations system works. [Online]. Available: https://help.netflix.com/en/node/100639

[47] S. Gibbons. (2022, dec) 3 e-commerce trends to watch in 2023. [Online]. Available: https://www.forbes.com/sites/serenitygibbons/2022/12/22/3-e-commerce-trends-to-watch-in-2023/?sh=4e8be85f5575

[48] Z. Nan. (2022, dec) Social commerce raises revenues. [Online]. Available: https://www.chinadaily.com.cn/a/202212/19/WS639fc1c3a31057c47eba5092.html

[49] C. Becdach. (2022, oct) Social commerce: The future of how consumers interact with brands. [Online]. Available: https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/social-commerce-the-future-of-how-consumers-interact-with-brands

[50] G. Llewellyn. (2019, nov) Social commerce trends for 2020 you need to look out for. [Online]. Available: https://www.smartinsights.com/ecommerce/social-commerce/social-commerce-trends-for-2020-you-need-to-look-out-for/

[51] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, vol. 26, 2013.

[52] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 381–392.

[53] S. E. Whang and J.-G. Lee, "Data collection and quality challenges for deep learning," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3429–3432, 2020.

[54] A. Balayn, C. Lofi, and G.-J. Houben, "Managing bias and unfairness in data for decision support: a survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems," *The VLDB Journal*, vol. 30, no. 5, pp. 739–768, 2021.

[55] E. Pitoura, K. Stefanidis, and G. Koutrika, "Fairness in rankings and recommenders: Models, methods and research directions," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2358–2361.

[56] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis, "Muve: Efficient multi-objective view recommendation for visual data exploration," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 731–742.

[57] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[58] A. Davoudi, "Effects of user interactions on online social recommender systems," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 1444–1448.

[59] H. Lim, Y.-C. Lee, J.-S. Lee, S. Han, S. Kim, Y. Jeong, C. Kim, J. Kim, S. Han, S. Choi *et al.*, "Airs: a large-scale recommender system at naver news," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3386–3398.

[60] P. Banerjee, L. Chu, Y. Zhang, L. V. Lakshmanan, and L. Wang, "Stealthy targeted data poisoning attack on knowledge graphs," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2069–2074.

[61] W. Fan, T. Derr, X. Zhao, Y. Ma, H. Liu, J. Wang, J. Tang, and Q. Li, "Attacking black-box recommendations via copying cross-domain user profiles," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1583–1594.

[62] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Ren, and K. Zhou, "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 14–26.

[63] Y. Lao, P. Yang, W. Zhao, and P. Li, "Identification for deep neural network: Simply adjusting few weights!" in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1328–1341.

[64] H. Li, S. Di, Z. Li, L. Chen, and J. Cao, "Black-box adversarial attack and defense on graph neural networks," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1017–1030.

[65] D. Ding, M. Zhang, Y. Huang, X. Pan, F. Feng, E. Jiang, and M. Yang, "Towards backdoor attack on deep learning based time series classification," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1274–1287.

[66] Y. Liu, Z. Liu, X. Feng, and Z. Li, "Robust attributed network embedding preserving community information," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1874–1886.

[67] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 181–192.

[68] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "Air: Attentional intention-aware recommender systems," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 304–315.

[69] K.-J. Cho, Y.-C. Lee, K. Han, J. Choi, and S.-W. Kim, "No, that's not my feedback: Tv show recommendation using watchable interval," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 316–327.

[70] Z. Zolaktaf, R. Babanezhad, and R. Pottinger, "A generic top-n recommendation framework for trading-off accuracy, novelty, and coverage," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 149–160.

[71] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in netease," in *2016 IEEE 32nd international conference on data engineering (ICDE)*. IEEE, 2016, pp. 1218–1229.

[72] Y. Tan, C. Yang, X. Wei, Y. Ma, and X. Zheng, "Multi-facet recommender networks with spherical optimization," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1524–1535.

[73] Z. Chen, T. Xiao, and K. Kuang, "Ba-gnn: On learning bias-aware graph neural network," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3012–3024.

[74] Z. Li, X. Shen, Y. Jiao, X. Pan, P. Zou, X. Meng, C. Yao, and J. Bu, "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1677–1688.

[75] Z. Huan, Y. Quanming, and T. Weiwei, "Search to aggregate neighborhood for graph neural network," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 552–563.

[76] Z. Wang, T. Xia, R. Jiang, X. Liu, K.-S. Kim, X. Song, and R. Shibasaki, "Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1751–1762.

[77] B. Ye, S. Yang, B. Hu, Z. Zhang, Y. He, K. Huang, J. Zhou, and Y. Fang, "Gaia: Graph neural network with temporal shift aware attention for gross merchandise value forecast in e-commerce," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3320–3326.

[78] J. Zhang, B. Dong, and S. Y. Philip, "Fakedetector: Effective fake news detection with deep diffusive neural network," in *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 2020, pp. 1826–1829.

[79] Y. Zhao, K. Zheng, J. Guo, B. Yang, T. B. Pedersen, and C. S. Jensen, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 265–276.

[80] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1442–1453.

[81] C. Yan, B. Li, Y. Vorobeychik, A. Laszka, D. Fabbri, and B. Malin, "Get your workload in order: Game theoretic prioritization of database auditing," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1304–1307.

[82] H.-S. Lim, G. Ghinita, E. Bertino, and M. Kantarcioglu, "A game-theoretic approach for high-assurance of data trustworthiness in sensor networks," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 1192–1203.

[83] Z. Liu, Z. Zhou, and T. Rekatsinas, "Picket: guarding against corrupted data in tabular data during learning and inference," *The VLDB Journal*, vol. 31, no. 5, pp. 927–955, 2022.

[84] J. J. Levandoski, M. D. Ekstrand, M. J. Ludwig, A. Eldawy, M. F. Mokbel, and J. T. Riedl, "Recbench: benchmarks for evaluating performance of recommender system architectures," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 911–920, 2011.

[85] Y. Liu, T.-A. N. Pham, G. Cong, and Q. Yuan, "An experimental evaluation of point-of-interest recommendation in location-based social networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1010–1021, 2017.

[86] J. Gao, J. Chen, Z. Li, and J. Zhang, "Ics-gnn: lightweight interactive community search via graph neural network," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 1006–1018, 2021.

[87] Y. Jiang, Y. Rong, H. Cheng, X. Huang, K. Zhao, and J. Huang, "Query driven-graph neural networks for community search: from non-attributed, attributed, to interactive attributed," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1243–1255, 2022.

[88] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Heterogeneous information networks: the past, the present, and the future," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3807–3811, 2022.

[89] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, 2020.

[90] Z. Chen, P. Tian, W. Liao, and W. Yu, "Zero knowledge clustering based adversarial mitigation in heterogeneous federated learning," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1070–1083, 2020.

[91] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3412–3425, 2021.

[92] H. Zhang, Y. Li, B. Ding, and J. Gao, "Loki: A practical data poisoning attack framework against next item recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[93] H. Xia, S. Shao, C. Hu, R. Zhang, T. Qiu, and F. Xiao, "Robust clustering model based on attention mechanism and graph convolutional network," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[94] J. Wu and J. He, "A unified framework for adversarial attacks on multi-source domain adaptation," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[95] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2013.

[96] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, S. Y. Philip, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[97] Z. Zhang, Q. Liu, Z. Huang, H. Wang, C.-K. Lee, and E. Chen, "Model inversion attacks against graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[98] I.-C. Hsieh and C.-T. Li, "Netfense: Adversarial defenses against privacy attacks on neural networks for graph data," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[99] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, X. Wang, W. Zhu, and J. Huang, "Adversarial attack framework on graph embedding models with limited knowledge," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[100] W. Yu, X. Lin, J. Liu, J. Ge, W. Ou, and Z. Qin, "Self-propagation graph neural network for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[101] M. Zhang, S. Wu, M. Gao, X. Jiang, K. Xu, and L. Wang, "Personalized graph neural networks with attention mechanism for session-aware recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[102] Y. Liu, S. Yang, Y. Xu, C. Miao, M. Wu, and J. Zhang, "Contextualized graph attention network for recommendation with item knowledge graph," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[103] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[104] T. Qian, Y. Liang, Q. Li, and H. Xiong, "Attribute graph neural networks for strict cold start recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[105] W. Wang, W. Zhang, S. Liu, Q. Liu, B. Zhang, L. Lin, and H. Zha, "Incorporating link prediction into multi-relational item graph modeling for session-based recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[106] Y. Liu, C. Liang, X. He, J. Peng, Z. Zheng, and J. Tang, "Modelling high-order social relations for item recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[107] P. Qiao, Z. Zhang, Z. Li, Y. Zhang, K. Bian, Y. Li, and G. Wang, "Tag: Joint triple-hierarchical attention and gcn for review-based social recommender system," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[108] Y. Zhu, Q. Lin, H. Lu, K. Shi, D. Liu, J. Chambua, S. Wan, and Z. Niu, "Recommending learning objects through attentive heterogeneous graph convolution and operation-aware neural network," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[109] Y. Liu, S. Yang, Y. Zhang, C. Miao, Z. Nie, and J. Zhang, "Learning hierarchical review graph representations for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[110] J. Chang, C. Gao, X. He, D. Jin, and Y. Li, "Bundle recommendation and generation with graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[111] N. Li, C. Gao, D. Jin, and Q. Liao, "Disentangled modeling of social homophily and influence for social recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[112] K. McCormick. (2022, apr) 22 ways to ask for reviews (with copy/paste templates). [Online]. Available: https://www.wordstream.com/blog/ws/2020/07/16/how-to-ask-for-reviews

[113] S. Bernazzani. (2022, jan) How to ask & get good customer reviews [+examples]. [Online]. Available: https://blog.hubspot.com/service/get-customer-reviews

[114] J. Wellemeyer. (2019, sep) This 23-year-old has made $120,000 buying and selling instagram accounts. [Online]. Available: https://www.marketwatch.com/story/this-23-year-old-has-made-120000-buying-and-selling-instagram-accounts-2019-08-12

[115] H. Britzky. (2018, jan) Nearly 50 million fake twitter accounts are being sold to real users. [Online]. Available: https://www.axios.com/2018/01/27/fake-twitter-accounts-sold-for-more-followers

[116] V. Leasure. (2022, dec) Hacker charged for attempting to make a purchase on amazon account. [Online]. Available: https://www.wxii12.com/article/alamance-hack-amazon-account-fraud-felony/42297990

[117] D. Ahmed. (2022, nov) 34 russian hacking groups stole 50 million user passwords. [Online]. Available: https://www.hackread.com/russian-hacking-groups-user-passwords/

[118] B. M. Marlin, "Modeling user rating profiles for collaborative filtering," *Advances in neural information processing systems*, vol. 16, 2003.

[119] C. Liu, C. Gao, Y. Yuan, C. Bai, L. Luo, X. Du, X. Shi, H. Luo, D. Jin, and Y. Li, "Modeling persuasion factor of user decision for recommendation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3366–3376.

[120] E. J. Horvitz, J. S. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The lumiere project: Bayesian user modeling for inferring the goals and needs of software users," *arXiv preprint arXiv:1301.7385*, 2013.

[121] A. Parsons, "Using social media to reach consumers: A content analysis of official facebook pages," *Academy of marketing studies Journal*, vol. 17, no. 2, p. 27, 2013.

[122] W. Chen, W. Hsu, and M. L. Lee, "Making recommendations from multiple domains," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 892–900.

[123] M. S. Pera, N. Condie, and Y.-K. Ng, "Personalized book recommendations created by using social media data," in *International conference on web information systems engineering*. Springer, 2011, pp. 390–403.

[124] T. Zhao, J. McAuley, and I. King, "Improving latent factor models via personalized feature projection for one class recommendation," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 821–830.

[125] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.

[126] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *ICML*. PMLR, 2018, pp. 1115–1124.

[127] J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, and J. Huang, "Adversarial attack on community detection by hiding individuals," in *Proceedings of The Web Conference 2020*, 2020, pp. 917–927.

[128] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *ICML*. PMLR, 2019, pp. 695–704.

[129] X. Liu, S. Si, J. Zhu, Y. Li, and C.-J. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[130] A. Jameson, M. C. Willemsen, A. Felfernig, M. d. Gemmis, P. Lops, G. Semeraro, and L. Chen, "Human decision making and recommender systems," in *Recommender systems handbook*. Springer, 2015, pp. 611–648.

[131] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.

[132] M. Luca and G. Zervas, "Fake it till you make it: Reputation, competition, and yelp review fraud," *Management Science*, vol. 62, no. 12, pp. 3412–3427, 2016.

[133] J. Stempel. (2015, apr) Amazon sues to block alleged fake reviews on its website. [Online]. Available: https://www.reuters.com/article/us-amazon-com-lawsuit-fake-reviews-idUSKBN0N02LP20150410

[134] Y. Wu, E. W. Ngai, P. Wu, and C. Wu, "Fake online reviews: Literature review, synthesis, and directions for future research," *Decision Support Systems*, vol. 132, p. 113280, 2020.

[135] J. Hawkins. (2022, jul) 1-star review attacks plague restaurants on google. [Online]. Available: https://searchengineland.com/google-1-star-restaurant-reviews-scam-386561

[136] R. Erskine. (2022, may) You just got attacked by fake 1-star reviews. now what? [Online]. Available: https://www.forbes.com/sites/ryanerskine/2018/05/15/you-just-got-attacked-by-fake-1-star-reviews-now-what/?sh=cffe6e610715

[137] L. Chen, Y. Xu, F. Xie, M. Huang, and Z. Zheng, "Data poisoning attacks on neighborhood-based recommender systems," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 6, p. e3872, 2021.

[138] Y. Deldjoo, T. D. Noia, and F. A. Merra, "A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[139] M. A. Cusumano, A. Gawer, and D. B. Yoffie, *The business of platforms: Strategy in the age of digital competition, innovation, and power*. Harper Business New York, 2019.

[140] O. C. Ferrell, M. Hartline, and B. W. Hochstein, *Marketing strategy*. Cengage Learning, 2021.

[141] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg security games: Looking beyond a decade of success." IJCAI, 2018.

[142] Z. Xiong, S. Feng, D. Niyato, P. Wang, Y. Zhang, and B. Lin, "A stackelberg game approach for sponsored content management in mobile data market with network effects," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5184–5201, 2020.

[143] J. Jin, C. Song, H. Li, K. Gai, J. Wang, and W. Zhang, "Real-time bidding with multi-agent reinforcement learning in display advertising," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 2193–2201.

[144] Y. Zhang, F. Rong, and Z. Wang, "Research on cold chain logistic service pricing—based on tripartite stackelberg game," *Neural Computing and Applications*, vol. 32, no. 1, pp. 213–222, 2020.

[145] L. Zhu and J. Lin, "A pricing strategy of e-commerce advertising cooperation in the stackelberg game model with different market power structure," *Algorithms*, vol. 12, no. 1, p. 24, 2019.

[146] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys (CSUR)*, 2022.

[147] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[148] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 393–402.

[149] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 322–330.

[150] H. Zhang, Y. Li, B. Ding, and J. Gao, "Practical data poisoning attack against next-item recommendation," in *Proceedings of The Web Conference 2020*, 2020, pp. 2458–2464.

[151] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *Proceedings of The Web Conference 2020*, 2020, pp. 3019–3025.

[152] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," *arXiv preprint arXiv:2101.02644*, 2021.

[153] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. Orgun, L. Cao, N. Wang, F. Ricci, and P. S. Yu, "Graph learning approaches to recommender systems: A review," *arXiv preprint arXiv:2004.11718*, 2020.

[154] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[155] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[156] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[157] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[158] P. J. Bickel and K. A. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC, 2015.

[159] R. Sato, M. Tanaka, and A. Takeda, "A gradient method for multilevel optimization," *arXiv preprint arXiv:2105.13954*, 2021.

[160] D. Fudenberg and D. Levine, "Subgame–perfect equilibria of finite–and infinite–horizon games," in *A Long-Run Collaboration On Long-Run Games*. World Scientific, 2009, pp. 3–20.

[161] E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala, "Generalized inner loop meta-learning," *arXiv preprint arXiv:1910.01727*, 2019.

[162] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[163] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[164] J. Tang, H. Gao, and H. Liu, "mtrust: Discerning multi-faceted trust in a connected world," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 93–102.

[165] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 2014, pp. 261–270.

[166] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, "Attacking recommender systems with augmented user profiles," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 855–864.

[167] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian *et al.*, "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4653–4664.

[168] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 4, pp. 23–es, 2007.

[169] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[170] M. Balaanand, N. Karthikeyan, S. Karthik, R. Varatharajan, G. Manogaran, and C. Sivaparthipan, "An enhanced graph-based semi-supervised learning algorithm to detect fake users on twitter," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 6085–6105, 2019.