Hindawi
Security and Communication Networks
Volume 2022, Article ID 2323132, 16 pages
https://doi.org/10.1155/2022/2323132

Research Article

A Shilling Group Detection Framework Based on Deep Learning Techniques

Shilei Wang, [1,2] **Peng Zhang**, [3] **Hui Wang**, [1,2] **Jinbo Chao**, [1,2] **and Fuzhi Zhang** [1,2]

[1] *School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei, China*
[2] *The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao, Hebei, China*
[3] *School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China*

Correspondence should be addressed to Fuzhi Zhang; xjzfz@ysu.edu.cn

Received 22 March 2022; Accepted 24 May 2022; Published 30 June 2022

Academic Editor: Marimuthu Karuppiah

Research studies have shown that online recommender systems are under the threat of group shilling attacks, in which attackers attempt to distort the recommendation results of particular items by cooperatively injecting fake profiles. Existing detection methods usually divide candidate groups at first and then use the hand-crafted features to recognize shilling groups. However, the detection performance of existing methods depends highly on the quality of candidate groups obtained. Moreover, extracting features manually is time-consuming. To overcome these limitations, we propose a shilling group detection framework based on the sparse autoencoder and modified GraphSAGE model. First, we use the sparse autoencoder to obtain rating features of users from the rating dataset. Second, we analyse user collusive degrees to calculate user transition probabilities. Third, we build a user relation graph and utilize the modified GraphSAGE model to perform user classification. Finally, shilling groups are gathered according to the neighbour relations. Extensive experiment demonstrates that the proposed framework performs better on different datasets than baseline methods.

1. Introduction

With the development of the information technology and Internet technology, people have gotten through the times of information deficit and stepped into the age of information overload [1]. Under this situation, it is becoming increasingly difficult for users to obtain useful information from a mass of information. Recommender systems can serve users with fast and accurate personalized recommendations according to the users' historical behaviour records, which alleviates the inconvenience caused by information overload to a large extent. Therefore, recommender systems have been widely used in various platforms and have brought huge profits to these platforms [2]. As reported, recommender systems bring 35% sales of Amazon, 75% consumption of Netflix, and 65% page view of YouTube. However, recommender systems are open to users due to the need of user information collection, which makes it vulnerable to malicious attacks. Consequently, how to detect malicious attacks becomes a key issue to maintain high-quality recommendations and the robustness of recommender systems.

Driven by interests, malicious users inject fake profiles into the recommender system in an attempt to twist the recommendation results, which is called a shilling attack [3]. In shilling attacks, attackers rate not only target items but also some normal items to camouflage their fraudulent behaviours. According to the selection strategy of camouflage items, shilling attacks are divided into individual shilling attacks and group shilling attacks. In the individual shilling attacks, the selection of camouflage items is random. In the group shilling attacks, attackers choose as many different camouflage items as possible for the purpose of reducing the similarities between their rating profiles. Therefore, group shilling attacks are more difficult to be detected than individual shilling attacks. To protect recommender systems from shilling attacks, great efforts have been made by researchers. Existing detection methods mainly focus on individual shilling attacks. While some methods have recently been put forward to detect group shilling attacks, they have the following two limitations. Firstly, the detection performance of the existing methods depends highly on the quality of generated candidate groups. As these methods mainly use the frequent itemset mining or clustering techniques to generate candidate groups, they are difficult to obtain high-quality candidate groups due to the sensitivity of support threshold or clustering parameters settings. Secondly, the existing methods rely on manual feature engineering to extract detection features, which is time-consuming.

In order to overcome these limitations, we design a shilling group detection framework based on sparse autoencoder and graph probability sampling and aggregating. We first analyse the collusive degree of user rating behaviours and calculate user transition probabilities as a reference for sampling user neighbours. Then, we employ the sparse autoencoder to automatically extract the user rating features from the rating dataset. Finally, we aggregate the features of user neighbours according to user transition probabilities to predict user labels. The sampled neighbour relations are also used to divide suspicious users into shilling groups.

The contributions of this paper are as follows:

(1) We improve the sampling mechanism of GraphSAGE model. The standard GraphSAGE model samples neighbours of central node randomly and has low stability. We calculate user transition probabilities based on rating values, rating times, and rating numbers and design an ordered loop sampling without replacement. The improved sampling mechanism can reduce the randomness of sampling results and optimize classification performance of the GraphSAGE model.

(2) We put forward a shilling group detection framework based on Sparse Autoencoder and Graph Probability Sampling and Aggregating, which is called SA-GPSA. SA-GPSA achieves the automatic extraction of user features and can adjust user features according to partial user information in the dataset. It is flexible with different datasets and has high universality.

(3) We carry out experiments to demonstrate the effectiveness of our method. First, we compare the sample results and classification performance of the GraphSAGE before and after improvement. Then, SA-GPSA is compared with six baseline methods in detection performance.

2. Background and Related Work

2.1. Threat Models. In recommender systems, shilling profiles are generated by specific methods which are defined as threat (attack) models. A shilling profile is a rating vector containing the selected items (I_S), the filler items (I_F), the target items (I_T), and some unrated items ($I_\varnothing$) [4]. The threat models are divided into two broad categories: individual shilling attack models and group shilling attack models. According to the generation methods, individual shilling attack models can be divided into traditional attack models and deep learning-based attack models. Traditional attack models are based on statistics methods, such as random (Ran) attack model, average (Avg) attack model, average over popular (AoP) items attack model, and power item attack (PIA) model. Deep learning-based attack models use deep learning methods to construct shilling profiles by simulating rating profiles of genuine users, such as Graph cOnvolution based generative shilling ATtack (GOAT) model [5].

Wang et al. [6] presented a Group Shilling Attack Generation model (*GSAGen*), including a loose version ($GSAGen_l$) and a strict version ($GSAGen_s$). The basic threat models used by them are random attack model and average attack model. In order to enrich attack types and make attack profiles more complicated, we also choose the AoP attack model, PIA model, and GOAT attack model as the basic models to generate shilling group profiles. As $GSAGen_l$ can generate larger shilling groups than $GSAGen_s$, we select $GSAGen_l$ to construct shilling groups in this paper. Table 1 shows the details of threat models used in this paper.

2.2. Related Work. Existing detection methods are mainly for two broad categories of shilling attacks, i.e., detection of individual shilling attacks and detection of group shilling attacks. In this section, we review related studies from these two categories.

The detection methods of individual shilling attacks include supervised methods, semi-supervised methods, and unsupervised methods. Supervised methods usually train a classifier to distinguish genuine users from attackers [7]. Based on the difference of rating distribution, Chirita et al. [8] proposed DegSim and RDMA to detect random attack and average attack. Burke et al. [9] summarized some characteristics on the basis of rating deviation and utilized these features to train a classifier for detecting attackers. Li et al. [10] put forward some features based on popularity of users and items and employed C4.5 to detect attackers. Liu et al. [11] proposed a detection method based on Kalman filter. They revealed abnormal time period to identify attackers. Tong et al. [12] proposed a convolutional neural network-based method to detect shilling attacks. Their method extracted deep features from user rating profiles and could precisely detect majority of obfuscated attacks. Li et al. [13] proposed a model to detect shilling attacks by fusing hypergraph spectral features. They employed the extracted explicit and implicit features to train a deep neural network for distinguishing attackers. Vivekanandan and Praveena [14] constructed a deep learning model to extract deep-level features from user profiles and used the deep-level features to detect shilling attacks. Supervised methods need to manually label training samples, which is costly and time-consuming. In order to overcome this limitation, unsupervised methods were used for detecting shilling attacks. In the unsupervised detection methods, the user behaviour features are extracted and the clustering algorithms are used to group attackers together [15–21]. There are also detection

Table 1: Summary of threat models.

Threat model	I_S Items	I_S Ratings	I_F Items	I_F Ratings	I_T Rating
$GSAGen_l$ Ran	Null		Item is randomly chosen and rated by at most two attackers	System mean	$r_{\max}$ or $r_{\min}$
$GSAGen_l$ Avg	Null		Item is randomly chosen and rated by at most two attackers	Item mean	$r_{\max}$ or $r_{\min}$
$GSAGen_l$ AoP	Null		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	Items are randomly chosen from power items and rated by less than three attackers	Item mean	Null		$r_{\max}$ or $r_{\min}$
$GSAGen_l$ GOAT	Items are randomly chosen from popular items rated by active users and rated by less than three attackers	Learned by GAN	Items are randomly chosen from popular items rated by active users and rated by less than three attackers	Learned by GAN	$r_{\max}-1$

methods that model the dataset as a graph and use the graph mining methods to identify attackers [22–25]. The unsupervised methods are regardless of specific attack types, but they usually need priori knowledge. Wu et al. [26–28] used a small number of labelled data to train a Naive Bayes classifier and used the unlabelled data to optimize the classifier. These semi-supervised methods are effective in detecting hybrid and obfuscated attacks, but the overall performance is not very high due to the limited labels. Zhou and Duan [29] also proposed a semi-supervised method to detect shilling attacks. They extracted a series of features to train an ensemble classifier for shilling attack detection. This method is effective for specific attack strategy, but it requires setting more parameters.

The above-mentioned methods are effective for detecting individual shilling attacks. As these methods do not take the cooperation of attackers into consideration, they are not suitable for group attacks. Over the past few years, some approaches for detecting group shilling attacks have been proposed. Zhou et al. [30] combined the modified DegSim and RDMA to measure the suspicious degrees of users and utilized the k-means clustering and target item identification to find shilling groups. Wang et al. [31] first transformed the rating dataset to market basket transactions and performed frequent itemset mining (FIM) to generate candidate groups. Then, they extracted six group features and utilized a PCA-based approach to calculate ranking scores of candidate groups. Finally, a ranked candidate groups was obtained. Zhang and Wang [32] proposed a shilling group detection approach with bisecting K-means clustering. In their method, candidate groups were divided in terms of rating times and group suspicious degrees were calculated from the item view and the user view. Then, the bisecting K-means algorithm was used to obtain shilling groups. Zhang et al. [33] proposed a shilling group detection method based on graph embedding. They first used Node2vec to get user embeddings and employed k-means++ to divide candidate groups. Then, some features are used to measure group suspicious degrees and the hierarchical clustering algorithm was used to obtain shilling groups. Cai and Zhang [34] constructed a weight user relation graph by computing user collusive degrees and discovered suspicious groups through topological potential analysis. Then, they used the clustering algorithm to obtain shilling groups. Yu et al. [35] employed the M-zoom model to obtain the maximum subtensors of each item and a density threshold is used to select candidate groups from these subtensors. Then, they designed a dual-input convolutional network to extract user features and identify group attackers. Wang et al. [36] employed a hierarchical topic model to generate user rating vectors and divided candidate groups according to rating values and rating times and detected shilling groups with the k-means algorithm.

3. The Framework of SA-GPSA

To detect shilling groups in recommender systems, we propose a detection framework based on sparse autoencoder and graph probability sampling and aggregating (SA-GPSA). The framework of SA-GPSA is shown in Figure 1.

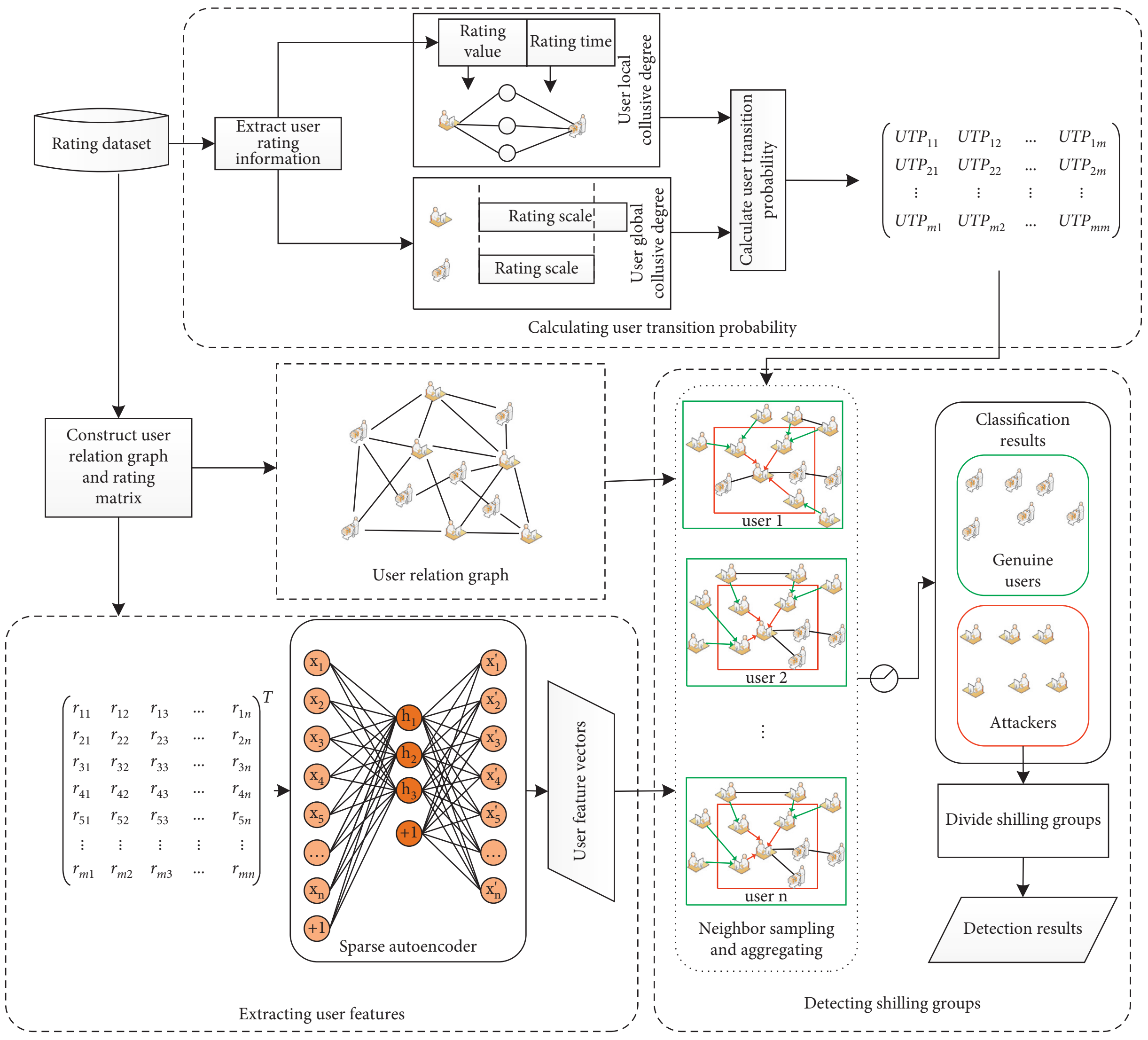
As shown in Figure 1, SA-GPSA first analyses user local collusive degrees and global collusive degrees to calculate user transition probabilities and uses the sparse autoencoder to extract user features from the rating dataset. Then, a user relationship graph is constructed according to co-rated items. Based on the user transition probabilities and user features, it performs neighbour sampling and neighbour features aggregation on the user relation graph to classify users. Finally, the neighbour relations are used to divide suspicious users into different shilling groups.

For the sake of discussion, Table 2 provides the notations and their descriptions.

3.1. Calculating User Transition Probabilities.

In this section, we calculate user transition probabilities to serve as the guidance for sampling user neighbours and dividing groups.

In a group shilling attack, the attackers cooperate with each other to attack the target item. In order to avoid detection, group attackers focus on increasing the similarity between their profiles and those of genuine users and decreasing the similarity between their own profiles. Thus, group attackers usually have many co-rated items with genuine users and a few co-rated items with their conspirators, which makes it difficult to reveal cooperation relations between group attackers from complicated user



Rating dataset
Extract user rating information
Rating value
Rating time
User local collusive degree
Rating scale
Rating scale
User global collusive degree
Calculate user transition probability
Calculating user transition probability
Construct user relation graph and rating matrix
User relation graph
user 1
user 2
user n
Neighbor sampling and aggregating
Classification results
Genuine users
Attackers
Divide shilling groups
Detection results
Detecting shilling groups
Sparse autoencoder
User feature vectors
Extracting user features

FIGURE 1: The framework of SA-GPSA.

TABLE 2: Notations.

Notations	Descriptions
U	Set of users
I	Set of items
u	A user
i	An item
$\mathbf{R}$	User rating matrix
$\mathbf{X}$	User feature matrix
r_{ui}	Rating of user u for item i
t_{ui}	Rating time of user u for item i
URI_u	Set of items rated by user u
$r_{\max}$	The maximum rating allowed in the dataset
$\lvert\cdot\rvert$	Number of elements in the set

relations. To solve this problem, we analyse user collusive degrees to calculate the user transition probabilities.

Firstly, the user collusive degrees are calculated according to local collusive degrees and global collusive degrees.

The local collusive degree is related to the deviation of rating values and rating time. For any two users $u, v \in U$, the set of items co-rated by users u and v is as follows:

$$CoRI_{uv} = \{i \mid i \in I \wedge r_{ui} \neq 0 \wedge r_{vi} \neq 0\}. \quad (1)$$

From the view of rating value, if two users give the highest rating for the same item, the two users have a high probability of cooperating to attack this item.

Definition 1 (Rating collusive degree, RCD). For users $u, v \in U$, the rating collusive degree of u and v on $i \in CoRI_{uv}$ is defined as

$$RCD_{uv}^{i} = \frac{r_{ui} + r_{vi}}{2 \times r_{\max}}. \quad (2)$$

From the view of rating time, if users $u, v \in U$ rate the same item in a short time period, users u and v have a high probability of cooperating to attack this item. We design a nonlinear function to measure the rating time interval (*TI*); that is,

$$TI_{uv}^{i} = \begin{cases} 1, & \left|t_{ui} - t_{vi}\right| \le \mu_t, \\ \dfrac{\left|t_{ui} - t_{vi}\right|}{30}, & \left|t_{ui} - t_{vi}\right| > \mu_t, \end{cases} \tag{3}$$

where μ_t is a time threshold and is set to 30 days.

Group attackers usually have high *RCD*s and short *TI*s on the target items. For users $u, v \in U$, the collusive degree of u and v on $i \in CoRI_{uv}$ is defined as

$$UCD_{uv}^{i} = \frac{RCD_{uv}^{i}}{TI_{uv}^{i}}. \tag{4}$$

The greater the UCD_{uv}^{i} is, the more likely users u and v cooperate to attack item i.

In a group shilling attack, attackers have more than one co-rated item, but they only care about the target item promoted. Therefore, group attackers should have the highest collusive degree on the target item.

Definition 2 (Local collusive degree, *LocalC*). For users $u, v \in U$, the local collusive degree of u and v is defined as the highest collusive degree on the co-rated items; that is,

$$LocalC_{uv} = \max_{i \in CoRI_{uv}} \left(UCD_{uv}^{i}\right). \tag{5}$$

Definition 3 (global collusive degree, GlobalC) For any two users $u, v \in U$, the global collusive degree of users u and v is defined as the similarity between their rating scales, which is calculated by

$$GlobalC_{uv} = 1 - \frac{\left|\left|URI_u\right| - \left|URI_v\right|\right|}{\left|URI_u\right| + \left|URI_v\right|}. \tag{6}$$

Attackers in the shilling group have the same rating polarity on the target item and at the same time select many filler items to rate. Thus, the global collusive degree between attackers is high.

Definition 4 (User collusive degree, *UCD*). For users $u, v \in U$, the collusive degree of u and v is defined as the product of their local collusive degree and global collusive degree; that is,

$$UCD_{uv} = LocalC_{uv} \times GlobalC_{uv}. \tag{7}$$

Secondly, the user transition probabilities are calculated based on user collusive degrees.

We model the dataset as a user relation graph by viewing users as nodes. For users $u, v \in U$, if u and v have co-rated the same item (s), there exists an edge connecting u and v . All users connected to user u constitute the neighbour set of user u, denoted as N_u.

Definition 5 (User transition probability, *UTP*). For any $v \in N_u$, the transition probability between u and v refers to probability that u cooperates with v, which is calculated by

$$UTP_{uv} = \frac{\exp\left(UCD_{uv}\right)}{\sum_{v\prime \in N_u} \exp\left(UCD_{uv\prime}\right)}. \tag{8}$$

According to (8), we can obtain the transition probabilities between users in the rating dataset and thus obtain a user transition probability matrix for the downstream tasks.

3.2. Extracting User Features.

3.2. Extracting User Features. We use the sparse autoencoder to extract user features.

For any user $u \in U$, the feature vector $\mathbf{x}_u = \left(r_{u1}, r_{u2}, \ldots, r_{un}\right)$ is first encoded:

$$\mathbf{h}_u = sigmoid\left(\mathbf{W}\mathbf{x}_u + b\right), \tag{9}$$

where $\mathbf{W}$ and b are the weight matrix and bias unit needed to be trained in the encoding process and $\mathbf{h}_u$ is a low-dimensional vector.

Then, $\mathbf{h}_u$ is decoded:

$$\widehat{\mathbf{x}}_u = sigmoid\left(\mathbf{W}'\mathbf{h}_u + b'\right), \tag{10}$$

where $\mathbf{W}'$ and b' are the weight matrix and bias unit needed to be trained in the decoding process. If $\widehat{\mathbf{x}}_u$ is similar or same to $\mathbf{x}_u$, $\mathbf{h}_u$ can replace $\mathbf{x}_u$ as the feature vector of u.

To make $\widehat{\mathbf{x}}_u$ as similar as possible to $\mathbf{x}_u$, a loss function is used to train parameters $\langle \mathbf{W}, \mathbf{W}', b, b' \rangle$. The loss function is

$$J_{\text{sparse}}\left(\mathbf{W}, b\right) = J\left(\mathbf{W}, b\right) + \beta \sum_{j=1}^{s_2} KL\left(\rho \| \widehat{\rho}_j\right), \tag{11}$$

where s_2 is the number of hidden neurons, β is the weight of the KL divergence, ρ is the expected degree of activity, and $\widehat{\rho}_j$ is the average degree of activity of neuron j. $J\left(\mathbf{W}, b\right)$ is the standard loss function and the latter term is a sparsity penalty factor, which are calculated by

$$J\left(\mathbf{W}, b\right) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2}\left\|a^{(3)}\left(\mathbf{x}_u^{(i)}\right) - \mathbf{x}_u^{(i)}\right\|^2\right), \tag{12}$$

$$KL\left(\rho \| \widehat{\rho}_j\right) = \rho \log \frac{\rho}{\widehat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \widehat{\rho}_j}, \tag{13}$$

where n is the number of neurons. Denoting $a_j^2\left(\mathbf{x}_u\right)$ as the degree of activity of neuron j when the input is $\mathbf{x}_u$, then the average degree of activity of neuron j can be denoted as

$$\widehat{\rho}_j = \frac{1}{n} \sum_{i=1}^{n} \left[a_j^{(2)}\left(\mathbf{x}_u^{(i)}\right)\right]. \tag{14}$$

In order to realize the sparse limitation, $\widehat{\rho}_j$ should meet the condition

$$\widehat{\rho}_j = \rho. \tag{15}$$

In the process of user feature extraction, ρ is set to 0.01, and β is set to 3.

With the loss function, the gradient descent is used to train parameters and the user feature matrix $\boldsymbol{H}$ is obtained. Algorithm 1 describes the process of extracting user features.

In Algorithm 1, the first part (line 1) performs the random initialization of weight matrices. The second part (lines 2–13) trains the sparse autoencoder to extract user features and obtain the user low-dimensional feature matrix $\mathbf{H}$.

3.3. Detection of Shilling Groups. To detect shilling groups, we design a graph probability sampling and aggregation model (GPSA) and use GPSA to classify users. Based on the user classification results, the user neighbour relations are utilized to divide shilling groups.

3.3.1. Construction of the GPSA Model. The standard GraphSAGE model [37] randomly samples neighbours to perform feature aggregation. However, in a shilling group, most neighbours of attackers are genuine users. As a result, the GraphSAGE model has a high probability to sample genuine neighbours of attackers and aggregate features of attackers with those of many genuine users. In addition, the sampling results of the GraphSAGE model have a high randomness. To solve these problems, we modify the sampling mechanism of the GraphSAGE model to design a two-layer model with a probability-based neighbour sampling mechanism, which is named GPSA.

Firstly, we establish the probability-based neighbour sampling mechanism.

The purpose of the sampling mechanism is to sample users with the same label to the central user. Therefore, we sample neighbours based on user transition probability. Particularly, for each $u \in U$, we first rank users in N_u according to transition probability in descending order. Then a sampling number K is set and we perform an ordered loop sampling without replacement. If $|N_u| > K$, sample the first K users in the sorted N_u; otherwise, repeat sampling process until K users are sampled. The sampling algorithm is described below.

In Algorithm 2, the first part (lines 1–10) calculates transition probabilities between users and constructs neighbours set for each user. The second part (lines 11–19) samples neighbours for each user according to the transition probability.

Secondly, we perform feature aggregation based on the sampling results to learn user embeddings.

In the first layer, for each user $u \in U$, calculate the average feature value of all neighbours in SN_u as a new neighbour feature:

$$\mathbf{h}_{SN_u}^{(1)} = \text{mean}\left(\left\{\mathbf{h}_v', \quad \forall v \in SN_u\right\}\right). \tag{16}$$

Then, add the new neighbour feature to the feature of u and utilize the weight matrix to reduce the dimension of the user feature:

$$\mathbf{h}_u^{(1)} = \text{Relu}\left(\mathbf{W}^{(1)}\text{sum}\left(\mathbf{h}_u^{(0)}, \mathbf{h}_{SN_u}^{(1)}\right)\right). \tag{17}$$

In the second layer, repeat the process of the first layer and get the normalized user embedding of u:

$$\mathbf{h}_{SN_u}^{(2)} = \text{mean}\left(\left\{\mathbf{h}_v^{(1)}, \quad \forall v \in SN_u\right\}\right), \tag{18}$$

$$\mathbf{h}_u^{(2)} = \mathbf{W}^{(2)}\text{sum}\left(\mathbf{h}_u^{(1)}, \mathbf{h}_{SN_u}^{(2)}\right), \tag{19}$$

$$\mathbf{z}_u = \frac{\mathbf{h}_u^{(2)}}{\left\|\mathbf{h}_u^{(2)}\right\|_2}. \tag{20}$$

In equations (16)–(20), Relu $(x) = \max\ (x, 0)$ is an activation function, *mean* and *sum* are the average and summation aggregator, and $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are weight matrices needed to learn.

For the constructed GPSA model, the vector dimensions of input layer, hidden layer, and output layer are 128, 64, and 2 separately.

3.3.2. User Classification and Shilling Group Division. After constructing the GPSA model, we use the supervised method to classify users. In order to train GPSA, we divide the rating dataset into the training set *Tr* and the test set *Te*. In this paper, the proportion of training set and test set is 6 : 4.

Based on $\mathbf{z}_u$, the softmax function is used to calculate the probability that u belongs to which label:

$$p_0^u = \frac{\exp\left(\mathbf{z}_u(1)\right)}{\exp\left(\mathbf{z}_u(1)\right) + \exp\left(\mathbf{z}_u(2)\right)}, \tag{21}$$

$$p_1^u = \frac{\exp\left(\mathbf{z_u}(2)\right)}{\exp\left(\mathbf{z}_u(1)\right) + \exp\left(\mathbf{z}_u(2)\right)}, \tag{22}$$

where $\mathbf{z}_u(1)$ is the first element in user feature vector and $\mathbf{z_u}(2)$ is the second element. If $p_0^u < p_1^u$, u is an attacker; otherwise, u is a genuine user.

For classification task, the cross-entropy loss function is usually used for parameters optimization, which is calculated by

$$\text{loss} = -\frac{1}{|Tr|}\sum_{u \in Tr}\left[l_1^u \times \log\left(p_0^u\right) + l_2^u \times \log\left(p_1^u\right)\right]. \tag{23}$$

In the training set, the real label of u is (l_1^u, l_2^u). If u is an attacker, $l_1^u = 1, l_2^u = 0$; otherwise, $l_1^u = 0, l_2^u = 1$. Stochastic gradient descent is used to optimize weight matrices.

The classification algorithm is described as follows.

In Algorithm 3, part 1 (lines 2–4) extracts user features by sparse autoencoder. Part 2 (lines 5–11) trains the GPSA model and part 3 (12–17) classifies users.

Based on the obtained *GSA*, the neighbour relations can be used to divide shilling groups.

In a shilling group, attackers need to cooperate with each other to promote the same target item. Therefore, if two attackers belong to the same shilling group, they must have a co-rated item, i.e., for each user $u, v \in GSA$, $URI_u \cap URI_v \neq \varnothing$. However, the fact that two users have a co-rated item is not enough to determine whether they

```
    Input: user-item rating matrix $\mathbf{R}$
    Output: user feature matrix $\mathbf{H}$
    Begin:
(1)     Randomly initialize weight matrices $\mathbf{W}$ and $\mathbf{W}'$
(2)     repeat
(3)         for each $u \in U$ do
(4)             $\mathbf{h}_u \longleftarrow \sigma(\mathbf{W}\mathbf{x}_u + b)$
(5)             $\widehat{\mathbf{x}}_u \longleftarrow \sigma(\mathbf{W}'\mathbf{h}_u + b')$
(6)         end for
(7)         calculate the loss according to equations (11)–(15)
(8)         perform back-propagation to update the weight matrices
(9)     until the loss converges
(10)    for each $u \in U$ do
(11)        $\mathbf{h}_u \longleftarrow \sigma(\mathbf{W}\mathbf{x}_u + b)$
(12)    end for
(13)    utilize all user vectors to construct user feature matrix $\mathbf{H}$
(14)    return $\mathbf{H}$
    End
```

Algorithm 1: Extract user features.

```
    Input: rating dataset
           sampling number $K$
    Output: sampling result $SN$
    Begin:
(1)     $SN \longleftarrow \varnothing$
(2)     for each $u \in U$ do
(3)         $N_u \longleftarrow \varnothing$
(4)         for each $v \in U$ DO
(5)             calculate $UTP_{uv}$ according to equations (1)–(8)
(6)             if $\exists i \in I, r_{ui} \neq 0 \wedge r_{vi} \neq 0$ then
(7)                 $N_u \longleftarrow N_u \cup \{v\}$
(8)             end if
(9)         end for
(10)    end for
(11)    for each $u \in U$ do
(12)        sort $N_u$ according to transition probability in descending order
(13)        while $|N_u| < K$ do
(14)            $N_u \longleftarrow N_u \| N_u$ //|| is an operation that joints two sets together.
(15)        end while
(16)        sample the first $K$ users in $N_u$ and obtain sampling neighbour set $SN_u$
(17)        $SN \longleftarrow SN \cup \{SN_u\}$
(18)    end for
(19)    return $SN$
    End
```

Algorithm 2: Transition probability-based user neighbour sampling.

belong to the same shilling group; they must have collusive behaviour on the co-rated item. The collusive behaviour between group attackers is mainly reflected in the rating value and rating time of the co-rated item. Generally, two attackers in a shilling group rate at least one item with similar rating values and close rating times, which leads to a high transition probability between them. Based on the comprehensive consideration of rating item, rating value, and rating time, the transition probabilities between users are used to divide shilling groups. Concretely, for each $u, v \in GSA$, their neighbours are sampled by Algorithm 2 and the sampling neighbour sets SN_u and SN_v are obtained.

```
Input: rating dataset
       sampling result $SN$
       training set $Tr$
       test set $Te$
       number of iterations loop
Output: the set of group shilling attackers GSA
Begin
 (1) $GSA \longleftarrow \varnothing$
 (2) for each $u \in U$ do
 (3)     obtain user feature vector $\mathbf{h}_u$ using sparse autoencoder
 (4) end for
 (5) for $k = 1$ to loop do
 (6)     for each $u \in Tr$ do
 (7)         calculate the user embeddings $\mathbf{z}_u$ according to equations (16)–(20)
 (8)         calculate the cross-entropy loss according to equations (21)–(23)
 (9)         perform back-propagation to update the weight matrices
(10)     end for
(11) end for
(12) for each $u \in Te$ do
(13)     calculate the user label probabilities $p_0^u$ and $p_1^u$ according to equations (16)–(22)
(14)     if $p_1^u > p_0^u$ then
(15)         $GSA \longleftarrow GSA \cup \{u\}$
(16)     end if
(17) end for
(18) return $GSA$
End
```

Algorithm 3: GPSA-based user classification.

If $u \in SN_v$ and $v \in SN_u$, then u and v are divided into the same shilling group. With this division method, we can obtain the final detection results.

4. Experiments

4.1. Datasets. To illustrate the effectiveness of SA-GPSA, we conduct experiments on the following datasets.

(1) Netflix [38] dataset contains 130297638 ratings, 480186 users, and 17770 movies. We randomly sample 2000 users, 3985 movies, and 215844 ratings and inject shilling groups to generate two synthetic datasets for experiments. Base on Netflix dataset, we construct two synthetic datasets.

Netflix-1: shilling groups are generated by $GSAGen_l$ Ran, $GSAGen_l$ Avg, $GSAGen_l$ AoP, and $GSAGen_l$ PIA. For each attack model, the attack size is set to {2.5%, 5%, 7.5%, 10%} and the filler size is set to {2.5%, 5%}. We use each combination of attack size and filler size to generate 2 groups and obtain 64 shilling groups including 555 attackers in total.

Netflix-2: in this dataset, 8 shilling groups are generated by *GSAGenl* GOAT. There are 432 attackers in these 8 shilling groups.

(2) Amazon [39] dataset is a review dataset including 645072 users, 136875 products, and 1205125 ratings. For evaluation, 5055 labelled users are extracted to construct a dataset, which contains 5055 users (1937 attackers and 3118 genuine users), 17610 products, and 53777 ratings.

(3) Movielens 1M dataset (http://www.grouplens.org) contains 1000209 ratings of 3947 movies from 6010 users. $GSAGen_l$ Ran, $GSAGen_l$ Avg, $GSAGen_l$ AoP, and $GSAGen_l$ PIA are used to generate shilling groups. For each attack model, the attack size is set to {3%, 6%, 9%, 12%} and the filler size is set to {2.5%, 5%}. We use each combination of attack size and filler size to generate 2 groups and obtain 64 shilling groups including 726 attackers in total.

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

$$\begin{aligned}
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}},\\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}},\\
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}},\\
F1 - \text{measure} &= \frac{2 \times \text{Precison} \times \text{Recall}}{\text{Precison} + \text{Recall}},
\end{aligned} \tag{24}$$

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

4.3. Comparison of Sampling and Classification Performance for GPSA and GraphSAGE. In this section, we compare the sampling performance and classification performance of GPSA and GraphSAGE.

Firstly, we compare the sampling performance of GraphSAGE and GPSA. We use GraphSAGE and GPSA to sample neighbours of 10 attackers selected randomly on the Netflix-1 dataset, Netflix-2 dataset, Amazon dataset, and Movielens 1M dataset, respectively. Then we separately calculate the ratios of genuine users and attackers in the sampling results. To avoid contingency, we repeat this process 20 times and use the average results as the final results, which are shown in Figures 2–5.

As can be seen from Figures 2–5, in the sampled neighbours of attackers, the attacker ratio of GPSA is much higher than that of GraphSAGE. This difference is especially obvious in the Amazon dataset; the attacker ratio is around 0.9 in the sampling result of GPSA but is only about 0.1 in sampling result of GraphSAGE. This phenomenon is explained as follows. In the Amazon dataset, most neighbours of attackers are genuine users. With a random sampling mechanism, GraphSAGE has a high probability to sample genuine neighbours of attackers. While GPSA samples neighbours according to user transition probabilities, as attackers have high transition probabilities with each other, GPSA can sample more attacker neighbours for attackers.

Secondly, we compare the classification performance of GraphSAGE and GPSA. We use GraphSAGE and GPSA to perform user classification on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets, respectively. Figure 6 shows the classification performance of GraphSAGE and GPSA on four datasets.

As shown in Figure 6, the overall classification performance of GPSA is better than that of GraphSAGE; the F1-measure of GPSA is 3.48%, 10.16%, 6.65%, and 2.29% higher than that of GraphSAGE on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets separately. These improvements are attributable to the superior sampling performance of GPSA. With the better sampling results, GPSA can reduce the influence of genuine user features on the attackers in feature aggregation. Thus, GPSA can learn better user embeddings and classify users more accurately.

4.4. Comparison of Detection Performance with Other Methods. To show the superiority of SA-GPSA on detection performance, we compare it with the following six detection methods:

(1) Catch the black sheep (CBS) [25] is a detection method which ranks users according to user suspicious degrees. CBS selects some attackers as seed users and performs suspicious degree propagations on the bipartite graph to calculate user suspicious degrees. In the experiments. We randomly select 0.5% attackers as seed users.

(2) GD-BKM [32] detects shilling groups based on time burst. This method divides candidate groups based on time interval. Then several features are extracted to measure group suspicious degrees and the bisecting K-means is used to detect shilling groups.

(3) CoDetector [40] 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, number of negative samples, number of iterations, and the learning rate are set to 10, 25, 200, and 0.01, respectively.

(4) DCEDM [41] uses the stacked denoising autoencoders to extract the robust graph features. Then the shilling attacks are detected based on these features. In the experiments, parameters ρ and β are set to 0.05 and 0.9, respectively, and the noise level α is set to 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1, respectively.

(5) TP-GBF [34] constructs weighted user relation graph and uses a topological potential method to find suspicious groups. Then the k-means is used to obtain shilling groups. In the experiments, parameter α is set to 1 on the Netflix and Movielens 1M datasets and 0.47 on the Amazon dataset.

(6) SA-GraphSAGE is a shilling group detection method based on GraphSAGE [33]. The detection framework is the same to SA-GPSA, but it performs user classification through the standard GraphSAGE model. In the experiments, the parameters of SA-GraphSAGE are set to the same values with those of SA-GPSA.

Table 3 shows the precision, recall, and F1-measure of SA-GPSA, CBS, DECEM, TP-GBF, GD-BKM, CoDetector, and SA-GraphSAGE on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets.

As shown in Table 3, SA-GPSA has better precision, recall, and F1-measure than CBS, DECEM, TP-GBF, GD-BKM, CoDetector, and SA-GraphSAGE whether on the synthetic datasets or on the real dataset. For example, the F1-measure of SA-GPSA on the Netflix-1 dataset is 0.9991, which is 12.78%, 28.32%, 11.61%, 6.25%, 61.37%, and 3.6% higher than that of CBS, DECDM, TP-GBF, GD-BKM, CoDetector, and SA-GraphSAGE. For another example, the F1-measure of SA-GPSA on the Movielens 1M dataset is 0.9924, which has an improvement of 13.01%, 25.87%, 8.97%, 61.87%, 79.17%, and 2.53% over CBS, DECDM, TP-GBF, GD-BKM, CoDetector, and SA-GraphSAGE. The high performance of SA-GPSA compared with other methods is mainly due to two reasons. On the one hand, the GPSA model introduces the probability sampling mechanism. In this sampling mechanism, the transition probabilities between users are calculated according to user collusive degrees. With the transition probability, the GPSA model is more biased in sampling neighbours. Therefore, the GPSA model can sample as many neighbours as possible with the

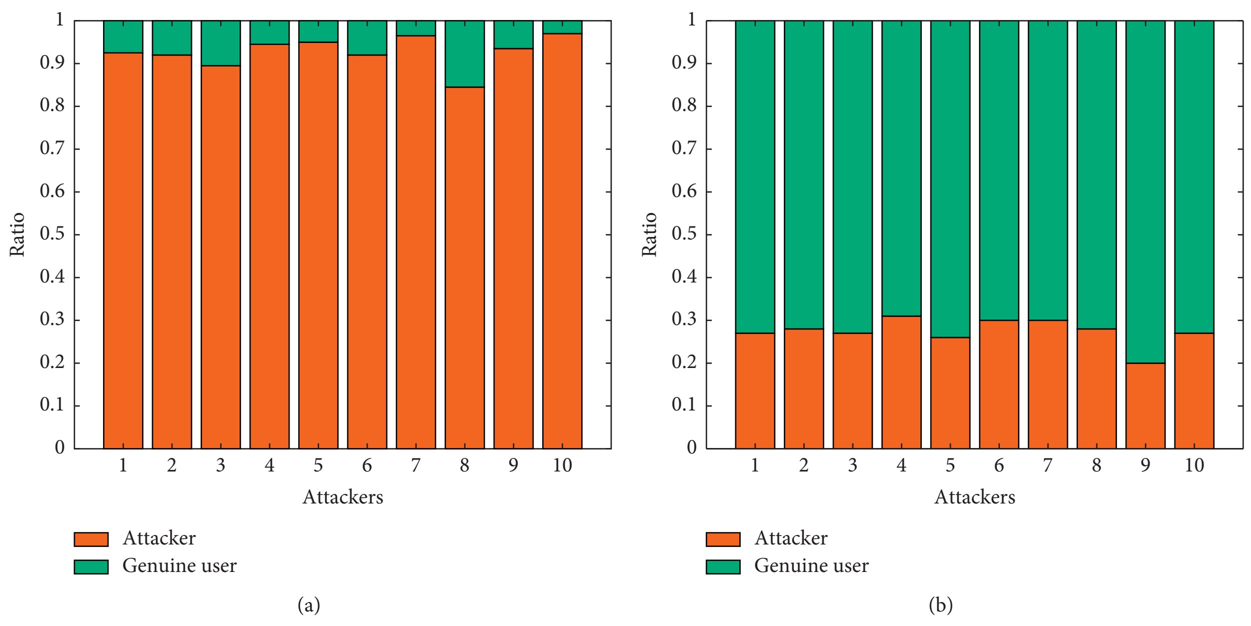


FIGURE 2: Comparison of sampling results for GPSA and GraphSAGE on Netflix-1 dataset. (a) Sampling result of GPSA. (b) Sampling result of GraphSAGE.

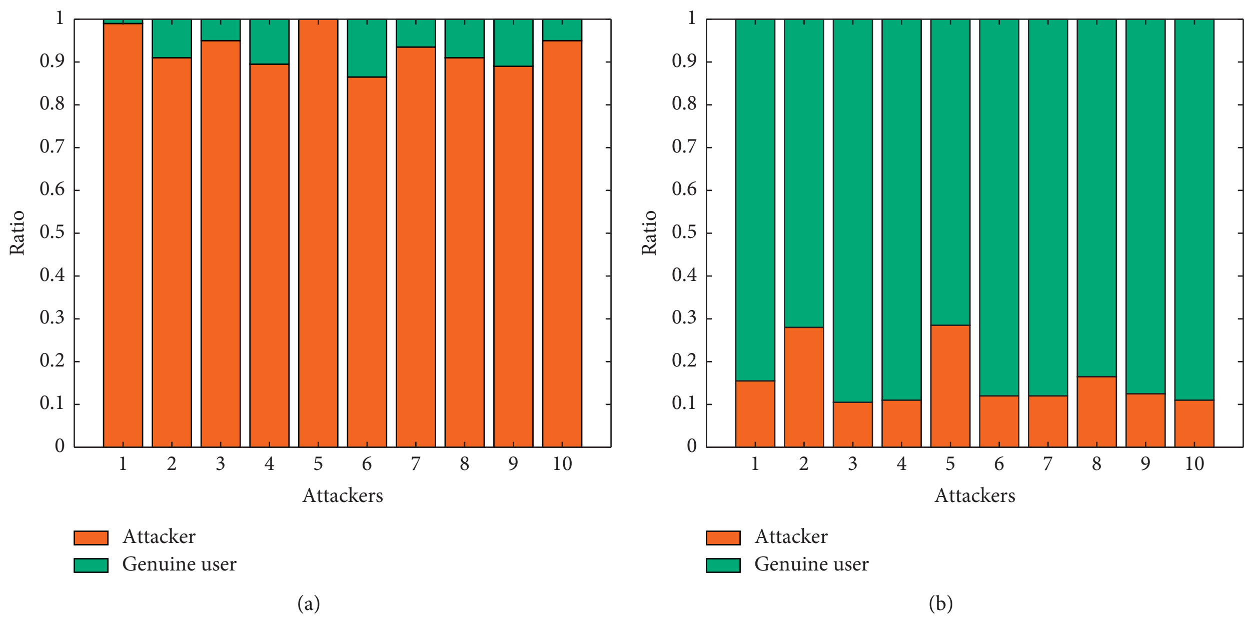


FIGURE 3: Comparison of sampling results for GPSA and GraphSAGE on Netflix-2 dataset. (a) Sampling result of GPSA. (b) Sampling result of GraphSAGE.

same labels as the central user, which makes it generate good and stable classification results for gathering the shilling groups. On the other hand, the integration of sparse autoencoder and GPSA realizes automatic extraction of user features, which can adjust user features according to rating datasets and learn the most suitable user features. As considering both the structural and behavioural features of users while designing the model, SA-GPSA can detect shilling groups effectively.

As a classic detection method of shilling attacks, CBS performs differently on the two Netflix synthetic datasets. The F1-measure of CBS is 0.8713 on the Netflix-1 dataset but is only 0.5867 on the Netflix-2 dataset. This difference is mainly caused by the basic attack models used in the group attack model. Traditional shilling attack models (random attack, average attack, AoP attack, and PIA) choose filler items from popular items while novel attack model (GOAT) chooses filler items form template user profiles. Therefore,

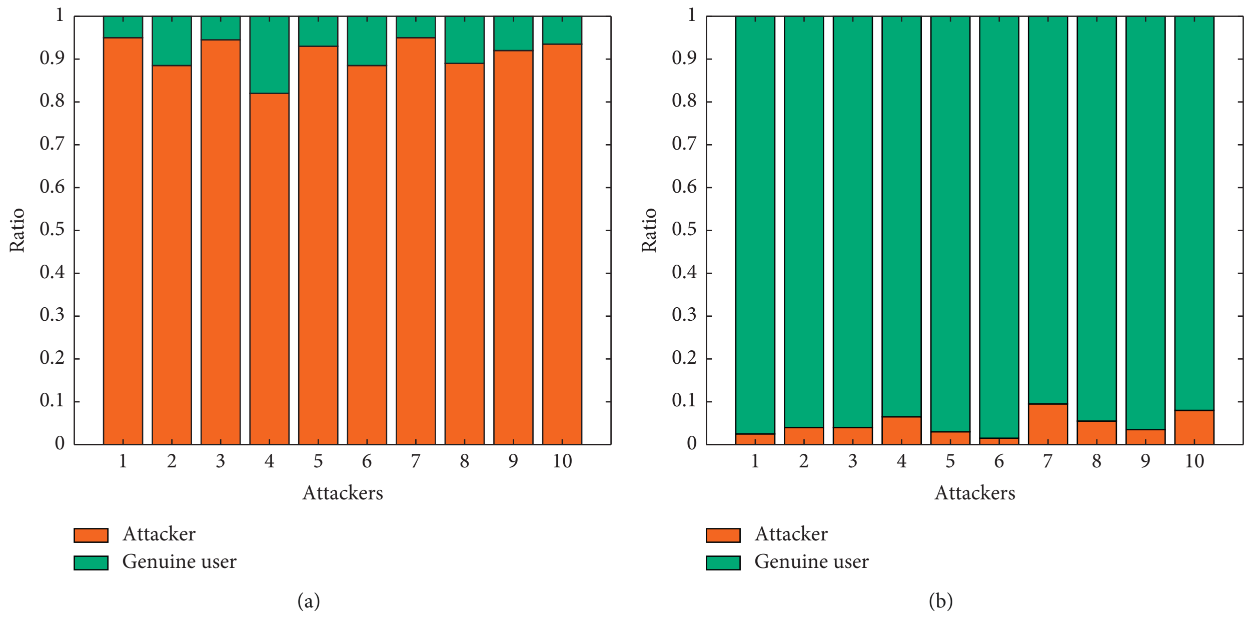


Figure 4: Comparison of sampling results for GPSA and GraphSAGE on Amazon dataset. (a) Sampling result of GPSA. (b) Sampling result of GraphSAGE.

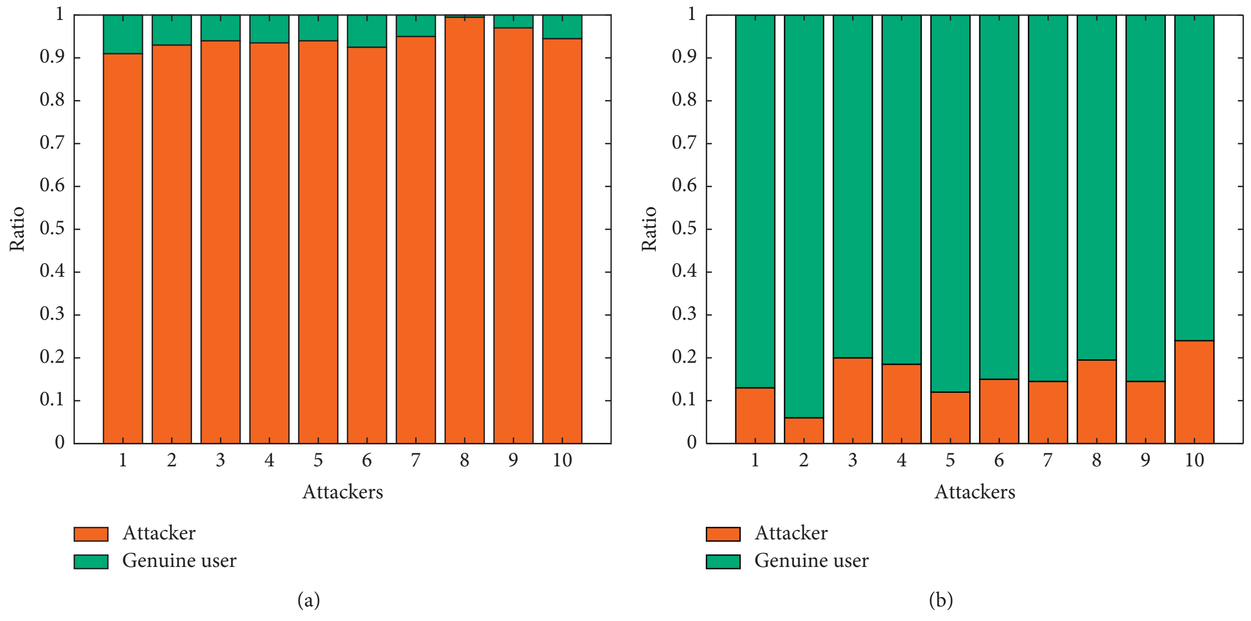


Figure 5: Comparison of sampling results for GPSA and GraphSAGE on Movielens 1M dataset. (a) Sampling result of GPSA. (b) Sampling result of GraphSAGE.

the size of fake profiles in the Netflix-2 dataset is much smaller than that in the Netflix-1 dataset. With fewer rating items, the transition probability between attackers may be lower and the final suspicious degrees of attackers may be smaller. In the Amazon dataset, rating behaviours of attackers are more diverse, so the F1-measure of CBS is about 0.85, which is slightly lower than that on the Netflix-1 dataset. On the Movielens 1M dataset, the detection performance of CBS is close to that on the Netflix-1 dataset.

GD-BKM performs well on the Netflix synthetic datasets; the F1-measure values are 0.93 and 0.80. In the Netflix-2 dataset, the rating behaviours of attackers are very similar to those of genuine users and some features proposed in GD-BKM are not very effective for distinguishing genuine users from attackers, so the detection performance of GD-BKM on the Netflix-2 dataset is worse than that on the Netflix-1 dataset. On the Amazon dataset, the F1-measure is 0.74, which is lower than that on the Netflix-1 and Netflix-2 datasets. The core idea of GD-BKM is that users may be attackers if users and items rated by them are both active in the same time period. In the Netflix synthetic datasets, attackers in the same group rate target items in a short time.

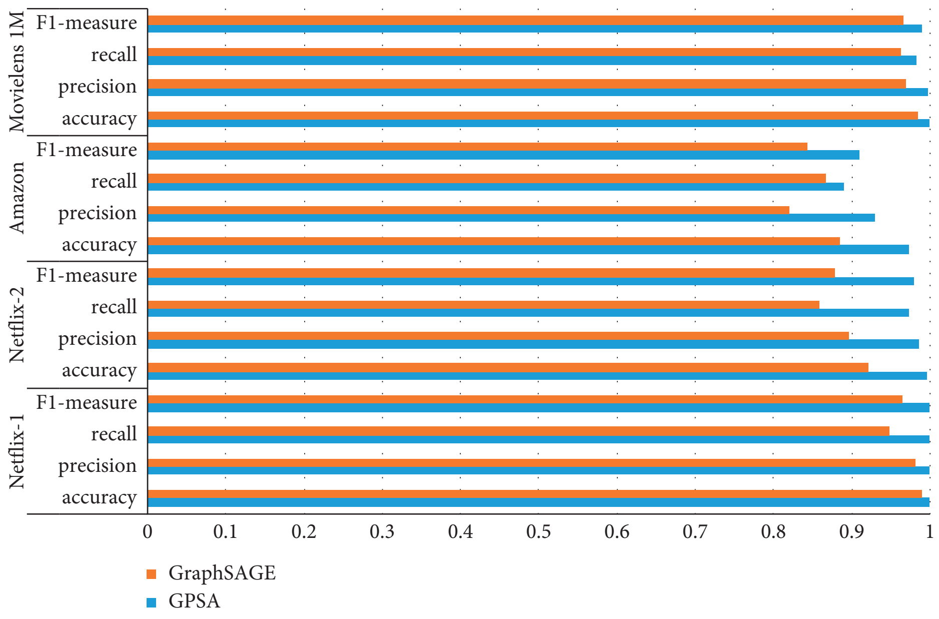


Figure 6: Comparison of classification performance for GraphSAGE and GPSA.

Table 3: Comparison of detection performance for seven methods

	Netflix-1			Netflix-2			Amazon			Movielens 1M		
	Precision	Recall	F1-measure	Precision	Recall	F1-measure	Precision	Recall	F1-measure	Precision	Recall	F1-measure
CBS	0.8973	0.8468	0.8713	0.6024	0.5718	0.5867	0.8793	0.8353	0.8567	0.8862	0.8396	0.8623
DCEDM	0.9838	0.5627	0.7159	0.7053	0.4599	0.5568	0.8637	0.7221	0.7866	0.9785	0.5869	0.7337
TP-GBF	0.9968	0.7926	0.8830	0.0124	0.0070	0.0089	0.9283	0.6467	0.7623	0.9856	0.8327	0.9027
GD-BKM	0.9044	0.9712	0.9366	0.7963	0.8052	0.8007	0.8234	0.6732	0.7408	0.2469	0.7683	0.3737
CoDetector	0.4328	0.3473	0.3854	0.4624	0.3094	0.3707	0.8056	0.8345	0.8198	0.1972	0.2043	0.2007
SA-GraphSAGE	0.9823	0.9446	0.9631	0.9023	0.8527	0.8768	0.8327	0.8652	0.8486	0.9756	0.9587	0.9671
SA-GPSA	**0.9991**	**0.9992**	**0.9991**	**0.9861**	**0.9726**	**0.9793**	**0.9299**	**0.8907**	**0.9099**	**0.9983**	**0.9865**	**0.9924**

However, the rating time of attackers is relatively scattered in the Amazon dataset. Therefore, GD-BKM has better detection performance on the Netflix synthetic datasets than that on the real dataset. As can be seen in Table 3, GD-BKM performs poorly on the Movielens 1M dataset; its F1-measure is only 0.3737. The reason is that the rating times of genuine users in the Movielens 1M dataset are very close, so it is difficult for GD-BKM to distinguish genuine users from attackers based on rating time.

In contrast to GD-BKM, the detection performance of CoDetector on the real dataset is better than that on the synthetic datasets. The F1-measure of CoDetector on the Amazon dataset is 0.8198 while it is only 0.3854, 0.3707, and 0.2007 on the Netflix-1, Netflix-2, and Movielens 1M datasets. This phenomenon is mainly due to the different distribution of datasets. The ratio of attackers is about 0.2 in the Netflix-1 and Netflix-2 datasets, 0.1 in the Movielens 1M dataset, and 0.4 in the Amazon dataset. As a supervised method, CoDetector can obtain more attackers to train the model on the Amazon dataset, so the detection performance of CoDetector is better on the Amazon dataset.

The precision of DCEDM is 0.9838 and the recall of DCEDM is only 0.5627 on the Netflix-1 dataset. Similar results are obtained on the Movielens 1M dataset. The possible reason is that DCEDM uses the base clustering results produced with the k-means clustering algorithm to reconstruct user-user graph and might divide small shilling groups and genuine users together. In the Netflix-1 and Movielens 1M datasets, there are 64 shilling groups that include 32 small shilling groups. In each small shilling group, there are about 5 attackers. Therefore, attackers in these small shilling groups are easily divided with genuine users together and identified as genuine users. On the Netflix-2 dataset, the precision, recall, and F1-measure of DCEDM are 0.7053, 0.4599, and 0.5568. In Neflix-2 dataset, the attack profiles are generated by imitating genuine user profiles, so attackers are similar to genuine users. Therefore, not only are attackers misjudged as genuine users, but also genuine users are misjudged as attackers. As a result, the precision and recall of DCEDM on the Netflix-2 dataset are lower than those on the Netflix-1 and Movielens 1M datasets. On the Amazon dataset, the F1-measure of

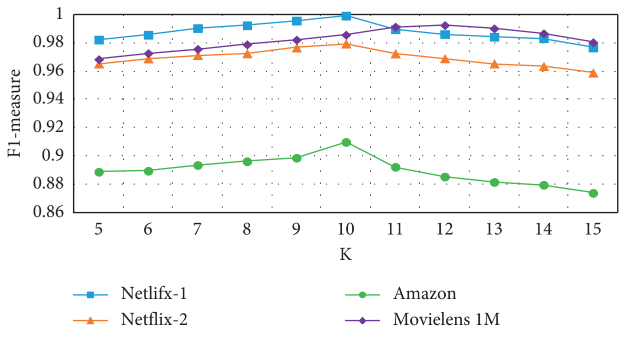


Figure 7: The impact of K on the F1-measure of SA-GPSA on different datasets.

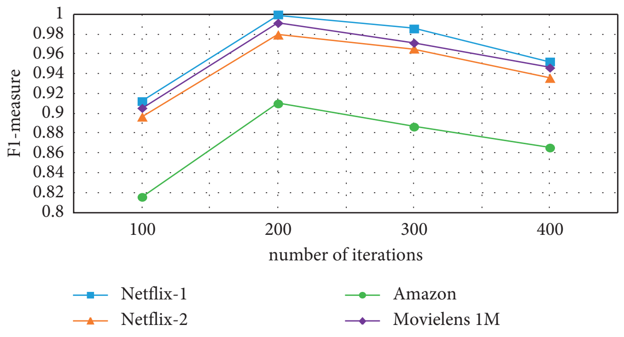


Figure 8: The impact of number of iterations on the F1-measure of SA-GPSA on different datasets.

DCEDM is better than that on the Netflix and Movielens 1M datasets, because a relatively small number of attackers are misclassified.

Similar to DCEDM, the performance of TP-GBF on the Netflix-1, Amazon, and Movielens 1M datasets 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 three datasets. On the Netflix-2 dataset, TP-GBF has the worst performance, because the topology potential is related to the number of neighbours and attackers have the least neighbours in the Netflix-2 dataset. Most attackers are filtered out when selecting suspicious groups.

As can be seen in Table 3, SA-GraphSAGE has the lower detection performance than SA-GPSA while it has the same parameter settings as SA-GPSA. For example, the F1-measure of SA-GraphSAGE is about 3.6%, 10.25%, 6.13%, and 2.53% lower than that of SA-GPSA on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets. The difference in detection performance between two methods is mainly attributed to the difference in their sampling strategies. The GraphSAGE model samples neighbours randomly, which may sample some genuine users as neighbours of the attackers during feature aggregation. As a result, SA-GraphSAGE may misjudge some users in performing user classification, which causes a decline in detection performance.

Based on the analyses above, it can be concluded that SA-GPSA has better performance than CBS, DECEM, TP-GBF, GD-BKM, and CoDetector on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets.

4.5. Analysis of Parameters. The parameters of the proposed SA-GPSA mainly include the sampling number K used in Algorithm 2, number of iterations (i.e., *loop*) used in Algorithm 3, and the learning rate. These parameters can be set by experiment. In order to select proper values of parameters, we conduct an experiment on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets. Figures 7–9 separately show the impact of parameters K, loop, and the learning rate on the F1-measure of SA-GPSA on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets.

As can be seen in Figure 7, on the Netflix-1, Netflix-2, and Amazon datasets, the F1-measure of SA-GPSA shows a steady upward trend with the increase in K and has the optimal value when K is equal to 10. Subsequently, the F1-measure of SA-GPSA on these three datasets tends to decrease as K value increases. Therefore, we set K to 10 on the Netflix-1, Netflix-2, and Amazon datasets, respectively. Similarly, we set K to 12 on the Movielens 1M dataset. As the size of shilling groups in the Movielens 1M dataset is larger than that in other three datasets, we can sample more neighbours on this dataset.

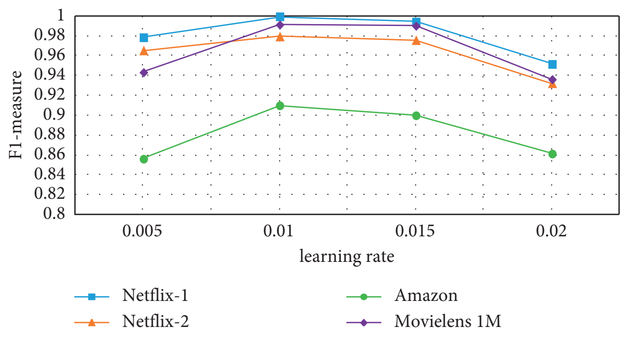


Figure 9: The impact of learning rate on the F1-measure of SA-GPSA on different datasets.

As shown in Figure 8, the number of iterations has a great influence on the F1-measure of SA-GPSA on four datasets. The F1-measure of SA-GPSA shows an upward trend as the number of iterations increases and reaches the best when the number of iterations is equal to 200. Subsequently, the F1-measure of SA-GPSA tends to decrease with the increase of the number of iterations. Therefore, we set number of iterations to 200 on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets, respectively.

It can be seen from Figure 9, on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets, the F1-measure of SA-GPSA increases gradually as the learning rate increases. When the learning rate is equal to 0.01, the F1-measure of SA-GPSA on four datasets reaches the optimal value. After that, with the increase of the learning rate, the F1-measure of SA-GPSA on four datasets tends to decrease. Therefore, we set the learning rate to 0.01 on the Netflix-1, Netflix-2, Amazon, and Movielens 1M datasets, respectively.

5. Conclusion

To detect group shilling attacks in online recommender systems, we propose a detection framework based on deep learning technology which is named SA-GPSA. In SA-GPSA, we have replaced the random sampling mechanism of the standard GraphSAGE model with a probability sampling mechanism to construct the GPSA model. Experiment results have indicated that the GPSA model has better classification performance than the GraphSAGE model. In this method, we have also proposed a shilling group detection framework by integrating sparse autoencoder and GPSA. The proposed framework can extract user features automatically without manual intervention. Extensive experiments have demonstrated that the proposed framework can detect various types of shilling groups and perform better than existing detection methods.

While SA-GPSA has a high performance in detecting shilling groups, it also has limitations to be further addressed. For example, the detection performance of SA-GPSA is not stable when the ratio of attackers in the dataset is too low. As a supervised detection method, SA-GPSA needs training with sample data. If there are not enough sample data of attackers for training, SA-GPSA cannot effectively learn the representations for user classification, especially when the ratio of attackers is lower than 5% in the dataset. Hence, how to solve the unbalanced classification problem in the proposed method is our future work. Another example, when the dataset (e.g., YelpChi [42]) is extremely sparse and there are only a few co-rated items among users, the detection performance of SA-GPSA is poor. Since it is difficult to find useful structure information in the user relation graph constructed from the extremely sparse dataset, the SA-GPSA model cannot be trained sufficiently. Therefore, how to effectively detect shilling groups in the extremely sparse dataset is another future work.

Data Availability

The data used to support the findings of this study are available from the first author (shileiwang546@163.com) upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

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

References

[1] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.
[2] B. Smith and G. Linden, "Two decades of recommender systems at amazon.com," *IEEE Internet Computing*, vol. 21, no. 3, pp. 12–18, 2017.
[3] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: a review," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 291–319, 2020.
[4] X. F. Su, H. J. Zeng, and Z. Chen, "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, New York, NY, USA, May 2005.
[5] Y. Wang, Z. Wu, J. Cao, and C. Fang, "Towards a tricksy group shilling attack model against recommender systems," in *Proceedings of the Towards a Tricksy Group Shilling Attack Model against Recommender Systems*, pp. 675–688Advanced Data Mining and Applications, Nanjing, China, December 2012.
[6] F. Wu, M. Gao, J. Yu, Z. Wang, K. Liu, and X. Wang, "Ready for emerging threats to recommender systems? A graph convolution-based generative shilling attack," *Information Sciences*, vol. 578, pp. 683–701, 2021.
[7] C. Panagiotakis, H. Papadakis, and P. Fragopoulou, "Unsupervised and supervised methods for the detection of hurriedly created profiles in recommender systems," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 9, pp. 2165–2179, 2020.
[8] P. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the*

7th International Workshop on Web Information and Data Management, pp. 67–74, Bremen, Germany, November 2005.

[9] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommendation systems," in *Proceedings of the 12th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pp. 542–547, Philadelphia, PA, USA, August 2006.

[10] W. Li, M. Gao, H. Li, Q. Xiong, J. Wen, and B. Ling, "An shilling attack detection algorithm based on popularity degree features," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 41, no. 9, pp. 1563–1575, 2015, in Chinese.

[11] X. Liu, Y. Xiao, Y. Xiao, X. Jiao, W. Zheng, and Z. Ling, "A novel kalman filter based shilling attack detection algorithm," *Mathematical Biosciences and Engineering*, vol. 17, no. 2, pp. 1558–1577, 2020.

[12] C. Tong, X. Yin, J. Li et al., "A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network," *The Computer Journal*, vol. 61, no. 7, pp. 949–958, 2018.

[13] H. Li, M. Gao, F. Zhou, Y. Wang, Q. Fan, and L. Yang, "Fusing hypergraph spectral features for shilling attack detection," *Journal of Information Security and Applications*, vol. 63, Article ID 103051, 2021.

[14] K. Vivekanandan and N. Praveena, "Hybrid convolutional neural network (CNN) and long-short term memory (LSTM) based deep learning model for detecting shilling attack in the social-aware network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 1197–1210, 2021.

[15] H. Cai and F. Zhang, "Detecting shilling attacks in recommender systems based on analysis of user rating behavior," *Knowledge-Based Systems*, vol. 177, pp. 22–43, 2019.

[16] H. Xia, B. Fang, M. Gao, H. Ma, Y. Tang, and J. Wen, "A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique," *Information Sciences*, vol. 306, pp. 150–165, 2015.

[17] Z. Yang and Z. Cai, "Detecting abnormal profiles in collaborative filtering recommender systems," *Journal of Intelligent Information Systems*, vol. 48, no. 3, pp. 499–518, 2017.

[18] R. Bhaumik, B. Mobasher, and R. Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in *Proceedings of the 7th IEEE International Conference on Data Mining*, pp. 181–187, Vancouver, Canada, December 2011.

[19] Z. Zhang and S. R. Kulkarni, "Detection of shilling attacks in recommender systems via spectral clustering," in *Proceedings of the 17th International Conference on Information Fusion*, pp. 1–8, Salamanca, Spain, July 2014.

[20] J.-S. Lee and D. Zhu, "Shilling attack detection-A new approach for a trustworthy recommender system," *INFORMS Journal on Computing*, vol. 24, no. 1, pp. 117–131, 2012.

[21] H. Cai and F. Zhang, "BS-SC: an unsupervised approach for detecting shilling profiles in collaborative recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1375–1388, 2021.

[22] Z. Zhang and S. R. Kulkarni, "Graph-based detection of shilling attacks in recommender systems," in *Proceedings of the 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Southampton, United kingdom, September 2013.

[23] Z. Batmaz, B. Yilmazel, and C. Kaleli, "Shilling attack detection in binary data: a classification approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 6, pp. 2601–2611, 2020.

[24] Z. Yang, Z. Cai, and X. Guan, "Estimating user behavior toward detecting anomalous ratings in rating systems," *Knowledge-Based Systems*, vol. 111, pp. 144–158, 2016.

[25] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, C. Tat-Seng, and S. Ma, "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, Buenos Aires, Argentina, July 2015.

[26] Z. Wu, J. Cao, B. Mao, and Y. Wang, "Semi-SAD: applying semi-supervised learning to shilling attack detection," in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 289–292, Chicago, IL, United states, October 2011.

[27] J. Cao, Z. Wu, B. Mao, and Y. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," *World Wide Web*, vol. 16, no. 5-6, pp. 729–748, 2013.

[28] Z. Wu, J. Wu, J. Cao, and D. Tao, "HySAD: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 985–993, Beijing, China, August 2012.

[29] Q. Zhou and L. Duan, "Semi-supervised recommendation attack detection based on Co-Forest," *Computers & Security*, vol. 109, Article ID 102390, 2021.

[30] W. Zhou, Y. S. Koh, J. Wen, S. Burki, and G. Dobbie, "Detection of abnormal profiles on group attacks in recommender systems," in *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 955–958, Gold Coast, QLD, Australia, July 2014.

[31] Y. Wang, Z. Wu, Z. Bu, J. Cao, and D. Yang, "Discovering shilling groups in a real e-commerce platform," *Online Information Review*, vol. 40, no. 1, pp. 62–78, 2016.

[32] F. Zhang and S. Wang, "Detecting group shilling attacks in online recommender systems based on bisecting k-means clustering," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 5, pp. 1189–1199, 2020.

[33] F. Zhang, Y. Qu, Y. Xu, and S. Wang, "Graph embedding-based approach for detecting group shilling attacks in collaborative recommender systems," *Knowledge-Based Systems*, vol. 199, Article ID 105984, 2020.

[34] H. Cai and F. Zhang, "An unsupervised approach for detecting group shilling attacks in recommender systems based on topological potential and group behaviour features," *Security and Communication Networks*, vol. 202118 pages, Article ID 2907691, 2021.

[35] H. Yu, H. Zheng, Y. Xu, R. Ma, D. Gao, and F. Zhang, "Detecting group shilling attacks in recommender systems based on maximum dense subtensor mining," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 644–648, Dalian, China, June 2021.

[36] S. Wang, H. Wang, H. Yu, and F. Zhang, "Detecting shilling groups in recommender systems based on hierarchical topic model," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Computer Applications*, pp. 832–837, Dalian, China, June 2021.

[37] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st Annual Conference on Neural Information Processing*

Systems, pp. 1025–1035, Long Beach, CA, United states, December 2017.

[38] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[39] C. Xu, J. Zhang, K. Chang, and C. Long, "Uncovering collusive spammers in Chinese review websites," in *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management*, pp. 979–988, San Francisco, CA, United states, November 2013.

[40] T. Dou, J. Yu, Q. Xiong, M. Gao, Y. Song, and Q. Fang, "Collaborative shilling detection bridging factorization and user embedding," in *Proceedings of the Collaborative Shilling Detection Bridging Factorization and User Embedding*, pp. 459–469Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Edinburgh, United kingdom, December 2018.

[41] Y. Hao and F. Zhang, "An unsupervised detection method for shilling attacks based on deep learning and community detection," *Soft Computing*, vol. 25, no. 1, pp. 477–494, 2021.

[42] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?" in *Proceedings of the 7th International Conference on Weblogs and Social Media*, pp. 409–418, Cambridge, MA, United states, July 2013.