Information Processing and Management 59 (2022) 103031

Contents lists available at ScienceDirect

Information Processing and Management

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

Detecting shilling groups in online recommender systems based on graph convolutional network

Shilei Wang [a], Peng Zhang [b], Hui Wang [a], Hongtao Yu [a], Fuzhi Zhang [a,*]

[a] School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei, China
[b] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

ARTICLE INFO

Keywords:
Recommender systems
Group shilling attacks
Graph convolutional network
Target item identification

ABSTRACT

Online recommender systems have been shown to be vulnerable to group shilling attacks in which attackers of a shilling group collaboratively inject fake profiles with the aim of increasing or decreasing the frequency that particular items are recommended. Existing detection methods mainly use the frequent itemset (dense subgraph) mining or clustering method to generate candidate groups and then utilize the hand-crafted features to identify shilling groups. However, such two-stage detection methods have two limitations. On the one hand, due to the sensitivity of support threshold or clustering parameters setting, it is difficult to guarantee the quality of candidate groups generated. On the other hand, they all rely on manual feature engineering to extract detection features, which is costly and time-consuming. To address these two limitations, we present a shilling group detection method based on graph convolutional network. First, we model the given dataset as a graph by treating users as nodes and co-rating relations between users as edges. By assigning edge weights and filtering normal user relations, we obtain the suspicious user relation graph. Second, we use principal component analysis to refine the rating features of users and obtain the user feature matrix. Third, we design a three-layer graph convolutional network model with a neighbor filtering mechanism and perform user classification by combining both structure and rating features of users. Finally, we detect shilling groups through identifying target items rated by the attackers according to the user classification results. Extensive experiments show that the classification accuracy and detection performance (F1-measure) of the proposed method can reach 98.92% and 99.92% on the Netflix dataset and 93.18% and 92.41% on the Amazon dataset.

1. Introduction

Recommender systems can filter the information that is attractive or valuable to their users and save the time of information retrieval for users (Wang et al., 2016b). As a result, recommender systems have been widely used in various fields such as e-commerce, social network, and financial planning. The application of recommender systems not only brings convenience to users, but also brings huge economic benefits to e-commerce platforms (Smith & Linden, 2017). For example, on an e-commerce platform, the recommendations produced by the recommender system will affect the purchase decisions of users, and thus affecting the product sales. Therefore, high-quality recommendations will bring a great business value. To guarantee the quality of recommender systems, two

* Corresponding author.
E-mail address: xjzfz@ysu.edu.cn (F. Zhang).

https://doi.org/10.1016/j.ipm.2022.103031
Received 14 November 2021; Received in revised form 30 April 2022; Accepted 13 July 2022
Available online 2 August 2022

aspects of research have been carried out. On the one hand, researchers (Hazrati & Ricci, 2021; Slokom et al., 2021) come up with proposals to improve the working mechanism of recommender systems. On the other hand, researchers aim to detect shilling attacks to enhance the robustness of recommender systems. Previous studies have indicated that recommender systems are vulnerable to shilling attacks. In such attacks, malicious users attempt to bias the output of a recommender system by providing fake profiles to achieve the purpose of promoting the sales of their products (Si & Li, 2020). Therefore, detection of shilling attacks is very important to guarantee the robustness of recommender systems.

1.1. Motivation

Over the past decade, researchers have put forward various methods to detect shilling attacks in order to enhance the robustness of recommender systems. Existing methods mainly focus on detection of individual shilling attacks that attackers inject fake profiles separately. However, the attackers might cooperate with each other to bias the output of a recommender system. This collaborative shilling behavior is termed a group shilling attack (Su et al., 2005), and a group of attackers with the same shilling behavior is called a shilling group. Group shilling attacks pose greater threat to recommender systems than individual ones, because attackers of a shilling group can elaborate their profiles to avoid detection. To detect shilling groups, some approaches (Cai & Zhang, 2021b; Wang et al., 2016a; Yu et al., 2021; Zhang et al., 2020; Zhou et al., 2014) have been proposed over the past few years. These approaches use frequent itemset (dense subgraph) mining or clustering method to obtain candidate groups and then utilize the hand-crafted features to detect shilling groups. While these approaches are effective for detecting shilling groups in recommender systems, they have the following two limitations. Firstly, these approaches need to divide candidate groups for the downstream detection task. However, it is difficult to generate high-quality candidate groups in real application scenarios due to the sensitivity of support threshold or clustering parameters setting. For example, in the detection methods that use frequent itemset mining (FIM) or dense subgraph mining to obtain candidate groups, if the support threshold is set too high, a very limited number of candidate groups will be generated. On the contrary, if the support threshold is set too low, it is possible to generate a large number of coincidental candidate groups. These two cases will cause a negative impact on the final detection performance. Secondly, the detection performance of these approaches depends highly on the quality of hand-crafted features. Moreover, the features extracted from a particular dataset lack universality, and thus limiting the application of these approaches in real scenarios. This necessitates designing an approach for shilling group detection that does not rely on manual feature engineering.

To cope with the limitations above, we present a graph convolutional network (GCN) based approach for detecting shilling groups. More specifically, we construct the suspicious user relation graph by analyzing the abnormal co-rating relations between users in the dataset and generate the user feature matrix using principal component analysis (PCA). Based on which, we design a three-layer GCN model with a neighbor filtering mechanism (named NFGCN), and take the suspicious user relation graph and the user feature matrix as the model's input to automatically extract features for user classification. Based on the user classification results, we obtain shilling groups through target item analysis.

This research differs from existing work in the following two aspects. Unlike existing methods that use FIM (Wang et al., 2016), the clustering (Zhang & Wang, 2020; Zhang et al., 2020), or the dense subgraph mining method (Cai & Zhang, 2021b; Yu et al., 2021) to divide candidate groups, the proposed approach does not need to divide candidate groups, which can avoid the negative impact of inaccurate group division on detection performance. Different from existing methods that use the hand-crafted features to detect shilling groups, the proposed approach can automatically extract features for shilling group detection, which improves the universality of group attack detection.

1.2. Research hypotheses

The aim of this research is to establish a framework for shilling group detection that does not rely on the manual feature engineering. In this work, we postulate the following three hypotheses.

(H1) Filtering out neighbors who have different labels with the center node can improve the classification performance of the GCN model.
(H2) Filtering out neighbors who have different labels with the center node can also reduce the computational cost of the GCN model.
(H3) Combining the proposed detection approach with specific recommendation algorithms can improve the robustness of the algorithms.

1.3. Contributions

The contributions of this paper include three aspects:

(1) We design a three-layer GCN model with a neighbor filtering mechanism (i.e., NFGCN). To realize the neighbor filtering mechanism, we calculate the user collaboration degrees based on rating values, rating scale, and rating time and design an activation function to filter neighbors with low collaboration degree. The introduction of neighbor filtering mechanism can optimize the aggregation of user features and greatly reduce the computational cost of the GCN model.

(2) We present a shilling group detection framework based on NFGCN and target item analysis (TIA), which is called NFGCN-TIA. The proposed detection framework combines both structure and rating features of users to perform user classification and utilizes the target item identification method to obtain shilling groups. This framework can not only avoid dividing candidate groups but also avoid the use of hand-crafted features.
(3) We conduct experiments to show the effectiveness of NFGCN-TIA. Particularly, we compare NFGCN with three representative graph neural network models to show its performance in user classification and compare NFGCN-TIA with seven baseline methods to illustrate its effectiveness in shilling group detection.

The rest of this paper is organized as follows. Sections 2 and 3 introduce the related work and preliminaries, respectively. Section 4 describes the methodology. Section 5 provides the experimental comparison and analysis. Section 6 concludes the paper.

Table 1
Summary of shilling attack detection methods.

Category	Method	Technique	Advantage	Disadvantage
Detection of individual attackers	Burke et al. (2006)	supervised	effective for specific attack strategies	less effective for obfuscated attacks
	Zhang and Zhou (2014)	supervised	individual profile based feature extraction method	the overall detection performance is not very high
	Li et al. (2015)	supervised	not easily affected by the obfuscated attack strategies	Not suitable for attacks with small filler and attack sizes
	Zhou et al. (2016)	supervised	high detection precision	low recall for attacks with small attack size
	Yang & Cai (2017)	supervised	effective for specific attack strategy	high computational cost
	Xu & Zhang (2019)	supervised	effective for attacks in social recommender system	sensitive to rating time
	Batmaz et al. (2020)	supervised	effective for binary data	need data labels
	Zhou & Duan (2021)	semi-supervised	effective for specific attack strategy	it requires to set more parameters
	Li et al. (2021)	supervised	it considers the high-order relationships in user-item interactions	it requires a large number of training samples
	Mehta & Nejdl (2009)	unsupervised	regardless of the specific attack strategy	poor performance for AoP attack and requires to know the attack size
	Lee & Zhu, 2012	unsupervised	regardless of the specific attack strategy	less effective for attacks with low filler size
	Zhang & Kulkarni (2013)	unsupervised	it does not need to specify the exact number of shilling profiles	it requires the shilling profiles to be highly correlated
	Zhang & Kulkarni (2014)	unsupervised	it does not need to specify the exact number of shilling profiles	it requires the shilling profiles to be highly correlated
	Zhang et al. (2015)	unsupervised	regardless of the specific attack strategy	it requires some seed spammers and attack size
	Xia et al. (2015)	unsupervised	effective for the target item with a large number of ratings from normal users	less effective for cold-start target items
	Yang et al. (2016)	unsupervised	it can detect various attacks with different filler sizes and attack sizes	require a set of seed spammers
	Zhang et al. (2018)	unsupervised	detection performance is high under various attacks	less effective for AoP attack
	Dou et al. (2018)	unsupervised	automatic feature extraction	sensitive to parameter settings
	Cai & Zhang (2019)	unsupervised	regardless of the specific attack strategy	less effective for identifying target items with a few ratings
	Cai & Zhang (2021a)	unsupervised	detection performance is high under various attacks	sensitive to parameter settings
Detection of shilling groups	Wang et al. (2016)	unsupervised	effective for tightly coupled shilling groups	less effective for loosely coupled shilling groups
	Zhou et al. (2014)	unsupervised	effective for specific shilling groups	less effective for shilling profiles with low similarity
	Zhang & Wang (2020)	unsupervised	effective for group shilling attacks	division of candidate groups is sensitive to rating time
	Wang et al. (2021)	unsupervised	effective for group shilling attacks	division of candidate groups is sensitive to parameters
	Zhang et al. (2020)	unsupervised	effective for group shilling attacks	not suitable for small shilling groups
	Yu et al. (2021)	unsupervised	effective for tightly coupled shilling groups	less effective for loosely coupled shilling groups
	Yu et al. (2021a)	supervised	automatic feature extraction	not suitable for small shilling groups
	Cai & Zhang (2021b)	unsupervised	effective for tightly coupled shilling groups	not suitable for small shilling groups

2. Related work

Currently, research on detection of shilling attacks in recommender systems focuses mainly on detection of individual attackers and detection of shilling groups. The difference between these two categories is as follows. On the one hand, from the perspective of feature extraction, the detection of individual attackers is mainly based on the individual behavior features of users while the detection of shilling groups is mainly based on the collusive behavior features between users. On the other hand, from the perspective of threat model, the detection of individual attackers is mainly aimed at individual shilling attacks while the detection of shilling groups is mainly aimed at group shilling attacks. In this section, we briefly review the related studies on the detection of individual attackers and shilling groups in recommender systems. Table 1 summarises the characteristics of methods for detecting individual attackers and shilling groups.

2.1. Detection of individual attackers

Most of existing methods focus on detecting individual attackers in recommender systems, which include supervised and unsupervised approaches. Supervised approaches (Batmaz et al., 2020; Burke et al., 2006; Li et al., 2015, 2021; Xu & Zhang, 2019; Yang & Cai, 2017; Zhang & Zhou, 2014; Zhou & Duan, 2021; Zhou et al., 2016) extract the features of attack profiles and use the labeled sample data to train a model for profile classification. Supervised approaches can obtain superior detection performance when having a large number of training samples. Nevertheless, such approaches can only detect known types of shilling attacks. Moreover, labeling a large number of training samples is an onerous task. Unsupervised approaches (Cai & Zhang, 2019, 2021a; Dou et al., 2018; Lee & Zhu, 2012; Mehta & Nejdl, 2009; Xia et al., 2015; Yang et al., 2016; Zhang & Kulkarni, 2013, 2014; Zhang et al., 2015; Zhang et al., 2018) extract the features that capture the rating behavior difference between genuine users and attackers, and use the clustering or graph mining methods to detect attackers. Unsupervised detection approaches do not need to consider specific attack types, nor do they require labeling sample data, but they need prior knowledge of shilling attacks. However, such prior knowledge is often difficult to obtain in the real recommender systems. As the aforementioned methods do not take into account the collaborative behavior between/among attackers, they are not applicable to detection of shilling groups in recommender systems.

2.2. Detection of shilling groups

Detection of shilling groups in recommender systems has recently received increasingly attention. In article Zhou et al. (2014), a two-step method was proposed to identify group attack profiles. This method applied two profile attributes and the K-means algorithm to generate two sets of suspicious users, respectively and obtained group shilling profiles by identifying target items. In article Wang et al. (2016b), a method for detecting shilling groups was proposed. This method first used FIM to divide candidate groups, then utilized six behavioral features to model the shilling group, and finally used a PCA-based unsupervised ranking method to obtain shilling groups. In article Zhang et al. (2020), a graph representation learning based approach was proposed for shilling group detection. This method utilized Node2vec to obtain user embeddings and employed the K-means++ algorithm to generate candidate groups. Based on which, it exploited a hierarchical clustering algorithm to obtain shilling groups. In article Zhang & Wang (2020), a bisecting K-means based approach was put forward to detect shilling groups. This method generated candidate groups from item rating tracks and calculated the suspicious degree of each candidate group using two group behavioral features, and obtained shilling groups with the bisecting K-means. Yu et al. (2021) used the dense subgraph mining method to detect shilling groups. In their method, candidate groups were acquired through triangle dense subgraph mining and several group indicators were used to compute group suspicious degrees. Wang et al. (2021) used a hierarchical topic model to obtain user rating vectors and divided candidate groups in line with rating value and rating time, and identified shilling groups by means of k-means algorithm. Yu et al. (2021a) proposed a shilling group detection method based on maximum dense subtensor mining. They first divided the rating time series of each item into time windows and constructed a three-dimensional tensor (i.e., user-rating-time window) model to generate item tensor groups. Then they used the M-Zoom model to mine the maximum dense subtensor of each item and selected the subtensor groups with high behavior consistency as candidate groups. Finally, they employed a dual-input convolutional neural network to extract features for user classification. Cai & Zhang (2021b) used the collusive relationship between users to build a user-user graph and applied the topological potential theory to acquire candidate groups. Based on which, they extracted the features of collusive behavior between group members and utilized the k-means algorithm to obtain shilling groups. Almost all of the aforementioned methods need to divide candidate groups. Moreover, they all rely on the manual feature engineering to extract features for shilling group detection, which is costly and time-consuming.

Compared with the existing detection approaches, the proposed NFGCN-TIA has two major characteristics. Firstly, NFGCN-TIA employs both structure and rating features of users in the rating database of the recommender system to detect shilling groups, which breaks through the limitation of low detection performance of existing methods that only consider user structure features or user rating features. Secondly, NFGCN-TIA exploits a two-stage method combining user classification and target item identification to identify shilling groups. This design not only enables NFGCN-TIA to detect shilling groups with various sizes, but also avoids the problem of existing methods that divide small shilling groups together with normal user groups when generating candidate groups.

3. Preliminaries

In this section, we give the threat models for group shilling attacks and briefly introduce the graph convolutional network.

3.1. Threat models

The threat models or attack models refer to the approaches that attackers construct shilling profiles to achieve their attack goals (e.g., promoting an item to be recommended). A shilling profile is a rating vector that consists of a number of biased ratings assigned with specific strategies, including a maximum or minimum rating for the target item to be promoted or demoted. The items in the shilling profile generally include the selected items (I_S), the filler items (I_F), the unrated items ($I_\emptyset$), and the target items (I_T) (Mobasher et al., 2007). The threat models for individual shilling attacks in recommender systems have been well studied, including random attack model, average attack model, average over popular (AoP) items attack model (Hurley et al., 2009), power item attack (PIA) model (Seminario & Wilson, 2014), and so on.

A tricksy group attack model was proposed by Wang et al. (2012) to generate shilling groups, including a strict version ($GSAGen_s$) and a loose version ($GSAGen_l$). $GSAGen_s$ ($GSAGen_l$) includes two types: $GSAGen_s$ Ran and $GSAGen_s$ Avg ($GSAGen_l$ Ran and $GSAGen_l$ Avg) in which the random attack model and average attack model are used for generating shilling groups, respectively. $GSAGen_s$ has more stringent constraints than $GSAGen_l$, so the size of shilling groups generated with $GSAGen_s$ is less than that generated with $GSAGen_l$. To enrich the types of group attack model, we also use the AoP attack model and PIA model as the basis to implement another four types of group attack model, which are named as $GSAGen_s$ AoP, $GSAGen_s$ PIA, $GSAGen_l$ AoP, and $GSAGen_l$ PIA, respectively. Table 2 summarizes the threat models for group shilling attacks.

3.2. Graph convolutional network

Graph convolutional network (GCN) (Kipf & Welling, 2017) provides an effective way to learn deep representations for graph data. Given an undirected graph G with N nodes, suppose $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix of graph G, $\boldsymbol{X}$ is a matrix of node feature vectors, $\boldsymbol{I_N} \in \mathbb{R}^{N\times N}$ is an identity matrix, and $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I_N}$ is the adjacency matrix of graph G with self-connection of nodes, then the forward propagation process of two-layer GCN can be expressed as

$$P = softmax\left(\widehat{\boldsymbol{A}} ReLU\left(\widehat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(0)}\right)\boldsymbol{W}^{(1)}\right) \tag{1}$$

where $\widehat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ and $\widetilde{\boldsymbol{D}}$ is the node degree matrix of $\widetilde{\boldsymbol{A}}$, $\boldsymbol{W}^{(0)}$ and $\boldsymbol{W}^{(1)}$ are trainable weight matrices, $ReLU()$ is a nonlinear activation function, and $softmax(P_i) = \exp(P_i)/\sum_i \exp(P_i)$.

For semi-supervised classification, the cross-entropy error over labeled examples is evaluated:

$$\mathscr{L} = -\sum_{l\in\mathscr{Y}_L}\sum_{f=1}^{F} Y_{lf} ln P_{lf} \tag{2}$$

where $\mathscr{Y}_L$ is the set of nodes with labels, F is the output dimension of node representations, $Y \in \mathbb{R}^{|\mathscr{Y}_L|\times F}$ is the label indicator matrix.

4. Methodology

The proposed NFGCN-TIA mainly includes two stages: user classification and shilling group detection, which is shown in Fig. 1. In the stage of user classification, we build a suspicious user relation graph and perform PCA on the rating matrix to obtain the user feature matrix. Both structure and rating features of users are input to the NFGCN model to perform user classification. In the stage of shilling group detection, we identify the target items using TIA method and divide attackers who have rated the same target items into

Table 2
The threat models for group shilling attacks.

Version	Type	I_s		I_F		I_T
		Items	Ratings	Items	Ratings	Rating
$GSAGen_s$	$GSAGen_s$ Ran	I_S=∅	item is chosen randomly and only rated by an attacker	system mean	r_{max} or r_{min}	
	$GSAGen_s$ Avg	I_S=∅	item is chosen randomly and only rated by an attacker	item mean	r_{max} or r_{min}	
	$GSAGen_s$ AoP	I_S=∅	item is chosen randomly from x% popular items and only rated by an attacker	item mean	r_{max} or r_{min}	
	$GSAGen_s$ PIA	item is chosen randomly from power items and only rated by an attacker	item mean	I_F=∅	r_{max} or r_{min}	
$GSAGen_l$	$GSAGen_l$ Ran	I_S=∅	item is randomly chosen and rated by at most two attackers	system mean	r_{max} or r_{min}	
	$GSAGen_l$ Avg	I_S=∅	item is randomly chosen and rated by at most two attackers	item mean	r_{max} or r_{min}	
	$GSAGen_l$ AoP	I_S=∅	item is chosen randomly from x% popular items and rated by at most two attackers	item mean	r_{max} or r_{min}	
	$GSAGen_l$ PIA	item is chosen randomly from power items and rated by at most two attackers	item mean	I_F=∅	r_{max} or r_{min}	

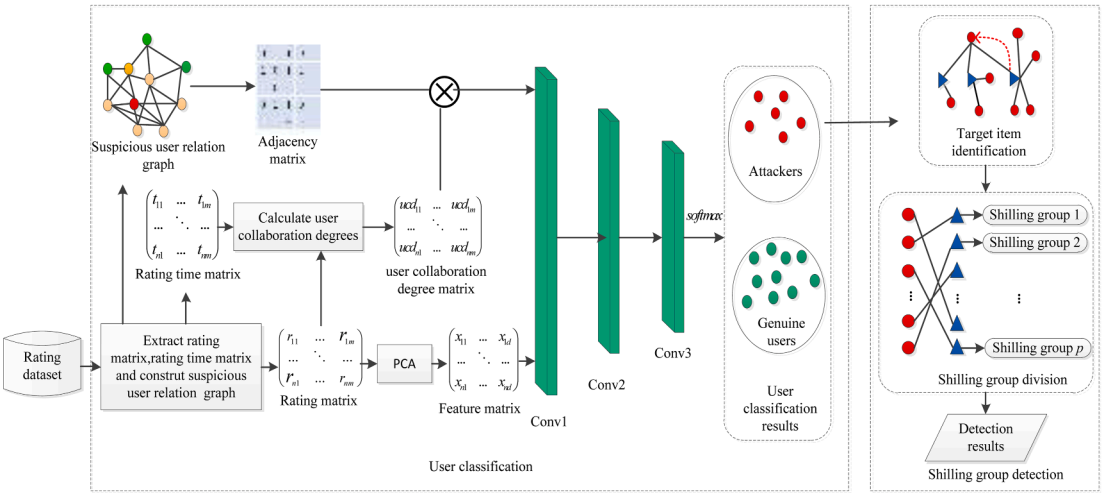

Fig. 1. The detection framework of NFGCN-TIA.

the same groups based on the user classification results, and thus obtain the shilling groups.

To facilitate discussion, Table 3 provides the notations and their descriptions.

4.1. User classification

In this section, we design a three-layer GCN model with a neighbor filtering mechanism (NFGCN) and take the structure and rating features of users as its input to perform user classification.

4.1.1. Obtaining the structure and rating features of users

We can obtain the structure features of users by constructing the suspicious user relation graph. First, we model the given dataset as a graph by treating users as nodes and co-rating relations between users as edges. Second, we add a weight 1 or 0 to each edge to reserve suspicious user relations and filter normal user relations. Here, the suspicious user relation refers to that two users rate the same item with r_{max} because attackers of shilling group usually provide the highest rating for the target item to be promoted. For users $u, v \in U$, if there exists an edge between u and v, the weight between them is defined as:

$$E_w(u,v) = \begin{cases} 1, & \exists i \in I_{uv} \wedge (r_{ui} = r_{\max}, r_{vi} = r_{\max}) \\ 0, & otherwise \end{cases} \tag{3}$$

where I_{uv} represents the set of items co-rated by u and v.

In the graph, we reserve edges with weight of 1 and obtain the suspicious user relation graph.

The user feature matrix can be constructed in the following way. First, we construct the user-item rating matrix $\boldsymbol{R}$ according to the users' ratings for items in the dataset. Second, we obtain the user feature matrix $\boldsymbol{X}$ through PCA dimensionality reduction of $\boldsymbol{R}$.

Table 3
Notations and descriptions.

Notations	Descriptions
U	set of users
I	set of items
$\boldsymbol{R}$	user rating matrix
$\boldsymbol{X}$	user feature matrix
$\boldsymbol{A}$	adjacency matrix
$\boldsymbol{UCDM}$	user collaboration degree matrix
$\boldsymbol{Z}$	user embedding representations
$\boldsymbol{W}^{(l)}$	the trainable weight matrix in layer l
$\boldsymbol{PL}$	the predicted user label matrix
r_{ui}	rating of user u for item i
r_{max}	the maximum rating allowed in the dataset
$\lvert \bullet \rvert$	number of elements in the set

4.1.2. Construction of NFGCN model

The standard GCN model (Kipf & Welling, 2017) aggregates all the neighbor node features of the center node. However, in the group shilling attacks, besides rating the target items, attackers also rate many non-target items to build relations with genuine users. As a result, there are many genuine user nodes in the neighbor nodes of attackers in the suspicious user relation graph. Therefore, the feature aggregation strategy of the standard GCN will make the features of attacker nodes be mixed with features of certain genuine user nodes. To solve this problem, we design a three-layer GCN model with a neighbor filtering mechanism.

First, we establish the neighbor filtering mechanism. The purpose of neighbor filtering is to reduce the influence of genuine user nodes on the features of attack user nodes when using the GCN model to learn the features of nodes in the suspicious user relation graph. To this end, we introduce the concept of user collaboration degree. Two users with high collaboration degree are highly likely to be attackers of the same group. Therefore, we use the user collaboration degree as the basis of neighbor filtering.

In a group shilling attack, attackers will provide the highest rating for the target item(s) to be promoted. To ensure the effect of attack, the rating time of attackers will not be far apart. In addition, although attackers rate different non-target items, the number of non-target items rated by attackers is very close under the same attack strategy. Therefore, we can calculate the user collaboration degree by analyzing the user rating behavior from the aspects of rating scale and rating time.

Definition 1. (*user rating scale ratio, URSR*). For users $u, v \in U$, the rating scale ratio between u and v is defined as follows:

$$URSR_{uv} = min(N_u / N_v, N_v / N_u) \tag{4}$$

where N_u and N_v represent rating numbers of u and v, respectively.

Since the rating time of attackers in a shilling group for the target item(s) is close, we can compute the rating time similarity between two users on each co-rated item with the highest rating.

Definition 2. (*user rating time similarity, URTS*). For users $u, v \in U$, suppose $MRI_{uv} = \{i | r_{ui} = r_{\max} \wedge r_{vi} = r_{\max}\}$ is a set of items co-rated by u and v with the highest rating r_{max}, the rating time similarity between u and v is calculated as follows:

$$URTS_{uv} = \frac{1}{|MRI_{uv}|} \sum_{i \in MRI_{uv}} Sim_{uv}^{i} \tag{5}$$

$$Sim_{uv}^{i} = \begin{cases} 1 & , \ |t_{ui} - t_{vi}| \leq \tau_T \\ \tau_T / |t_{ui} - t_{vi}| & , \ |t_{ui} - t_{vi}| > \tau_T \end{cases} \tag{6}$$

where Sim_{uv}^{i} denotes the rating time similarity between u and v on the co-rated item i, t_{ui} and t_{vi} denote the rating time of u and v for item i, and $\tau_T$is the time interval threshold and is set to 30.

Definition 3. (*user collaboration degree, UCD*). For users $u, v \in U$, the collaboration degree between u and v is defined as follows:

$$UCD_{uv} = URSR_{uv} \times URTS_{uv} \tag{7}$$

We can use Eq. (7) to calculate the collaboration degrees between users and obtain the user collaboration degree matrix ***UCDM***.

Based on the user collaboration degree matrix obtained, we design a High-pass Linear Units (HpLU) to achieve neighbor filtering, which is defined as follows:

$$HpLU(x) = \begin{cases} 1, & x > \theta \\ 0, & x \leq \theta \end{cases} \tag{8}$$

where θ is a threshold parameter that is set by experiment.

Second, we take $HpLU(x)$ as an activation function to construct a three-layer GCN model with a neighbor filtering mechanism, namely NFGCN.

The forward propagation of NFGCN can be expressed as

$$\widehat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}(HpLU(\widetilde{\boldsymbol{A}} \otimes \boldsymbol{UCDM}))\widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \tag{9}$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ is the adjacency matrix with self-connection of nodes and $\boldsymbol{I}_N$ is the identity matrix, $\widetilde{\boldsymbol{D}}$ is the degree matrix of $\widetilde{\boldsymbol{A}}$, $HpLU(\bullet)$ is used to filter out neighbors, and $\otimes$ represents the multiplication of the elements in the corresponding positions of two matrices.

The output features (i.e., the user embeddings) of NFGCN are as follows:

$$\boldsymbol{Z} = \widehat{\boldsymbol{A}} ReLU\left(\widehat{\boldsymbol{A}} ReLU\left(\widehat{\boldsymbol{A}} \boldsymbol{X} \boldsymbol{W}^{(1)}\right) \boldsymbol{W}^{(2)}\right) \boldsymbol{W}^{(3)} \tag{10}$$

where $ReLU(\bullet) = max(\bullet, 0)$.

The use of neighbor filtering mechanism can ensure that pairwise user nodes whose collaboration degree is greater than the given threshold participate in feature aggregation. This can not only reduce the impact of genuine user nodes on the features of attack user nodes, but also reduce the computational cost of the GCN model.

4.1.3. NFGCN-based user classification

After constructing the NFGCN model, we use the semi-supervised method to classify users. In order to train NFGCN, we need a small number of user labels for semi-supervised learning. To this end, we sample a certain number of users and their labels from the dataset to build the training set *Tr*. *Tr* consists of a number of triples $< u, l_1^u, l_2^u >$ where l_1^u, l_2^u denote the labels of user u. If u is a genuine user, l_1^u and l_2^u are 1 and 0, respectively. If u is an attacker, l_1^u and l_2^u are 0 and 1, respectively.

Based on the output features of NFGCN, for each $u \in U$, we can use the *softmax* function to compute the probabilities that user u belongs to two labels, that is:

$$s_1^u = \frac{\exp(\boldsymbol{Z}_{u,1})}{\exp(\boldsymbol{Z}_{u,1}) + \exp(\boldsymbol{Z}_{u,2})} \tag{11}$$

$$s_2^u = \frac{\exp(\boldsymbol{Z}_{u,2})}{\exp(\boldsymbol{Z}_{u,1}) + \exp(\boldsymbol{Z}_{u,2})} \tag{12}$$

If $s_1^u > s_2^u$, user u is a genuine user (i.e., $l_1^u = 1$ and $l_2^u = 0$); otherwise, user u is an attacker (i.e., $l_1^u = 0$ and $l_2^u = 1$).

The weight matrices of the NFGCN model are initialized randomly. To improve the model performance, we utilize the training set to optimize the weight matrices.

The cross-entropy loss is used to measure the classification effect of NFGCN model:

$$loss = -\frac{1}{|Tr|}\sum_{u \in Tr}\left[l_1^u \times \log(s_1^u) + l_2^u \times \log(s_2^u)\right] \tag{13}$$

Based on Eq. (13), we use the stochastic gradient descent method to optimize the weight matrices. Particularly, we use part of user labels to calculate the loss according to Eq. (13) in each forward propagation. The loss is propagated back through the stochastic gradient descent algorithm and the weights are updated. This process is repeated until the loss is small enough. The hyper-parameters are adjusted according to the performance of the model. In our model, the learning rate is set to 0.01, the dropout is set to 0.5, and the epoch is set to 200.

We can use the trained NFGCN model to predict the labels for all users and obtain the predicted user label matrix $\boldsymbol{PL}$. Algorithm 1

Algorithm 1
NFGCN-based user classification.

Input: rating dataset
training set *Tr*
dimension of user feature vector f
threshold parameter θ
number of iterations *loop*
Output: the predicted user label matrix $\boldsymbol{PL}^{|U|\times 3}$
Begin
1: construct the suspicious user relation graph from the dataset according to Eq. (3) and obtain the adjacency matrix $\boldsymbol{A}$
2: construct the user rating matrix$\boldsymbol{R}^{|U|\times|I|}$ from the dataset and use PCA to obtain the user feature matrix $\boldsymbol{X}^{|U|\times f}$
3: calculate the collaboration degrees between users according to Eqs. (4)–(7) and obtain the user collaboration degree matrix $\boldsymbol{UCDM}$
4: initialize the weight matrices with random values
5: **for** $k = 1$ to *loop* **do**
6: calculate the user embeddings $\boldsymbol{Z}$ according to Eq. (10)
7: calculate the cross-entropy loss according to Eq. (13) and perform back-propagation to update the weight matrices
8: **end for**
9: obtain the predicted user label matrix $\boldsymbol{PL}$
10: return $\boldsymbol{PL}$
End

describes the process of NFGCN-based user classification.

In Algorithm 1, the time complexity for constructing the adjacency matrix $\boldsymbol{A}$ (Line 1), constructing the user rating matrix $\boldsymbol{R}$ and obtaining the user feature matrix $\boldsymbol{X}$ (Line 2), constructing the user collaboration degree matrix (Line 3), and updating model parameters and predicting user labels (Lines 4–9) is $O(|U|^2 \times |I|)$, $O(|U| \times |I|) + O(|U| \times f^2 + f^3)$, $O(|U|^2)$, and $O(loop \times |U|^2)$ respectively. Therefore, the time complexity of Algorithm 1 is $O(|U| \times (|U| \times |I| + |U| + |I| + f^2 + |U| \times loop) + f^3)$.

4.2. Shilling group detection

According to the user classification results obtained with NFGCN, we utilize the target item identification method for shilling group detection. First, we determine the target items for the attackers by analyzing their rating behaviors on the co-rated items. Second, we divide the attackers into different shilling groups according to the target items on which they attack.

In a shilling group, the attackers usually provide the highest rating r_{max} for the target item in order to promote it, so the items rated by the attackers with r_{max} are deemed as suspicious items. For an attacker u, the set of suspicious items rated by u is defined as follows:

$$SI_u = \left\{ i \middle| i \in I \wedge r_{ui} = r_{\max} \wedge \boldsymbol{PL}_{u,3} = 1 \right\}$$

The attackers of the shilling group usually work together to promote the target item. Therefore, the more attackers who rate a suspicious item with r_{max}, the more likely that it is a target item. For the set of suspicious items rated by each attacker, we can sort the suspicious items according to the number of attackers who rate them and select the suspicious item rated by the largest number of attackers as the target item of this attacker. The attackers who rate the same target items are divided into the same shilling groups.

Algorithm 2 describes the process of shilling group detection, in which Lines 2–13 identify the target items and attackers who rate these target items, and Lines 14–24 obtain the shilling groups according to the target items identified.

In Algorithm 2, the time complexity for identifying the target items and attackers who rate these target items (Lines 2–13) and obtaining the shilling groups (Lines 14–24) is $O((|U| + |I|) \times |U| \times |I|)$ and $O(|U| \times |I|)$, respectively, so the time complexity of Algorithm 2 is $O((|U| + |I| + 1) \times |U| \times |I|)$.

Algorithm 2
shilling group detection.

Input: U, I, $\boldsymbol{PL}$
Output: the set of shilling groups SG
Begin
1: $SG \leftarrow \varnothing$, $UTI \leftarrow \varnothing$
2: **for** $\forall u \in U$ **do**
3: **if** $\boldsymbol{PL}_{u,3} = 1$ **then**
4: $SI_u \leftarrow$ construct u's suspicious item set
5: $UNC_u \leftarrow \varnothing$
6: **for** $\forall i \in SI_u$ **do**
7: $nc_{ui} \leftarrow$ count the number of attackers who rate item i
8: $UNC_u \leftarrow UNC_u \cup \{(i, nc_i)\}$
9: **end for**
10: $i_t \leftarrow$ obtain i with the largest nc_i in UNC_u
11: $UTI \leftarrow UTI \cup \{(u, i_t)\}$
12: **end if**
13: **end for**
14: **for** $\forall i \in I$ **do**
15: $SG_i \leftarrow \varnothing$
16: **for** $\forall u \in U$ **do**
17: **if** $(u, i) \in UTI$ **then**
18: $SG_i \leftarrow SG_i \cup \{u\}$
19: **end if**
20: **end for**
21: **if** $|SG_i| \geq 2$ **then**
22: $SG \leftarrow SG \cup \{SG_i\}$
23: **end if**
24: **end for**
25: return SG
End

5. Experiments

In this section, we conduct experiments to test the hypotheses (H1), (H2), and (H3) described in Section 1.2 and simultaneously address the following three research questions.

(RQ1) How does the number of layers affect the classification performance of the NFGCN model?
(RQ2) How does the proposed approach perform vs. state-of-the-art baseline methods in detecting shilling groups?
(RQ3) How does the neighbor filtering threshold parameter affect the detection performance of the proposed approach?

5.1. Experimental datasets

In order to show the effectiveness of the proposed approach, experiments are conducted on the following datasets.

(1) Netflix dataset (Koren, 2010) includes 130,297,638 ratings provided by 480,186 users for 17,770 movies. For the purpose of evaluation, we randomly sample 215,844 ratings from 2000 users on 3985 movies and inject 64 shilling groups generated with the threat models described in Section 2.1 to construct a synthetic dataset. To ensure the size of the generated shilling groups, the loose version threat models (i.e., $GSAGen_l$ Ran, $GSAGen_l$ Avg, $GSAGen_l$ AoP, and $GSAGen_l$ PIA) are used to generate shilling groups. The target item for each shilling group is randomly chosen from the items with few ratings. For each of four basic shilling attack models used for generating shilling groups, the filler size is set to {2.5%, 5%} and the attack size is set to {2.5%, 5%, 7.5%, 10%}, respectively. We use each combination of the basic shilling attack model, the filler size, and the attack size to generate 2 shilling groups, respectively and obtain a total of 64 shilling groups in which there are 552 attackers.
(2) Amazon review dataset (Xu et al., 2013) consists of 1,205,125 ratings provided by 645,072 users for 136,785 products. For the purpose of evaluation, we extract 5055 users with labels and their ratings for products to construct a labeled dataset, which consists of 53,777 ratings from 5055 users (1937 attackers and 3118 genuine users) on 17,610 products.
(3) Movielens 1 M dataset contains 1,000,209 anonymous ratings of 3947 movies from 6010 MovieLens users who joined MovieLens in 2000. $GSAGen_l$ Ran, $GSAGen_l$ Avg, $GSAGen_l$ AoP, and $GSAGen_l$ PIA are used to generate shilling groups. Half of the target items are randomly chosen from popular items and Half of the target items are randomly chosen from unpopular items. The remaining settings are the same as those on the Netflix dataset. Finally, 64 shilling groups including a total of 690 attackers are injected into the Moivelens 1 M dataset for experimental evaluation.

The characteristics of the datasets are summarized in Table 4.

5.2. Evaluation metrics

The precision, recall, accuracy, and F1-measure are used for evaluation, which are defined as follows:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

Table 4
The characteristics of the datasets.

Dataset	Netflix	Amazon	Movielens 1M
number of total users	2552	5055	6700
number of total items	3985	17,610	3947
number of ratings	288,554	53,777	1,079,706
maximum rating	5	5	5
minimum rating	1	1	1
sparsity	97.16%	99.94%	95.92%
number of genuine users	2000	3118	6010
number of attackers	552	1937	690
proportion of genuine users	78.37%	61.68%	89.7%
proportion of targets rating	25.18%	33.92%	7.36%
average number of ratings per item	72.41	3.05	273.55
average number of ratings per user	113.06	10.64	161.15
average number of ratings per attacker	130.94	9.41	122.22
average number of ratings per genuine user	107.99	11.39	166.42
average rating value of the dataset	3.62	4.41	3.57

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{16}$$

$$F1 - measure = \frac{2 \times Precison \times Recall}{Precison + Recall} \tag{17}$$

where *FN* and *TP* are numbers of attackers misclassified and correctly classified respectively, *FP* and *TN* are numbers of genuine users misclassified and correctly classified respectively.

5.3. Comparison of classification performance with other models

To test the first hypothesis (H1), we conduct experiments to evaluate the NFGCN model on node classification task. Particularly, we carry out experiments on the Netflix and Amazon datasets to compare NFGCN with three graph neural network models in classification performance. The input to the graph neural network models is the constructed suspicious user relation graph. The ratio of training set, validation set, and test set is 6:2:2. In the experiments, the learning rate of NFGCN is 0.01 and the number of iterations of NFGCN is 200.

(1) GCN (Kipf & Welling, 2017) is an end-to-end model, which generates embeddings by aggregating neighbor node features. In the experiments, the learning rate is 0.01 and the number of iterations is 200.
(2) GraphSAGE (Hamilton et al., 2017) is a representation learning method for large graphs, which samples and aggregates features from a node's local neighbors. In the experiments, the learning rate is 0.01, the number of iterations is 400, the sampling depth is 2, and the number of neighbors sampled in each layer is 10.
(3) GAT (Velickovic et al., 2018) introduces an attention mechanism to GCN, which not only learns the weight of each dimension of features, but also learns the weight of each edge. In the experiments, the learning rate is 0.005, the number of iterations is 1000, the number of attention heads is 8, and α in the activation function LeakyReLU is 0.2.

Table 5 shows the classification performance of four models on the Netflix and Amazon datasets.

In Table 5, the classification performance of GCN on the Amazon dataset is much better than that on the Netflix dataset, and the precision, recall, accuracy, and F1-measure increase by 26.07%, 39.6%, 16.04%, and 33.28%, respectively. This is not difficult to explain. On the Netflix dataset, there are 67.51% genuine users in the neighbors of all the attackers, so the features of attackers are similar to those of genuine users after neighbor feature aggregation, which has a negative effect on the classification performance. On the Amazon dataset, the proportion of genuine users is only 9.53% in all the neighbors of attackers, so the genuine users' features have little impact on the attackers' features.

GraphSAGE randomly samples a certain number of neighbors to perform feature aggregation. Moreover, the probability of sampling genuine users for attackers is much higher on the Netflix dataset than that on the Amazon dataset. Therefore, the classification performance of GraphSAGE on the Amazon dataset is much better than that on the Netflix dataset, and the precision, recall, accuracy, and F1-measure increase by 20.81%, 27%, 7.89%, and 24.13%, respectively.

Compared with the GCN and GraphSAGE models, the classification performance of GAT on two datasets is relatively stable and is very close. For example, the classification accuracy of GAT is 0.6132 on the Netflix dataset and 0.6261 on the Amazon dataset. This is because GAT assigns weights to neighbors before aggregating features. After training, GAT can learn the important degree of neighbors, so it is almost not affected by the neighbor distribution and has little difference in classification performance on two datasets.

As shown in Table 5, NFGCN has the best classification performance on the Netflix and Amazon datasets among the four models. For example, the classification accuracy of NFGCN on the Netflix dataset is 0.9892, which has an improvement of 22.13%, 20.44%, and 37.6% respectively in comparison with GCN, GraphSAGE, and GAT. This is because NFGCN can filter out genuine user neighbors of attackers in the process of feature aggregation and thus learns better user vector representations for classification.

Based on the analyses above, it can be concluded that NFGCN performs better than GCN, GraphSAGE, and GAT in classifying users in the Netflix and Amazon datasets.

5.4. Model evaluation by considering the output of each layer

To answer the first research question (RQ1), we conduct experiments on the Netflix, Amazon, and Movielens 1 M datasets to

Table 5
Comparison of classification performance for four models.

	Netflix precision	recall	accuracy	F1-measure	Amazon precision	recall	accuracy	F1-measure
GCN	0.6426	0.5378	0.7679	0.5855	0.9033	0.9338	0.9283	0.9183
GraphSAGE	0.6276	0.5534	0.7848	0.5882	0.8357	0.8234	0.8637	0.8295
GAT	0.3225	0.317	0.6132	0.3197	0.3498	0.3931	0.6261	0.3702
NFGCN	**0.9376**	**0.9992**	**0.9892**	**0.9674**	**0.9175**	**0.9438**	**0.9318**	**0.9305**

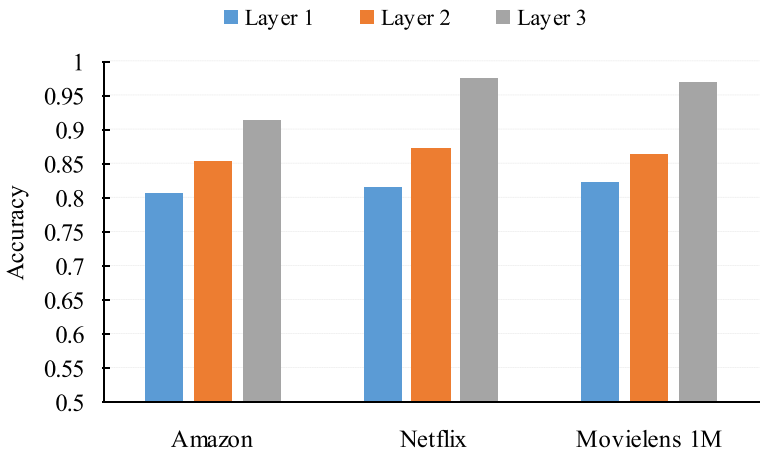

Fig. 2. The classification accuracy of the NFGCN model with different numbers of convolutional layers.

observe the classification accuracy of the NFGCN model from the output of each convolutional layer. Firstly, we obtain the embeddings of users from each convolutional layer. The dimensions of embeddings are 64, 16, and 2 respectively. Secondly, we use the K-means clustering algorithm to perform a binary classification task and calculate the classification accuracy obtained in line with the embeddings of each layer. Fig. 2 shows the classification accuracy of the NFGCN model with different numbers of convolutional layers on three datasets.

As can be seen in Fig. 2, the classification accuracy obtained through the embeddings of the first convolutional layer (Layer 1) is above 0.8, which is the worst on three datasets. The classification accuracy obtained through the embeddings of the second convolutional layer (Layer 2) is above 0.85, which is better than that of Layer 1 on three datasets. The classification accuracy obtained through the embeddings of the third convolutional layer (Layer 3) is all above 0.9, which is the best on three datasets. This indicates that the classification accuracy of the NFGCN model increases as the number of convolutional layers increases. Moreover, the NFGCN model with three convolutional layers can yield the best classification accuracy.

5.5. Comparison of detection performance with other methods

To answer the second research question (RQ2), we carry out experiments on three datasets and compare NFGCN-TIA with the following seven methods in detection performance.

(1) FAP (Zhang et al., 2015) detects shilling attacks by ranking users according to their suspicious degrees. This method requires labeling a part of attackers as seed users and conducts recursive propagation on the bipartite graph to calculate user suspicious degrees. In the experiments, we select 0.5% attackers as the seed users.
(2) UD-HMM (Zhang et al., 2018) uses hidden Markov model to model the users' rating behaviors and calculates suspicious degrees of users by analyzing the users' preference sequences. Based on which, it utilizes a hierarchical clustering algorithm to detect attackers. In the experiments, parameter N is set to 5 and parameter α is set to 0.8.
(3) DCEDM (Hao & Zhang, 2021) detects shilling attacks by extracting the robust graph features using the stacked denoising autoencoders. In the experiments, parameters ρ and β are set to 0.05 and 0.9 respectively, the noise level α is set to 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1, respectively.
(4) TP-GBF (Cai & Zhang, 2021b) is a three-stage approach for shilling group detection. This method builds the weighted user relation graph based on user lockstep behaviors and introduces a topological potential method to find suspicious groups. Then it

Table 6
Comparison of detection performance for eight methods.

	Netflix			Amazon			Movielens 1M		
	precision	recall	F1-measure	precision	recall	F1-measure	precision	recall	F1-measure
FAP	0.8388	0.8279	0.8333	0.4952	0.4924	0.4938	0.4362	0.4288	0.4325
UD-HMM	0.9509	0.9961	0.9730	0.3755	0.4196	0.3963	0.5873	0.6784	0.6295
DCEDM	0.9964	0.5327	0.6942	0.8637	0.7221	0.7866	0.6354	0.4327	0.5148
TP-GBF	0.9977	0.7808	0.8760	0.9283	0.6467	0.7623	0.4362	0.9564	0.5991
GD-BKM	0.9875	0.9888	0.9881	0.8234	0.6732	0.7408	0.1391	0.7818	0.2362
CoDetector	0.4126	0.3371	0.3711	0.8056	0.8345	0.8198	0.1771	0.1586	0.1673
GAGE	0.9961	0.4692	0.6379	0.8579	0.8694	0.8636	0.2734	0.9055	0.4199
NFGCN-TIA	**0.9986**	**0.9998**	**0.9992**	**0.9343**	**0.9142**	**0.9241**	**0.9965**	**0.9637**	**0.9798**

utilizes the K-means algorithm to detect shilling groups. In the experiments, parameter α is set to 1 on the Netflix dataset, 0.47 on the Amazon dataset, and 0.9 on the Moivelens 1 M dataset.

(5) GD-BKM (Zhang & Wang, 2020) is a shilling group detection approach based on rating time burstiness. It extracts user features and item features to compute group suspicious degrees and uses the bisecting K-means algorithm to detect shilling groups. In the experiments, parameters *K* and *TIL* are set to 4 and 30, respectively.

(6) CoDetector (Dou et al., 2018) uses matrix factorization and user embedding to reveal the implicit interaction between users and items, and utilizes the decision tree algorithm to detect the attackers. In the experiments, 80% of the data are selected as the training set and the remaining 20% as the test set. The dimension of the latent factor, the number of negative samples, the number of iterations, and the learning rate are set to 10, 25, 200, and 0.01, respectively

(7) GAGE (Zhang et al., 2020) is a deep learning-based method to detect group shilling attacks. This method analyzes the user rating behaviors to build a user relationship graph, uses the Node2vec model to obtain user node vector representations, employs the K-means++ clustering algorithm to generate candidate groups, and exploits the hierarchical clustering algorithm to obtain shilling groups. In the experiments, Parameters *GS, p*, and *q* are set to 30, 7, and 0.2, respectively.

5.5.1. *Comparison of performance in detecting traditional attack model-based shilling groups*

Table 6 shows the precision, recall, and F1-measure of NFGCN-TIA, FAP, UD-HMM, DCEDM, TP-GBF, GD-BKM, CoDetector, and GAGE on three datasets in detecting traditional attack model-based shilling groups.

In Table 6, on the Netflix dataset, NFGCN-TIA has better precision, recall, and F1-measure than FAP, UD-HMM, DCEDM, TP-GBF, GD-BKM, CoDetector, and GAGE. Moreover, these three metrics are close to 1. On the Amazon dataset, the three metrics of NFGCN-TIA are also better than those of FAP, UD-HMM, DCEDM, TP-GBF, GD-BKM, CoDetector, and GAGE. For example, the F1-measure of NFGCN-TIA is 0.9241, which has an improvement of 43.03%, 52.78%, 13.75%, 16.18%, 18.33%, 10.43%, and 6.05% respectively in comparison with FAP, UD-HMM, DCEDM, TP-GBF, GD-BKM, CoDetector, and GAGE. The main reason is that the proposed neighbor filtering strategy can filter genuine neighbors of attackers. When NFGCN aggregates neighbor features, the features of attackers will not be affected by the features of genuine users, which further enlarges the difference between attackers and genuine users.

For FAP, parts of attackers are selected as seed users in advance and the suspicious degrees of these seed users are set to 1, then the suspicious degrees of the rest users are calculated through the suspicious degree propagation. As 0.5% of attackers are used as the seed users, FAP performs well on the Netflix dataset and three metrics are all above 0.8. As shown in Table 6, the three metrics of FAP on the Amazon dataset are lower than those on the Netflix dataset, which are no more than 0.5. This is because the shilling groups of the Netflix dataset are generated with group attack models and the attackers in the same shilling groups have similar rating behaviors, which leads to a high transition probability between attackers.

As shown in Table 6, UD-HMM performs well on the Netflix dataset and three metrics are all above 0.95, which are much better than those on the Amazon dataset. This is not difficult to explain. On the Netflix dataset, target items of the shilling groups are randomly chosen from unpopular items and non-target items are randomly chosen from popular items, leading to the difference in preference sequences between attackers and genuine users. In addition, the Amazon dataset is very sparse and there is little difference in popularity between target items and other items, which makes the preference sequences of attackers and genuine users have little difference.

The detection precision of DCEDM can reach 0.9964 on the Netflix dataset, but the recall is only 0.5327. The possible reason is that DCEDM uses the base clustering results produced with the K-means clustering algorithms to reconstruct user-user graph and might divide small shilling groups and genuine users together. In the Netflix dataset, there are 64 shilling groups that include 32 small shilling groups. In each small shilling group, there is about 5 attackers. Therefore, these small shilling groups are easily divided together with genuine users and identified as genuine users. On the Amazon dataset, the F1-measure of DCEDM is better than that on the Netflix dataset, because a relatively small number of attackers are misclassified.

Similar to DCEDM, the performance of TP-GBF is also affected by the size of shilling groups. TP-GBF selects users with the maximum local topology potential to construct suspicious groups. The greater the size of shilling groups is, the larger the topology potentials of attackers are. Therefore, attackers in the large shilling groups are more likely to be divided into suspicious groups. As a result, the precision of TP-GBF is higher than the recall on the Netflix and Amazon datasets.

As shown in Table 6, the three metrics of GD-BKM on the Netflix dataset are close to 0.99, which are much better than those on the Amazon dataset. This is not hard to explain. In the Netflix dataset, the rating time of attackers in the same group is concentrated within a period of time, while the rating time of attackers in the Amazon dataset is relatively scattered.

As shown in Table 6, the three metrics of Codetector on the Netflix dataset are lower than those on the Amazon dataset. This is because the ratio of attackers in the Netflix dataset is about 22% while the ratio of attackers in the Amazon dataset is about 38%. As a supervised method, CoDetector can use more attackers to train the model on the Amazon dataset so that it has better detection performance on this dataset.

As shown in Table 6, the detection precision of GAGE can reach 0.9961 on the Netflix dataset, but the recall is only 0.4692. The possible reason is that GAGE uses the K-means++ clustering algorithm to divide candidate groups and the attackers of small shilling groups are easily divided together with normal user groups. As a result, the candidate groups that contain more genuine users and a few attackers are identified as the normal user groups, leading to a low recall on the Netflix dataset. While there are also some small shilling groups in the Amazon dataset, the proportion of these groups is very low. Therefore, GAGE has good precision and recall on the Amazon dataset.

As can be seen in Table 6, on the Movielens 1 M dataset, NFGCN-TIA still has the best detection performance among eight methods

while its precision, recall, and F1-measure values have a slight decrease in comparison with those on the Netflix dataset. This indicates that the change of target item selection strategy (i.e., half of the target items promoted by shilling groups are selected from popular items, instead all the target items are selected from unpopular items) has no great influence on the detection performance of NFGCN-TIA. In addition, by analyzing the datasets, we found that the rating time interval of each item in the Movielens 1 M dataset is smaller than that in the Netflix dataset. The small rating time interval of items can lead to high rating time similarity between genuine users and attackers on the co-rated items, making it possible to misjudge some attackers as genuine users. As a result, the detection performance of NFGCN-TIA on the Movielens 1 M dataset is slightly lower than that on the Netflix dataset. Except for NFGCN-TIA, other seven methods perform poorly on the Movielens 1 M dataset, indicating that the change of target item selection strategy and the rating time of items in the Movielens 1 M dataset is relatively close have a great influence on their detection performance. Take GAGE for an example, as half of target items promoted by the shilling groups are selected from the popular items in the Movielens 1 M dataset, the embeddings of genuine users who have rated these popular items are similar to those of attackers. Therefore, these genuine users may be divided into shilling groups by GAGE, leading to very poor detection precision on the Movielens 1 M dataset.

Based on the analyses above, it can be concluded that NFGCN-TIA has better performance than FAP, UD-HMM, DCEDM, TP-GBF, GD-BKM, CoDetector, and GAGE when detecting the traditional attack model-based shilling groups in the Netflix, Amazon, and Movielens 1 M datasets.

5.5.2. Comparison of performance in detecting new attack model-based shilling groups

In recent years, some deep learning-based shilling attack models (Barbieri et al., 2021; Wu et al., 2021) have been proposed to bias the output of recommender systems. These new shilling attack models use deep learning techniques to generate shilling profiles that are highly similar to normal profiles in recommender systems, which are difficult to detect by existing shilling attack detection methods. To show whether the proposed approach is effective for detecting shilling groups generated from deep learning-based shilling attack models, we select a graph convolution-based generative shilling attack (GOAT) model (Wu et al., 2021) as the basic model to generate shilling groups. GOAT uses a generative adversarial network integrated with a graph convolution structure to learn the real rating distribution to generate fake ratings. We employ $GSAGen_l$ GOAT to generate 8 shilling groups that include a total of 432 attackers and inject them into the sampled Netflix dataset (including 215,844 ratings from 2000 users on 3985 movies) to obtain a new synthetic dataset (named Netflix-1) for experimental evaluation. Table 7 shows the precision, recall, and F1-measure of eight methods on the Netflix-1 dataset.

As shown in Table 7, FAP, UD-HMM, DCEDM, TP-GBF, CoDetector, and GAGE perform poorly when detecting shilling groups generated from the GOAT attack model. This is because the fake profiles generated with the GOAT attack model are highly similar to normal profiles. While NFGCN-TIA and GD-BKM perform much better than other six methods on the Netflix-1 dataset, their precision, recall, and F1-measure are lower than those on the Netflix dataset (see Table 6). This indicates that the shilling groups generated from the deep learning-based shilling attack models are more difficult to detect than those generated from the traditional shilling attack models.

5.6. Analysis of parameter

To answer the third research question (RQ3), we conduct experiments on three datasets to analyze the impact of neighbor filtering threshold parameter on the detection performance of NFGCN-TIA.

Parameter θ of NFGCN-TIA is used as a threshold to filter the edges between attacker nodes and genuine user nodes in the suspicious user relation graph in feature aggregation, which is set by experiment. Fig. 3 shows the impact of parameter θ on the F1-measure of NFGCN-TIA on the Netflix, Amazon, and Movielens 1 M datasets.

As shown in Fig. 3, the F1-measure of NFGCN-TIA on the Netflix and Movielens 1 M datasets has a steady increase with the increase in θ and reaches the optimal value when θ is equal to 0.9. Hence, parameter θ is set to 0.9 on the Netflix and Movielens 1 M datasets. On the Amazon dataset, the F1-measure of NFGCN-TIA decreases gradually as θ increases from 0.1 to 0.3 and then tends to increase as θ increases from 0.3 to 0.4. Subsequently, the F1-measure of NFGCN-TIA decreases gradually as θ increases from 0.4 to 0.9. Thus, parameter θ is set to 0.1 on the Amazon dataset.

It can be seen from Fig. 3 that parameter θ shows a significant different impact on the F1-measure of NFGCN-TIA on the Netflix (Movielens 1 M) and Amazon datasets. This phenomenon is mainly attributed to the difference in sparsity between two datasets. The sparsity of the Netflix dataset is 97.16% while the sparsity of the Amazon dataset is 99.94%. Thus, on the Amazon dataset, a small value

Table 7
The detection performance of eight methods on the Netflix-1 dataset.

	precision	recall	F1-measure
FAP	0.6024	0.5718	0.5867
UD-HMM	0.3056	0.2873	0.2962
DCEDM	0.7053	0.4599	0.5568
TP-GBF	0.0124	0.0070	0.0089
GD-BKM	0.7963	0.8052	0.8007
CoDetector	0.4624	0.3094	0.3707
GAGE	0.4964	0.5327	0.5139
NFGCN-TIA	**0.8559**	**0.8341**	**0.8449**

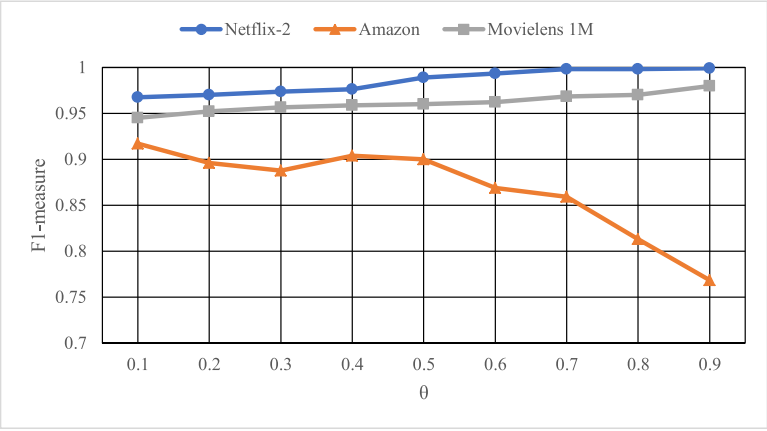

Fig. 3. The impact of θ on the F1-measure of NFGCN-TIA on three datasets.

Table 8
The running times of four GNN models on three datasets (seconds).

	Netflix			Amazon			Movielens 1M		
	Training time	Validating time	Testing time	Training time	Validating time	Testing time	Training time	Validating time	Testing time
GCN	2921	42	15	439	16	6	37,981	540	193
GAT	20,472	67	21	63,550	206	64	83,373	256	83
GraphSAGE	61.59	3.24	2.16	105.37	4.78	4.28	140.08	6.45	4.37
NFGCN	37.73	–	1.58	29.36	–	1.15	47.67	–	1.73

Table 9
The running times of eight detection methods on three datasets (seconds).

	Netflix	Amazon	Movielens 1M
FAP	74	541	206
UD-HMM	17	24	70
DCEDM	362	654	1136
TP-GBF	448	1963	3265
GD-BKM	779	3055	5749
CoDetector	886	4856	2594
GAGE	15,637	7918	26,594
NFGCN-TIA	63	132	196

of θ can effectively reduce the influence of genuine users on attackers. As θ increases, some useful structure information may be destroyed. On the Netflix (Movielens 1 M) dataset, a relatively larger θ value needs to be set. As a result, the impact of parameter θ on the Netflix (Movielens 1 M) and Amazon datasets shows a significant difference.

5.7. *Comparison of computational efficiency*

To test the second hypothesis (H2), we conduct experiments on three datasets to compare the computational efficiency of four graph neural network (GNN) models and eight detection methods. Tables 8 and 9 show the running times of four GNN models and eight detection methods on three datasets, respectively.

NFGCN is a semi-supervised classification model, which has no validating time, so we only give its training and testing times in Table 8. As can be seen in Table 8, NFGCN has obvious advantages in running time on three datasets. The running time of GCN and GAT models mainly depends on the number of edges in the suspicious user relation graph. Due to the tightly connected relations among attackers of shilling groups, the GCN and GAT models have higher training time, validating time and testing time on three datasets. The running time of GraphSAGE depends on the number of nodes in the graph. As the GraphSAGE model samples a fixed number of neighbors for each node, it has lower training time, validating time and testing time than the GCN and GAT models on three datasets. Among the four GNN models, NFGCN has the shortest training and testing times on three datasets, indicating that the computational cost can be significantly reduced by using the neighbor filtering mechanism.

It can be seen from Table 9 that UD-HMM has the shortest running time among eight detection methods. Nevertheless, the detection performance of UD-HMM on three datasets is much worse than that of NFGCN-TIA. Except for UD-HMM, NFGCN-TIA has the shortest

running time on three datasets, which again indicates that the computational cost can be significantly reduced with the neighbor filtering mechanism. Therefore, it is desirable to have a significant improvement in detection performance at the cost of a small amount of time.

5.8. Case study

To test the third hypothesis (H3), we use a case study to illustrate the effectiveness of NFGCN-TIA in a real recommendation scenario.

The proposed NFGCN-TIA can be used to detect and remove malicious user groups in the rating database of the recommender system before executing the specific recommendation algorithm. To show whether the proposed NFGCN-TIA is effective for defending group shilling attacks, we use the average prediction shift (PS) for all target items (I_T) to evaluate the effectiveness of using NFGCN-TIA in a real recommendation scenario, which is defined as follows (Mobasher et al., 2007):

$$PS = \frac{1}{|I_T|} \times \frac{1}{|U|} \times \left(\sum_{i \in I_T} \sum_{u \in U} \left| p'_{u,i} - p_{u,i} \right| \right) \tag{18}$$

where $p'_{u,i}$ and $p_{u,i}$ denote the pre- and post-attack predicted ratings for user u on item i, respectively.

We carry out experiment on the Netflix dataset and use the standard matrix factorization (MF) model (Koren et al., 2009) as the recommendation model for rating prediction. We extract 80% ratings of the Netflix dataset as the training set and the rest 20% ratings as the test set. The recommendation model is trained on the training set without shilling group detection and on the training set after detecting with NFGCN-TIA, respectively. We use the trained recommendation model to make rating predictions and calculate the average prediction shifts. Fig. 4 shows comparison of the average prediction shifts without shilling group detection and after detecting with NFGCN-TIA.

As shown in Fig. 4, the average prediction shift of the MF recommendation model is about 2.025 before shilling group detection, which means group shilling attacks cause the MF recommendation model to produce a large deviation in rating prediction. The average prediction shift of the MF recommendation model has a significant decrease after performing shilling group detection with NFGCN-TIA, which indicates that NFGCN-TIA can effectively detect shilling groups. Therefore, combining NFGCN-TIA with the existing recommendation algorithms can reduce the influence of group shilling attacks on the recommendation results and improve the robustness of recommendation algorithms and thus ensure the recommendation quality of recommender systems.

5.9. Discussion

Group shilling attacks pose a serious challenge to the security of recommender systems. Compared with the individual attackers, shilling groups have obvious collaborative behavior characteristics such as concentration of rating time and consistency of ratings for the target item(s). Therefore, how to accurately identify the collaborative behaviors among attackers is the key to successfully detect shilling groups.

In this research, we propose a deep learning-based approach for shilling group detection. On the one hand, we integrate a neighbor filtering mechanism with the standard GCN model to design a NFGCN model. The use of neighbor filtering mechanism can reduce the association between attackers and genuine users and reserve the association between attackers. This structural design of the NFGCN model has two outstanding characteristics in graph-based classification task. Firstly, neighbors who have different labels with the center node are filtered out when performing feature aggregation, so the classification performance of the NFGCN model can be improved significantly. The experimental comparison of four GNN models on the Netflix and Amazon datasets (see Table 5) shows that the NFGCN model has significant improvement over the GCN, GAT and GraphSAGE models in classification performance. Secondly, the NFGCN model employs a threshold parameter to filter the edges between attacker nodes and genuine user nodes in the suspicious user relation graph, so the computational cost of the NFGCN model is greatly reduced. The comparison of running times for four GNN models on the Netflix, Amazon and Movielens 1 M datasets (see Table 8) indicates that the NFGCN model has a significant advantage in computational efficiency compared with the GCN, GAT and GraphSAGE models.

On the other hand, we combine the NFGCN model with the target item identification method to propose a novel two-stage framework (i.e., NFGCN-TIA) for shilling groups detection. NFGCN-TIA first learns the user features from the suspicious user relation graph and combines with the user rating features refined with PCA to classify users, and then identifies shilling groups according to the user classification results and the attacked items. From the experimental results of NFGCN-TIA and baseline methods on the Netflix, Amazon, and Movielens 1 M datasets, it can be concluded that: 1) combining both structure and rating features of users can yield better user classification results and therefore improves the detection performance of NFGCN-TIA; 2) automatic feature extraction can deal with various attack strategies and has better generalization ability than manual feature extraction; 3) the detection performance and computational efficiency of NFGCN-TIA can be improved by selecting the proper neighbor filtering threshold.

There are two lessons can be learned from this research. Firstly, sampling neighbor nodes that have same label with the center node can improve the classification performance of the GCN model. Secondly, learning features automatically can enhance the universality of detection methods. These are the strong points of the proposed approach, which can be applied to the constructions of new approaches in the future.

There are also two limitations need to be solved in the future work. Firstly, in our approach, we use a threshold parameter to filter

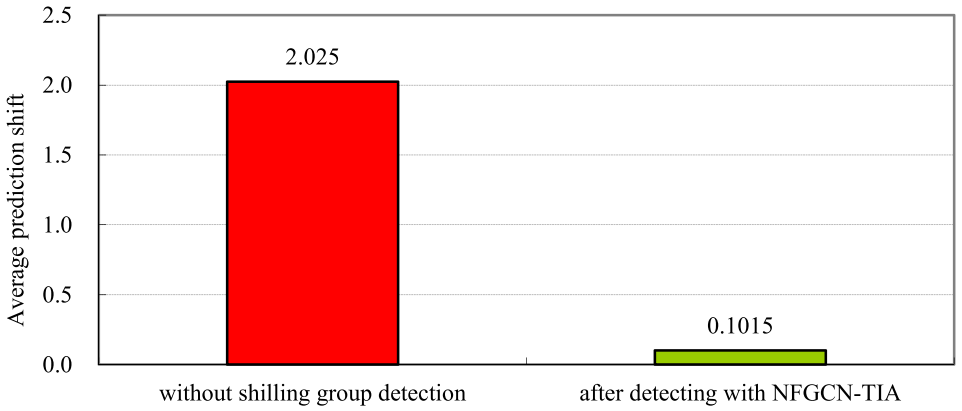

Fig. 4. Comparison of the average prediction shifts without shilling group detection and after detecting with NFGCN-TIA.

out neighbor nodes, which needs to be set by experiment. How to automatically set the threshold parameter according to the rating dataset is worth further research. Secondly, the NFGCN model is a transductive learning model. When adding new nodes in the user relation graph, it needs to be retrained. Therefore, how to construct a dynamic graph-oriented GNN model for user classification is also worth investigating.

6. Conclusions and future work

With the wide application of recommender systems, how to find out the group attackers with cooperative behavior from a large number of users is an important topic related to the credibility of recommender systems. To defend recommender systems, a group shilling attack detection method based on GCN and target item identification has been proposed. In this method, a neighbor node filtering mechanism has been integrated into the standard GCN to obtain the NFGCN model with a two-layer activation structure. Experimental results have indicated that the NFGCN model has better classification performance than other graph neural network models. By combining the NFGCN model and target item identification, we have proposed a shilling group detection framework. Extensive experiments have demonstrated the effectiveness of the proposed framework in detecting shilling groups in recommender systems. Our work improves the detection performance of shilling groups in the recommender systems. Combining the proposed detection method with existing recommendation algorithms can effectively improve the robustness of the recommendation algorithms and ensure the credibility of the recommendation results.

With the development of shilling attack strategies, deep learning-based shilling attack models against recommender systems are emerging (Barbieri et al., 2021; Wu et al., 2021). Such shilling attacks can better imitate the rating behaviors of genuine users in recommender systems and are more difficult to detect. In our future work, we will investigate deep learning-based shilling attacks in recommender systems and explore the effective way to detect such attacks.

CRediT authorship contribution statement

Shilei Wang: Methodology, Writing – original draft. **Peng Zhang:** Methodology, Validation. **Hui Wang:** Software. **Hongtao Yu:** Investigation. **Fuzhi Zhang:** Conceptualization, Supervision, Funding acquisition, Project administration, Writing – review & editing.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61772452, 62072393).

References

Barbieri, J., Alvim, L. G. M., Braida, F., & Zimbrão, G. (2021). Simulating real profiles for shilling attacks: A generative approach. *Knowledge-Based Systems, 230*, Article 107390. https://doi.org/10.1016/j.knosys.2021.107390

Batmaz, Z., Yilmazel, B., & Kaleli, C. (2020). Shilling attack detection in binary data: A classification approach. *Journal of Ambient Intelligence and Humanized Computing, 11*(6), 2601–2611. https://doi.org/10.1007/s12652-019-01321-2

Burke, R., Mobasher, B., Williams, C., & Bhaumik, R. (2006). Classification features for attack detection in collaborative recommendation systems. In *Proceedings of the 12th international conference on knowledge discovery and data mining* (pp. 542–547). https://doi.org/10.1145/1150402.1150465

Cai, H., & Zhang, F. (2019). Detecting shilling attacks in recommender systems based on analysis of user rating behavior. *Knowledge-Based Systems, 177*, 22–43. https://doi.org/10.1016/j.knosys.2019.04.001

Cai, H., & Zhang, F. (2021b). An unsupervised approach for detecting group shilling attacks in recommender systems based on topological potential and group behaviour features. *Security and Communication Networks, 18*. https://doi.org/10.1155/2021/2907691

Cai, H., & Zhang, F. (2021a). BS-SC: An unsupervised approach for detecting shilling profiles in collaborative recommender systems. *IEEE Transactions on Knowledge and Data Engineering, 33*(4), 1375–1388. https://doi.org/10.1109/tkde.2019.2946247

Dou, T., Yu, J., Xiong, Q., Gao, M., Song, Y., & Fang, Q. (2018). Collaborative Shilling Detection Bridging Factorization and User Embedding. In *Proceedings of the 13th European Alliance for Innovation (EAI) international conference on collaborative computing - networking, applications and worksharing (CollaborateCom)* (pp. 459–469). https://doi.org/10.1007/978-3-030-00916-8_43

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* (pp. 1025–1035). https://dl.acm.org/doi/abs/10.5555/3294771.3294869.

Hao, Y., & Zhang, F. (2021). An unsupervised detection method for shilling attacks based on deep learning and community detection. Soft Computing, 25 (1), 477–494. 10.1007/s00500-020-05162-6.

Hazrati, N., & Ricci, F. (2021). Recommender systems effect on the evolution of users' choices distribution. *Information Processing and Management, 59*, Article 102766. https://doi.org/10.1016/j.ipm.2021.102766

Hurley, N., Cheng, Z., & Zhang, M. (2009). Statistical attack detection. In *Proceedings of the 3rd international conference on recommender systems* (pp. 149–156). https://doi.org/10.1145/1639714.1639740

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th international conference on learning representations* (pp. 1–14). https://doi.org/10.48550/arXiv.1609.02907

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM, 53*(4), 89–97. https://doi.org/10.1145/1721654.1721677

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer, 42*(8), 30–37. https://doi.org/10.1109/MC.2009.263

Lee, J., & Zhu, D. (2012). Shilling attack detection–A new approach for a trustworthy recommender system. *Informs Journal on Computing, 24*(1), 117–131. https://doi.org/10.1287/ijoc.1100.0440

Li, H., Gao, M., Zhou, F., Wang, Y., Fan, Q., & Yang, L. (2021). Fusing hypergraph spectral features for shilling attack detection. *Journal of Information Security and Applications, 63*, Article 103051. https://doi.org/10.1016/j.jisa.2021.103051

Li, W. T., Gao, M., Li, H., Xiong, Q., Wen, J., & Ling, B. (2015). An shilling attack detection algorithm based on popularity degree features. *Zidonghua Xuebao/Acta Automatica Sinica, 41*(9), 1563–1575. https://doi.org/10.16383/j.aas.2015.c150040

Mehta, B., & Nejdl, W. (2009). Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User Adapted Interaction, 19*(1–2), 65–97. https://doi.org/10.1007/s11257-008-9050-4

Mobasher, B., Burke, R., Bhaumik, R., & Williams, C. (2007). Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology, 7*(4), 38. https://doi.org/10.1145/1278366.1278372

Seminario, C. E., & Wilson, D. C. (2014). Attacking item-based recommender systems with power items. In *Proceedings of the 8th ACM conference on recommender systems* (pp. 57–64). https://doi.org/10.1145/2645710.2645722

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

Slokom, M., Hanjalic, A., & Larson, M. (2021). Towards user-oriented privacy for recommender system data: A personalization-based approach to gender obfuscation for user profiles. *Information Processing and Management, 58*(6), Article 102722. https://doi.org/10.1016/j.ipm.2021.102722

Smith, B., & Linden, G. (2017). Two decades of recommender systems at amazon.com. *IEEE Computer Society, 21*(3), 12–18. https://doi.org/10.1109/MIC.2017.72

Su, X. F., Zeng, H. J., & Chen, Z. (2005). Finding group shilling in recommendation system. In *Proceedings of the special interest tracks and posters of the 14th international conference on world wide web* (pp. 960–961). https://doi.org/10.1145/1062745.1062818

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Li`o, P., & Bengio, Y. (2018). Graph attention networks. In *Proceedings of the 6th international conference on learning representations* (pp. 1–12). https://doi.org/10.48550/arXiv.1710.10903

Wang, S., Wang, H., Yu, H., & Zhang, F. (2021). Detecting shilling groups in recommender systems based on hierarchical topic model. In *Proceedings of IEEE international conference on artificial intelligence and computer applications* (pp. 832–837). https://doi.org/10.1109/icaica52286.2021.9498071

Wang, W., Zhang, G., & Lu, J. (2016a). Member contribution-based group recommender system. *Decision support systems, 87*, 80–93. https://doi.org/10.1016/j.dss.2016.05.002

Wang, Y., Wu, Z., Bu, Z., Cao, J., & Yang, D. (2016b). Discovering shilling groups in a real e-commerce platform. *Online Information Review, 40*(1), 62–78. https://doi.org/10.1108/OIR-03-2015-0073

Wang, Y., Wu, Z., Cao, J., & Fang, C. (2012). Towards a tricksy group shilling attack model against recommender systems. In *Proceedings of the international conference on advanced data mining and applications* (pp. 675–688). https://doi.org/10.1007/978-3-642-35527-1_56

Wu, F., Gao, M., Yu, J., Wang, Z., Liu, K., & Wang, X. (2021). Ready for emerging threats to recommender systems? A graph convolution-based generative shilling attack. *Information Sciences, 578*, 683–701. https://doi.org/10.1016/j.ins.2021.07.041

Xia, H., Fang, B., Gao, M., MA, H., Tang, Y., & Wen, J. (2015). A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique. *Information Sciences, 306*, 150–165. https://doi.org/10.1016/j.ins.2015.02.019

Xu, C., Zhang, J., Chang, K., & Chong, L. (2013). Uncovering collusive spammers in Chinese review websites. In *Proceedings of the 22nd ACM international conference on information and knowledge management* (pp. 979–988). https://doi.org/10.1145/2505515.2505700

Xu, Y., & Zhang, F. (2019). Detecting shilling attacks in social recommender systems based on time series analysis and trust features. *Knowledge-Based Systems, 178*, 25–47. https://doi.org/10.1016/j.knosys.2019.04.012

Yang, Z., & Cai, Z. (2017). Detecting abnormal profiles in collaborative filtering recommender systems. *Journal of Intelligent Information Systems, 48*(3), 499–518. https://doi.org/10.1007/s10844-016-0424-5

Yang, Z., Cai, Z., & Guan, X. (2016). Estimating user behavior toward detecting anomalous ratings in rating systems. *Knowledge-Based Systems, 111*, 144–158. https://doi.org/10.1016/j.knosys.2016.08.011

Yu, H., Yuan, S., Xu, Y., Ma, R., Gao, D., & Zhang, F. (2021b). Group attack detection in recommender systems based on triangle dense subgraph mining. In *Proceedings of IEEE international conference on artificial intelligence and computer applications* (pp. 649–653). https://doi.org/10.1109/icaica52286.2021.9497958

Yu, H., Zheng, H., Xu, Y., Ma, R., Gao, D., & Zhang, F. (2021a). Detecting group shilling attacks in recommender systems based on maximum dense subtensor mining. In *Proceedings of IEEE international conference on artificial intelligence and computer applications* (pp. 644–648). https://doi.org/10.1109/ICAICA52286.2021.9498095

Zhang, F., Qu, Y., Xu, Y., & Wang, S. (2020). Graph embedding-based approach for detecting group shilling attacks in collaborative recommender systems. *Knowledge-Based Systems, 199*, Article 105984. https://doi.org/10.1016/j.knosys.2020.105984

Zhang, F., & Wang, S. (2020). Detecting group shilling attacks in online recommender systems based on bisecting k-means clustering. IEEE Transactions on Computational Social Systems, 7(5), 1189–1199. 10.1109/TCSS.2020.3013878.

Zhang, F., Zhang, Z., Zhang, P., & Wang, S. (2018). UD-HMM: An unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering. *Knowledge-Based Systems, 148*, 146–166. https://doi.org/10.1016/j.knosys.2018.02.032

Zhang, F., & Zhou, Q. (2014). HHT-SVM: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems, 65*, 96–105. https://doi.org/10.1016/j.knosys.2014.04.020

Zhang, Y., Tan, Y., Zhang, M., Liu, Y., Tat-Seng, C., & Ma, S. (2015). Catch the black sheep: Unified framework for shilling attack detection based on fraudulent action propagation. In *Proceedings of the 24th international conference on artificial intelligence* (pp. 2408–2414). https://dl.acm.org/doi/abs/10.5555/2832581.2832585.

Zhang, Z., & Kulkarni, S. R. (2013). Graph-based detection of shilling attacks in recommender systems. In *Proceedings of IEEE international workshop on machine learning for signal processing* (pp. 1–6). https://doi.org/10.1109/MLSP.2013.6661953

Zhang, Z., & Kulkarni, S. R. (2014). Detection of shilling attacks in recommender systems via spectral clustering. In *Proceedings of the 17th international conference on information fusion* (pp. 1–8).

Zhou, Q., & Duan, L. (2021). Semi-supervised recommendation attack detection based on Co-Forest. *Computer & Security, 109*, Article 102390. https://doi.org/10.1016/j.cose.2021.102390

Zhou, W., Koh, Y. S., Wen, J., Alam, S., & Dobbie, G. (2014). Detection of abnormal profiles on group attacks in recommender systems. In *Proceedings of the 37th international ACM conference on research & development in information retrieval* (pp. 955–958). https://doi.org/10.1145/2600428.2609483

Zhou, W., Wen, J., Xiong, Q., Gao, M., & Zeng, J. (2016). SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems. *Neurocomputing, 210*, 197–205. https://doi.org/10.1016/j.neucom.2015.12.137