Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 2854874, 15 pages
https://doi.org/10.1155/2023/2854874

Research Article

KC-GCN: A Semi-Supervised Detection Model against Various Group Shilling Attacks in Recommender Systems

Hongyun Cai,[1,2] **Jichao Ren**,[1,2] **Jing Zhao**,[3] **Shilin Yuan**,[1,2] **and Jie Meng**[1,2]

[1]*School of Cyber Security and Computer, Hebei University, Baoding 071000, China*
[2]*Key Laboratory on High Trusted Information System in Hebei Province, Hebei University, Baoding 071000, China*
[3]*Security Department, Hebei University, Baoding 071000, China*

Correspondence should be addressed to Jichao Ren; renjich@163.com

Received 7 July 2022; Revised 7 November 2022; Accepted 1 February 2023; Published 16 February 2023

Academic Editor: Yawen Chen

Various detection methods have been proposed for defense against group shilling attacks in recommender systems; however, these methods cannot effectively detect attack groups generated based on adversarial attacks (e.g., GOAT) or mixed attack groups. In this study, we propose a two-stage method, called KC-GCN, which is based on k-cliques and graph convolutional networks. First, we construct a user relationship graph, generate suspicious candidate groups, and extract influential users by calculating the user nearest-neighbor similarity. We construct the user relationship graph by calculating the edge weight between any two users through analyzing their similarity over suspicious time intervals on each item. Second, we combine the extracted user initial embeddings and the structural features hidden in the user relationship graph to detect attackers. On the Netflix and sampled Amazon datasets, the detection results of KC-GCN surpass those of the state-of-the-art methods under different types of group shilling attacks. The F1-measure of KC-GCN can reach above 93% and 87% on these two datasets, respectively.

1. Introduction

The amount of Internet data is exploding with the rapid development of information technology, consequently leading to the increasingly prominent problem of information overload. By analyzing a user's historical behavior information, recommender systems can extract user preferences and automatically recommend favorite items or services to users [1–3], which have become an essential component of many online information services, including e-commerce [4, 5], live broadcast platforms [6], personalized travel recommendation systems [7], and Internet of Vehicles wireless systems [8], among many others. However, due to their openness, fraudulent users can create and inject a large number of fake user profiles into recommender systems, which can change recommendation results and reduce user experience. For example, The New York Times and Buzzfeed News have reported that many sellers turned to black hat tactics to drive Amazon sales on their products (https://pattern.com/news/pattern-analysis-on-amazon-star-rating-featured-in-new-york-times-buzzfeed/). In recent years, various types of shilling attack models have been presented, including random attacks [9], average attacks [10], and the latest adversarial attacks [11]. Group shilling attacks have also been proposed to generate a group of attack profiles on the basis of the abovementioned individual shilling attacks [12]. Research on group shilling attacks showed that group attacks greatly affect recommender systems when compared to traditional individual attacks [13, 14] because attack users in the same shilling group collude with each other to attack targets, while each attack profile looks more like a genuine profile [15]. Nowadays, people have become increasingly conscious of the importance of shilling attack governance in recommender systems. Many service platforms, such as Amazon, Tripadvisor, and Taobao, are constantly seeking efficient mechanisms to enhance user experience and satisfaction (https://www.bbc.com/news/business-61154748). Therefore, accurate detection under group shilling attacks has emerged as a crucial problem for the existing recommender system security.

In recent years, various detection approaches have been put forward to defend recommender systems from group

shilling attacks [16–21]. Most of these methods detect shilling groups based on frequent synchronization behaviors on more than one item or through the analysis of the differences in the rating pattern of genuine and attack users. However, these approaches do not perform well if the group attack profiles are generated and injected based on AOP [9], adversarial attacks, or mixed attacks because all attackers in the same shilling group may not attack identical target items. The injected attack profiles are also diverse and look more like genuine ones under these attacks.

To solve the abovementioned constraints, we present herein a two-stage method, called KC-GCN. It is a semi-supervised group shilling attack detection model based on k-cliques and the graph convolutional network (GCN) [22, 23]. First, a user relationship graph is generated, and influential users are extracted using the k-clique algorithm and the user nearest-neighbor similarity on the graph. We construct the user relationship graph by calculating the edge weight between any two users through the analysis of their similarity over suspicious time intervals on each item. Second, we obtain the user initial embeddings and train a GCN-based classifier.

The significant contributions of this work are as follows:

(1) We construct a weighted user relationship graph, in which the weight is calculated from the perspectives of user preference, attack intention, and time synchronization, to highlight the user relationship between attack users

(2) We use the multilayer graph convolution network to fuse the initial embedded features extracted from the user rating behavior with the structural features hidden in the user relationship graph to extract more effective detection features

(3) The experiments on the Netflix and Amazon datasets demonstrate that KC-GCN outperforms baseline methods in terms of detecting various types of group shilling attacks

The rest of this paper is structured as follows: Section 2 presents the background information and related work, Section 3 provides a detailed description of the proposed detection methodology primarily divided into two sections (i.e., extracting influential users and identifying attack users using the trained semi-supervised classifier), Section 4 provides a comparative analysis of the experimental findings, and Section 5 presents the conclusions.

2. Background and Related Work

2.1. Group Shilling Attacks. To escape from the existing methods of detecting individual shilling attacks (e.g., random attack, average attack, and AOP attack), Wang et al. [24] proposed two generative models of group shilling attack, called GSAGen_s and GSAGen_l. In these attack models, fake profiles are first generated based on one type of individual shilling attacks. Based on which, group shilling attack profiles are then constructed and injected into a set of genuine profiles. The GSAGen_s model has more stringent conditions when generating group shilling profiles; hence, the group size under GSAGen_s is smaller than that under GSAGen_l. Considering the attack effect on the target items, we only use GSAGen_l to generate the group shilling attack profiles, in which the fake profile includes the selected item set, the filler item set, the target item set, and the unrated item set. More details for the group shilling attacks used in this paper are described as follows:

(1) *$GSAGen_l$ Ran*: generate loose group attack profiles based on a random attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates the items. The filler item rating is the system mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(2) *$GSAGen_l$ Ave*: generate loose group attack profiles on the basis of an average attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates the items. The filler item rating is the item mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(3) *$GSAGen_l$ AOP*: generate loose group attack profiles based on 50% AOP attack, where the selected items are null, the filler items are randomly chosen, and only one attacker from the whole group rates those items with top 50% popularity. The filler item rating is the item mean. The target item rating is set to $r_{\max}$ or $r_{\min}$

(4) *$GSAGen_l$ GOAT*: generate loose group attack profiles based on the adversarial attack, called GOAT [11], where each fake user's selected and filler items are randomly chosen from an item-item graph based on genuine user profiles. A generative adversarial network is used to generate the ratings of the selected and filler items based on the genuine rating distribution. The target items have ratings of $r_{\max} - 1$ or $r_{\min} + 1$

(5) *$GSAGen_l$ Mixed*: generate mixed multiple shilling groups generated according to the four abovementioned group shilling attacks

2.2. Related Work

2.2.1. Individual Shilling Attack Detection Methods. Chirita et al. [25] and Burke et al. [26] proposed attack user detection indicators to identify shilling attackers based on fraudulent user rating behavior patterns. These indicators were suitable for detecting specific attacks (i.e., random attack), but failed under obfuscated attacks (e.g., AOP attack). To detect various attack types, Zhang et al. [27] proposed an attack detection framework based on label propagation. The framework used the label propagation algorithm to obtain the suspicious probability of each user. Although this method did not require the consideration of particular attack strategies, it needs a certain number of seed users and must know the number of attackers in advance. Zhang et al. [28]

put forward an unsupervised attack detection method, called UD-HMM, which first determined each user's degree of suspicion based on their hidden Markov model behavior before utilizing hierarchical clustering to identify attackers. However, this method did not work for detecting the AOP attack profiles. Yang et al. [29] proposed a unified detection framework that can detect various malicious attacks, including common access injection and shilling attacks. Their framework transformed the user rating behavior into a coupled association network. The network connections and nodes were assessed for trustworthiness by utilizing coupling factor graphs and label propagation algorithms. Meanwhile, Hao and Zhang [30] proposed a deep learning-based and community detection unsupervised approach, called DECDM. They constructed a graph of weighted user relationships based on user behavior similarity and then reconstructed the user relationship graph using stacked denoising autoencoders (SDAEs) and the k-means algorithm. The experiments showed that the method has excellent detection performance on multiple individual shilling attacks. However, it uses the SDAEs multiple times to extract graph features with different damage rates, resulting in a high algorithm time complexity. Ebrahimian and Kashef [31] proposed a hybrid shilling attack detection model based on the convolutional and recurrent neural networks, which first converted the rating matrix into a three-dimensional array of users, products, and days; extracted the user feature vector by using the convolutional neural network (CNN) model; and finally used the RNN model to divide users into two categories: genuine and attacker users. This model did not rely on specific types of attacks and considered the user characteristics in the time dimension. However, the experimental results on the two datasets of Netflix and MovieLens showed that the detection performance was extremely unstable as the filler size changed. Zhou and Duan [32] proposed a coforest algorithm-based semi-supervised recommendation attack detection method that requires setting a reasonable value for each hyperparameter. Zhang et al. [33] proposed GraphRfi, which trains the GCN to obtain the prediction error and introduces neural random forests to detect fraudulent users. Similar to that in [32], the method also requires multiple hyperparameters, and the detection result is easily affected by the hyperparameters.

2.2.2. Group Attack Detection Methods. Zhou et al. [16] proposed the DeR-TIA to identify group attack profiles. In the first stage, they calculated the user profile attributes using improved RDMA and DegSim. In the second stage, they filtered out attack profiles by using the target item analysis. This method works well for identifying high-correlation attack profiles but fails to detect attack groups with a strong diversity. Zhou et al. [17] proposed a detection method, called SVM-TIA, based on the support vector machine and target item analysis. This method can improve the detection precision by using target items but does not have a high recall. Zhang and Wang [18] proposed the GD-BKM method to detect group shilling attacks. They generated candidate groups based on the rating tracks for each item and calculated the candidate group suspiciousness using the user activity and group item attention degree. They then finally spotted attack groups by using the bisecting k-means algorithm. This method can exhibit an excellent detection performance, regardless of the number of target items. However, it becomes less effective when the size of the shilling group is small. Zhang et al. [19] proposed the GAGE method based on graph embedding. First, they extracted user embeddings using the Node2vec method. Next, they obtained candidate groups by using the k-means++ algorithm and calculated the group suspicious degrees. Ultimately, they identified attack groups using Ward's hierarchical-clustering algorithm. Their method uses Node2vec sampling with a certain randomness, thereby leading to deviations in the candidate group division and unstable detection results. Meanwhile, Yu et al. [20] proposed the GAD-MDST method based on maximum dense subtensor mining. This method can automatically generate multiscale user features by fusing a CNN and a feature pyramid network but is not suitable for detecting smaller-sized shilling groups. In our previous work [21], we proposed the TP-GBF method by using strongly correlated behaviors among group members and group behavior characteristics, which combined indirect behaviors with the direct collusion behaviors to highlight the collusive relationship between attackers in the same shilling group. TP-GBF performed well on the Netflix dataset but was less effective on the real dataset because it failed to detect smaller-sized attack groups.

For easy comparison of the above works, we summarize them in Table 1.

3. GCN-Based Group Shilling Attack Detection Model

Figure 1 depicts the two stages of the KC-GCN detection framework: influential user extraction and attack user identification. In the first stage, we build the user relationship graph by determining the user similarity based on the item suspicious time window. Next, we use the k-clique community discovery algorithm to generate suspicious candidate groups. Finally, we obtain the influential users by calculating the user nearest-neighbor similarity. In the second stage, we extract the users' initial embeddings from four dimensions. We then combined the extracted user initial embeddings with the structural features hidden in the user relationship graph to train a semi-supervised classifier based on a two-layer GCN, which only utilizes the labels of the identified influential users.

The notations used in this paper are described in Table 2.

3.1. Extracting Influential Users

3.1.1. Constructing a Weighted User Relationship Graph. Attackers in a shilling group typically cooperate to quickly enhance or demote the recommendation of one or more target items. Based on this characteristic of group attacks, the rating distribution of a target item may fluctuate during the attacked time period. Therefore, we construct a weighted user relationship graph by extracting the suspicious time windows of the suspicious items and calculating the correlation between users within the suspicious time windows.

Table 1: Comparison of different shilling attack detection methods.

Category	Approaches	Advantage	Disadvantage
Individual shilling attack detection methods	Chirita et al. [25] and Burke et al. [26]	Effective for specific attacks	Less effective under obfuscated attacks, e.g., AOP attack
	Zhang et al. [27]	A unified framework for detecting various shilling attacks	Require prior knowledge of attacks
	Zhang et al. [28]	Effective for a wide variety of shilling attacks	Less effective under the AOP attack
	Yang et al. [29]	A unified framework for detecting common access injection and shilling attacks	Require setting more parameters
	Hao and Zhang [30]	Automatic feature learning	High computational cost
	Ebrahimian and Kashef [31]	Regardless of the specific attacks	Unstable detection performance
	Zhou and Duan [32]	High detection precision	Require setting hyperparameters
	Zhang et al. [33]	Consider both user preference and reliability	Require multiple hyperparameters
Group attack detection methods	Zhou et al. [16]	Effective for detecting those shilling group profiles with high-correlation	Less effective for attack group profiles with a strong diversity
	Zhou et al. [17]	High detection precision	Low recall under attacks with a small attack size
	Zhang and Wang [18]	Excellent detection performance regardless of the number of target items	Less effective under smaller-sized groups
	Zhang et al. [19]	Automatic feature extraction	Unstable detection results
	Yu et al. [20]	Automatic feature extraction	Less effective under a smaller group size
	Cai and Zhang [21]	Effective for detecting tightly coupled shilling groups	Less effective under smaller-sized shilling groups on the Amazon dataset

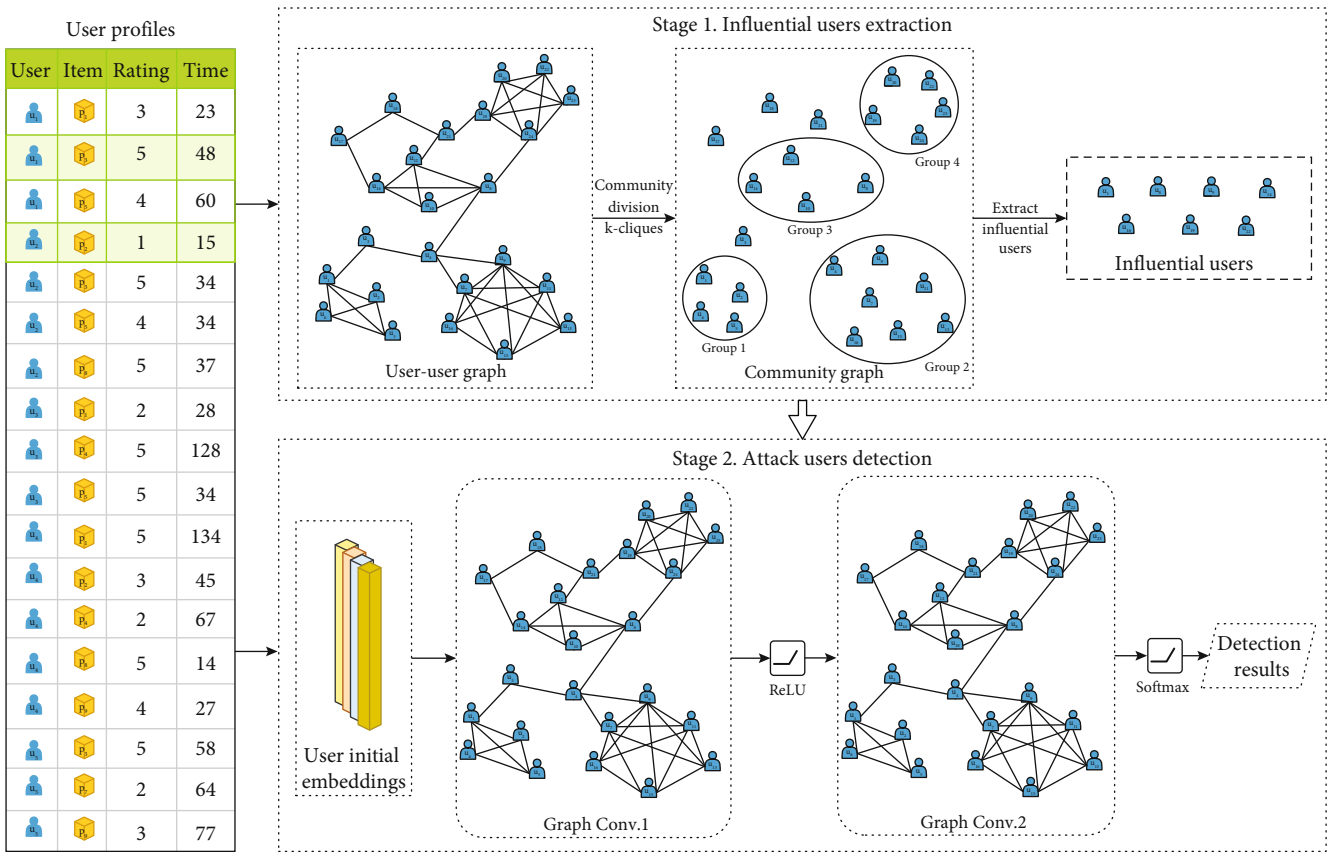


Figure 1: Detection framework of KC-GCN.

Table 2: Notations and their descriptions.

Notation	Description
$U = \{u_1, u_2, \cdots, u_m\}$	Set of users in the rating dataset
$P = \{p_1, p_2, \cdots, p_n\}$	Set of items in the rating dataset
$R = [r_{ij}]_{m\times n}$	User-item rating matrix
$T = [t_{ij}]_{m\times n}$	User-item rating time matrix
$W = \{w_1, w_2, \cdots, w_v\}$	Set of time windows in the rating dataset
$G = \langle U, E, G\rangle$	A weight user relationship graph
$\lvert\cdot\rvert$	The number of elements in a set
X	Users' initial embedding matrix

Definition 1 (item window abnormal degree, *IWAD*). For $\forall p \in P$ and $\forall w \in W$, the abnormal degree of item p on window m refers to the ratio of the number of users who rated item p with high ratings to the total number of users who rated it on time window w, which is referred to $\mathrm{IWAD}_{p,w}$ and calculated by

$$\mathrm{IWAD}_{p,w} = \frac{\sum_{u\in U} \Gamma(u,p,w)}{\mathrm{NR}_{p,w}}, \tag{1}$$

where $\mathrm{NR}_{p,w}$ represents the number of ratings of item p on the time window w. The time window is regarded as suspicious if $\mathrm{IWAD}_{p,w} > 0.5$. $\Gamma_{(u,p,w)}$ is an indicator function, which is formulated as

$$\Gamma_{u,p,w} = \begin{cases} 1, & \text{if } r_{u,p} \geq 4, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Definition 2 (user rating synchronization, URS). For $\forall u_i, u_j \in U$, their rating synchronization refers to how close their rating behavior is within the suspicious time window, which is denoted as $\mathrm{URS}(u_i, u_j)$ and calculated by

$$\mathrm{URS}(u_i, u_j) = \sum_{p\in N(u_i,u_j), r(u_i,p)\geq 4, r(u_j,p)\geq 4} \left(1 - \frac{t(u_i,p) - t(u_j,p)}{\tau}\right), \tag{3}$$

where $N(u_i, u_j)$ represents the set of items corated by users u_i and u_j, and the rating time is within the suspicious time window of item p;that is, $\mathrm{IWAD}_{p,w} > 0.5$.

Definition 3 (user short preference similarity, USPS). For $\forall u_i, u_j \in U$, their short preference similarity is defined as the ratio of $|N(u_i, u_j)|$ to $|N(u_i)| \cup |N(u_j)| - |N(u_i, u_j)|$, which is denoted as $\mathrm{USPS}(u_i, u_j)$ and calculated by

$$\mathrm{USPS}(u_i, u_j) = \frac{|N(u_i, u_j)|}{|N(u_i)| \cup |N(u_j)| - |N(u_i, u_j)|}, \tag{4}$$

where $N(u_i)$ and $N(u_j)$ represent the rating item set of users u_i and u_j, respectively. $|N(u_i, u_j)|$ represents the number of items for which user u_i and user u_j have the same preference within the suspicious time window, and $|N(u_i)| \cup |N(u_j)|$ represents the total number of items rated by user u_i and user u_j.

Definition 4 (user similarity, US). For $\forall u_i, u_j \in U$, their user similarity refers to the closeness of their rating times and similarity of their preferences on suspicious items, which is denoted as $\mathrm{US}(u_i, u_j)$ and calculated by

$$\mathrm{US}(u_i, u_j) = \mathrm{URS}(u_i, u_j) \times \mathrm{USPS}(u_i, u_j). \tag{5}$$

Based on the above definition, a weighted user relationship graph can be constructed. The weighted user relationship graph construction algorithm is described as follows.

Algorithm 1 is divided into two parts. The first part (lines 1–6) calculates the suspicious time window for each item, with a time complexity of $O(n * v)$. The second part (lines 7–21) calculates the relevance degree US of each user and constructs a user relationship graph, with a time complexity of $O(m^2)$. In conclusion, Algorithm 1 has a time complexity of about $O(m^2)$.

3.1.2. Extracting Influential Users. Li et al. [34] proposed the maximization problem that is aimed at selecting seed nodes from numerous nodes, thereby maximizing the influence of information on large-scale network transmission [35]. Inspired by the seed node idea, we only used the influential node labels to reduce the cost of labeling a large number of samples.

We present herein a two-stage method for extracting influential users. First, we generate suspicious candidate groups on the weighted user relationship graph using the k-clique algorithm [21]. Next, we extract influential users by calculating the user nearest-neighbor similarity.

The main steps of generating candidate groups based on the k-clique algorithm are as follows:

(1) Traverse each node in the user relationship graph to find a complete subgraph $G_i = \{u_1^i, u_2^i, \cdots, u_k^i\}$ containing k users, and add the users in G_i into the tightness community set TCS

(2) Convert the TCS into an overlapping community matrix O, where the diagonal elements in the matrix O represent the number of users in the community, and the off-diagonal elements represent the number of shared users in adjacent communities

(3) Merge the small communities in the overlapping community matrix O to obtain the community adjacency matrix A. In the matrix O, these diagonal elements with a value less than k and off-diagonal elements with a value less than $k-1$ are set to 0, while the left elements are set to 1

(4) Generate the suspicious candidate group based on the community adjacency matrix A

Input: the rating matrix R, the rating time matrix T, the size of sliding time window W_s, the time window anomaly threshold δ, and the relationship strength threshold σ
Output: a weighted user relationship graph G

```
1.  E ⟵ ∅ ; C ⟵ 0_{m×n}
2.  for each item p ∈ P do
3.      for each time window ∀w ∈ W do
4.          compute IWAD_{p,w} according to Eq. (1)
5.      end for
6.  end for
7.  for each user u_i ∈ U do
8.      for each user u_j ∈ U do
9.          compute URS(u_i, u_j) according to Eq. (3)
10.             compute USPS(u_i, u_j) according to Eq. (4)
11.             if URS(u_i, u_j) > σ then
12.                 compute US(u_i, u_j) according to Eq. (5)
13.                 C[u_i][u_j] ⟵ US(u_i, u_j)
14.                 E ⟵ E ∪ {(u_i, u_j)}
15.             else
16.                 US(u_i, u_j) ⟵ 0
17.             end if
18.         end for
19.     end for
20.     construct a weight user relationship graph G = ⟨U, E, C⟩
21.     return G
```

Algorithm 1: Constructing a weighted user relationship graph.

Definition 5 (user nearest neighbor similarity, *UNNS*). For $\forall u_i \in U$, the user's nearest neighbor similarity refers to the average similarity between the user and its neighbors, which is denoted as $\mathrm{UNNS}(u_i)$ and calculated by

$$\mathrm{UNNS}(u_i) = \frac{\sum_{u_j \in \mathrm{Neighbor}(u_i)} \mathrm{UR}(u_i, u_j)}{|\mathrm{Neighbor}(u_i)|}, \tag{6}$$

where $UR(u_i, u_j)$ represents the similarity of users u_i and u_j.

Definition 6 (influential user, IU). Influential users refer to those users whose nearest neighbor similarity is larger than that of all its first-order neighbors.

The algorithm for extracting influential users based on the k-clique algorithm and user nearest neighbor similarity is described as follows.

Algorithm 2 is divided into four parts. The first part (lines 1–12) identifies tight communities in the graph and generates a community relationship matrix with a time complexity of $O(m * l) + O(l^2)$. The second part (lines 13–19) merges communities to generate a community adjacency matrix with a time complexity of $O(l^2) + O(l^2)$. The third part (lines 20–24) generates candidate suspicious groups according to the community adjacency matrix, with a time complexity of $O(1)$. The last part (lines 25–29) extracts an influential user set based on the user nearest neighbor similarity, with a time complexity of $O(l * m)$. In conclusion, Algorithm 2 has a time complexity of about $O(m * l)$.

3.2. Detecting Attack Users

3.2.1. Generating User Initial Embeddings. Some node-embedding methods (e.g., matrix factorization and autoencoders) are automatic but usually generated using a randomization strategy and cannot represent well the initial node embeddings. Therefore, we generate the user embeddings herein from four perspectives.

Definition 7 (user lifetime proportion, ULP). For $\forall u_i \in U$, the user lifetime proportion refers to the ratio of the lifetime of user u_i in the system to the lifetime of the entire system, which is denoted as $\mathrm{ULP}(u_i)$ and calculated by

$$\mathrm{ULP}(u_i) = \frac{\mathrm{URT}(u_i, \max) - \mathrm{URT}(u_i, \min)}{\mathrm{SL}}, \tag{7}$$

where $\mathrm{URT}(u_i, \max)$ and $\mathrm{URT}(u_i, \min)$ represent the latest and the earliest rating time of user u_i, respectively. SL represents the lifetime of the entire system.

Definition 8 (user nearest neighbor rating synchronization, UNNRS). For $\forall u_i \in U$, the user's neighbor rating synchronization refers to the average rating synchronization between the user and its neighbors, which is denoted as $\mathrm{UNNRS}(u_i)$ and calculated by

$$\mathrm{UNNRS}(u_i) = \frac{\sum_{u_j \in \mathrm{Neighbor}(u_i)} \mathrm{URS}(u_i, u_j)}{|\mathrm{Neighbor}(u_i)|}, \tag{8}$$

where $\mathrm{URS}(u_i, u_j)$ means the synchronization degree of user

Input: the user's relationship graph $G = \langle U, E, C \rangle$, the size of smallest clique k
Output: set of influential users IUS

CSG $\longleftarrow \varnothing$; $O \longleftarrow 0_{|L|\times|L|}$; $A \longleftarrow 0_{|L|\times|L|}$
for each user $u_i \in U$ **do**
  if $G_i = \{u_1^i, u_2^i, \cdots, u_k^i\}$ and $G_i \subseteq G$ and $\forall C(u_m^i, u_n^i) \neq 0$ **then**
    $TG \longleftarrow TG \cup G_i$
  end if
end for
for $G_i \in TG$ **do**
  $O[i][i] \longleftarrow |G_i|$
  for $G_j \in TG$ **do**
    $O[i][j] \longleftarrow |G_i \cap G_j|$
  end for
end for
for $\forall i, j \in L$ and $i \neq j$ **do**
  if $O[i][i] < k$ or $O[i][j] \leq k - 1$ **then**
    $A[i][i] \longleftarrow 0$
    $A[i][j] \longleftarrow 0$
  else
    $A[i][i] \longleftarrow 1$
    $A[i][j] \longleftarrow 1$
  end if
end for
for $\forall i, j \in L$ **do**
  if $A[i][j] = 1$ **then**
    CSG $\longleftarrow G_i \cup G_j$
  end if
end for
for each community $cs \in$ CSG **do.**
  for each user $u \in cs$ **do**
    if $\text{UNNS}_u > \text{UNNS}_{\text{Neighbor}(u)}$ **then**
      IUS $\longleftarrow$ IUS $\cup \{u\}$
return IUS

Algorithm 2: Extracting influential users.

u_i and user u_j and $|\text{Neighbor}(u_i)|$ represents the number of first-order neighbors of user u_i.

Definition 9 (user nearest neighbor preference similarity, UNNPS). For $\forall u_i \in U$, the user's nearest neighbor preference refers to the average preference similarity between the user and its direct neighbors, which is denoted as $\text{UNNPS}(u_i)$ and calculated by

$$\text{UNNPS}(u_i) = \frac{\sum_{u_j \in \text{Neighbor}(u_i)} \text{UPS}(u_i, u_j)}{|\text{Neighbor}(u_i)|}, \tag{9}$$

where $\text{UPS}(u_i, u_j)$ represents the preference similarity of user u_i and user u_j.

Based on the above definition, we extract the initial embedding $X_u = (\text{ULP}_u, \text{UNNRS}_u, \text{UNNPS}_u, \text{UNNS}_u)$ of user u.

3.2.2. Identifying Attack Users Based on the GCN. In previous graph embedding-based group shilling attack detection methods, researchers focused on how to obtain high-quality user node embeddings in the graph. Zhang et al. [19] obtained a low-dimensional embedding vector of nodes in the graph by adopting the Node2vec method that focuses on obtaining the structural characteristics of the user's topological neighborhood but ignores the characteristic information of the nodes themselves. The existing group attack detection methods also use hard classification, in which members from the same group are classified as genuine users or attackers, resulting in the misclassification of some users [18–21]. To this end, we utilize user high-quality embedding features from both implicit and explicit perspectives by combining user initial embeddings with their higher-order topological neighborhood structures based on the GCN and employing influential node labels to identify attack users.

We first extract the high-quality embeddings of user nodes based on the user initial embedding matrix X and the weighted matrix C. The GCN propagation process is formulated as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{C}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right), \tag{10}$$

where $H^{(l+1)}$ represents the output after one convolutional layer. $H^{(l)}$ is the input of the lth layer. $H^{(0)} = X_{|N|\times 4}$

represents the user initial embedding matrix. N is the total number of user nodes in the graph, and the feature vector of each user is represented as $X_u = (\mathrm{ULP}_u, \mathrm{UNNRS}_u, \mathrm{UNNPS}_u, \mathrm{UNNS}_u)$. $\tilde{C} = C + I_N$ is the adjacency matrix by adding self-connection. $\tilde{D}$ is the degree matrix, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)} \in R^{P \times H}$ denotes the parameter matrix to be trained, P represents the length of the feature matrix, and H represents the number of hidden units. σ is the corresponding activation function, such as $\mathrm{Re\,LU}(\cdot) = \max(0, \cdot)$.

High-quality user embeddings can be obtained after multiple convolutional layers. We utilize two convolutional layers and ReLU as the activation function.

We then calculate the cross-entropy between the real label one-hot vector Y of all influential user nodes and the label vector T predicted by softmax. Subsequently, we utilize the gradient descent method to train the parameter matrix $W^{(0)}$ and $W^{(1)}$. The formula for calculating the loss function is as follows.

$$\mathrm{loss} = -\sum_{l \in \Upsilon_L} \sum_{t=1}^{T} Y_{lf} \ln Z_{lf}, \tag{11}$$

where Υ_L is the set of influential user nodes with labels.

Finally, the resulting model is expressed as

$$Z = f(X, C) = \mathrm{softmax}\left(\hat{C}\mathrm{ReLU}\left(\hat{C}XW^{(0)}\right)W^{(1)}\right), \tag{12}$$

where Z represents the set of user labels after classification by the softmax function. $\hat{C} = \tilde{D}^{-1/2}\tilde{C}\tilde{D}^{-1/2}$ represents the weighted matrix C after symmetric normalization.

The algorithm for detecting attackers is described as follows.

Algorithm 3 is divided into two parts. The first part (lines 1–8) uses GCN semi-supervised classification model training to get the classification result Z of all user nodes. The second part (lines 9–14) filters out the attack users according to the classification result Z.

4. Experimental Evaluation

4.1. Experimental Datasets. The following two datasets are utilized as the experimental datasets to evaluate how well the proposed KC-GCN method performs.

(1) *Netflix dataset (this dataset was constructed to support the participants in the Netflix prize (*http://netflixprize.com*))*: this dataset contains 1,032,938 ratings and the rating time for 17,770 movies by 480,186 users. The ratings are expressed in integers from 1 to 5, where 1 and 5 indicate disliked and most liked, respectively. We randomly sample 215,884 ratings and the rating time of 2000 users on 4000 movies for use in the experimental dataset. Similar to the previous research, the 2000 extracted users are regarded as genuine users. Multiple group attack profiles are generated and injected into the dataset by using the group shilling attack model introduced in Section 2.1. Under $\mathrm{GSAGen_l}$ Ave, $\mathrm{GSAGen_l}$ Ran, and $\mathrm{GSAGen_l}$ AOP, 10 attack groups are generated each time. The filler size is set to 2%, and the attack size is set to 2.5%, 5%, 7.5%, and 10%. The target items in each attack group are randomly selected from unpopular items. Two target item strategies are set (denoted as ST1 and ST2) to prove the influence of the relationship between the attack users in the same group on the detection performance. ST1 means that all attackers of the same group rate all the target items (number of target items in the experiments: 3). ST2 means that each attacker of the same group rate any three of the five target items. This results to $4 * 2 * 3 * 2 = 48$ experimental datasets generated. We generate loose group attack profiles to verify the universality of the proposed method using the $\mathrm{GSAGen_l}$ GOAT and $\mathrm{GSAGen_l}$ Mixed attack models introduced in Section 2.1 and two target item strategies. The dataset generated based on the $\mathrm{GSAGen_l}$ GOAT attack model specifically contains eight attack groups. The dataset generated based on the $\mathrm{GSAGen_l}$ Mixed attack model contains 26 attack groups. For convenience of description, under the condition of the target item strategies ST1 and ST2, the shilling attack groups generated are denoted as loosely and tightly coupled shilling groups, respectively

(2) *Amazon dataset [36]*: this dataset contains 1,205,125 ratings and the rating time on 136,785 products from 645,072 users crawled from Amazon.cn until August 20, 2012. The ratings are integers between 1 and 5, which indicate disliked and most liked, respectively. We evaluate the proposed method using a sampled dataset with 5055 labeled users. The dataset consists of 53,777 ratings of 17,610 products by 3118 genuine users and 1937 attack users

4.2. Evaluation Metrics. Three metrics including precision, recall, and F1-measure are used to evaluate the detection performance of the KC-GCN:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{13}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{14}$$

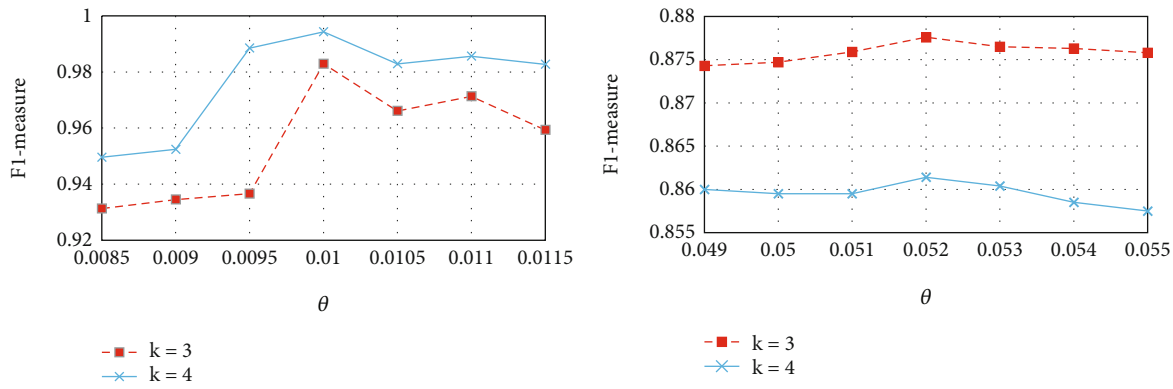
$$\mathrm{F1\text{-}measure} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{15}$$

where TP represents the number of attackers accurately recognized, FN represents the number of attackers mistaken for genuine users, and FP represents the number of genuine users mistaken for attackers.

4.3. Parameter Selection. Figure 2 shows how the F1-measure of the KC-GCN is influenced by parameters θ and k on the Netflix and Amazon datasets. In Figure 2(a), the F1-measure of KC-GCN is the highest for detecting the

Input: the weighted user's relationship graph G, the influential user's set IUS, the user initial embedding matrix X, and maximum training epoch K
Output: set of attack users AU
1. AU $\longleftarrow \varnothing$
2. **for** $k = 1$ to K **do**
3. compute Z according to Eq. (12)
4. compute loss according to Eq. (11)
5. Gradient zeroing
6. Back propagation calculation gradient value
7. Update parameters by using gradient descent
8. **end for**
9. **for** each $z \in Z$ **do**
10. **if** $z = 1$ **then**
11. AU $\longleftarrow$ AU $\cup \{z\}$
12. **end if**
13. **end for**
14. **return** AU

Algorithm 3: Detecting attack users.



(a) Netflix dataset

(b) Amazon dataset

Figure 2: The influence of parameters θ and k on the F1-measure of KC-GCN.

$GSAGen_l$ Ran attack on the Netflix dataset under a θ value set to 0.01. Under a smaller θ value, the user relationship graph contains a large number of weak relationship edges, and the community structure is not obvious, leading to a decrease of the detection precision. At a larger θ, the user relationship graph shows an obvious community structure, but some user nodes are filtered from the graph, thereby degrading the detection recall. Moreover, $k = 4$ has a superior detection performance than $k = 3$; therefore, for the Netflix dataset, we set k to 4 and θ to 0.01. Figure 2(b) shows that when $\theta = 0.052$ and $k = 3$, the F1-measure of KC-GCN is close to 0.8776 on the sampled Amazon dataset, which is the best. Therefore, we set k to 3 and θ to 0.052 for the sampled Amazon dataset.

4.4. Experimental Results and Analysis. To verify the effectiveness of KC-GCN, we compare the precision, recall, and F1-measure of KC-GCN with the following methods.

We assess the precision, recall, and F1-measure of KC-GCN in comparison to the following methods to confirm its efficacy.

(1) *Catch the Black Sheep (CBS) [27]*: this detection method uses label propagation to iteratively calculate the malicious probability of users and items, which needs the number of spammers and a certain number of seed users in advance. In contrast to the experiments, the number of seed users on the two datasets is consistent with that of our method

(2) *GAGE [19]*: this is an unsupervised group shilling attack detection method based on graph embedding, which learns the low-dimensional vector representation of nodes in the user relationship graph using Node2vec and obtains attack groups through clustering. In the experiments, the working strategy is adjusted by setting parameters $p = 7$ and $q = 0.2$ and group size (GS) = 30

(3) *TP-GBF [20]*: this is an unsupervised group shilling attack detection method based on the strong association between the group members and the group behavior features, which uses a topological potential-based community partition algorithm to

generate tight subgraphs as candidate groups and cluster attack groups by group behavior features. In the experiments, parameter θ is set to 2, while parameter σ is set to 1 and 0.47 in the Netflix and Amazon datasets, respectively

4.4.1. Comparison of the Detection Results on the Netflix Dataset. Table 3 compares KC-GCN and three baseline methods to identify the group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset. In Table 3, the precision and recall values of the CBS remain stable for detecting three types of group shilling attacks, only slightly changing the attack size from 2.5% to 10%. The CBS detection performance is much lower than that of KC-GCN when detecting various types of group shilling attacks with tightly coupled shilling groups, albeit the number of attackers is assumed in advance. Meanwhile, the precision values of GAGE under three types of group attacks are the worst, indicating the misclassification of a large number of genuine users as attack ones. This happened because GAGE generates the user node feature vectors using Node2-vec, from which a certain degree of randomness may cause some genuine and attack users to be divided into the same candidate group. The detection performance of TP-GBF is better than those of CBS and GAGE when detecting the group shilling attacks with tightly coupled shilling groups. The detection recall was not high under the GSAGen_l AOP. Compared with CBS, GAGE, and TP-GBF, KC-GCN shows the best detection performance because it can extract more effective features when correctly differentiating attack profiles from genuine ones. KC-GCN uses a weighted graph to aggregate the neighbor features, thereby effectively avoiding the merging of user features with different labels. It can fully integrate the user node and structural features, further increasing the difference between attackers and normal users. In conclusion, KC-GCN outperforms the baselines for detecting various types of group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset.

Table 4 compares the performances of our proposed KC-GCN and three baseline methods in terms of detecting group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset. In Table 4, the precision and recall values of CBS under the three attack models significantly decrease when the relationship between users within the attack group is weakened. This indicates that improving the detection performance is difficult when relying only on the rating bias. The GAGE performance becomes better with the attack size increase, but its precision greatly fluctuates because it may falsely identify some normal users as attackers. TP-GBF shows an excellent detection performance under the GSAGen_l Ran and GSAGen_l Ave attacks but is less effective under the GSAGen_l AOP attack. Its detection performance becomes extremely unstable with the change of the attack size. KC-GCN yields the best detection performance among the four methods. It shows a slight decline in detecting loosely coupled shilling groups mainly because the feature differences between the attackers and the genuine users are weakened with a looser relationship in a group. In conclusion, KC-GCN outperforms the baseline methods in detecting various types of group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset.

Figure 3 compares the detection results of the four detection methods under the GSAGen_l GOAT attack on the Netflix dataset. In the Netflix dataset, the precision, recall, and F1-measure of CBS when identifying tightly and loosely coupled shilling groups are 0.6791, 0.8184, and 0.7422 and 0.6352, 0.7656, and 0.6943, respectively. The detection performance of CBS is constrained by the number and influence of seed users. These results also indicate that CBS can achieve superior detection performance when a closer relationship exists between the group members. The precision, recall, and F1-measure of GAGE for detecting the tightly and loosely coupled shilling groups are 0.4046, 0.6968, and 0.5119 and 0.3157, 0.9120, and 0.4690, respectively. These results indicate that GAGE is less effective on the Netflix dataset under the GSAGen_l GOAT attack because the GOAT attack model uses the genuine user profile as a template to generate the attack profile, which is highly similar to the genuine user. However, the user node feature vector obtained by the Node2vec method cannot effectively distinguish genuine users and attackers. For TP-GBF, the precision, recall, and F1-measure of the tightly coupled shilling groups are 0.9905, 0.7269, and 0.8385, respectively, while those for the loosely coupled shilling groups are 0.7036, 0.5000, and 0.5846, respectively. TP-GBF shows an extremely high precision in identifying the tightly coupled shilling groups; nevertheless, the recall of TP-GBF is poor when identifying the loosely coupled shilling groups because it cannot distinguish weakly related attack groups. Figure 3 shows that GAGE and TP-GBF have poor performances when detecting attack groups generated based on GOAT because the profiles generated by GOAT are very similar to the genuine profiles. The precision, recall, and F1-measure of KC-GCN for detecting the tightly coupled shilling groups are 1, 0.9857, and 0.9928, respectively, while those for the loosely coupled shilling groups are 1, 0.9282, and 0.9628, respectively. These results show that KC-GCN is effective and outperforms the three baseline methods for detecting groups under the GSAGen_l GOAT attack on the Netflix dataset. In other words, the feature differences between the attackers and the genuine users can be reinforced by using the weighted GCN to aggregate the user node features.

Figure 4 compares the detection results of the four detection methods under the GSAGen_l Mixed attack on the Netflix dataset. In this dataset, the precision, recall, and F1-measure of CBS for identifying the tightly and loosely coupled shilling groups are 0.8190, 0.9992, and 0.9002 and 0.8191, 0.9996, and 0.9004, respectively. CBS remains stable when detecting the tightly and loosely coupled group shilling attacks. GAGE shows precision, recall, and F1-measure of 0.9542, 0.9275, and 0.9407, respectively, for the tightly coupled shilling groups. For the loosely coupled shilling groups, the precision, recall, and F1-measure of GAGE are 0.8212, 0.9376, and 0.8759, respectively. Its detection performance significantly declines with the weakening user relationships. The main reason for this is that with the

Table 3: Comparison between KC-GCN and other detection methods for detecting group shilling attacks with tightly coupled shilling groups at various attack sizes on the Netflix dataset.

Attack type	Metrics	Method	Attack size			
			2.5%	5%	7.5%	10%
$GSAGen_l$ Ran	Precision	CBS	0.7811	0.7979	0.7974	0.8019
		GAGE	0.5630	0.8906	0.7856	0.9124
		TP-GBF	0.9944	1.0000	0.9987	0.9973
		KC-GCN	0.9954	0.9937	0.9965	1.0000
	Recall	CBS	0.9716	0.9879	0.9870	0.9912
		GAGE	0.9918	0.9904	0.9815	0.9696
		TP-GBF	0.9152	0.9428	0.9054	0.8189
		KC-GCN	0.9487	0.9822	0.9725	0.9886
	F1-measure	CBS	0.8660	0.8828	0.8821	0.8866
		GAGE	0.7183	0.9379	0.8727	0.9401
		TP-GBF	0.9532	0.9706	0.9498	0.8993
		KC-GCN	0.9715	0.9879	0.9844	0.9943
$GSAGen_l$ Ave	Precision	CBS	0.8045	0.8076	0.8125	0.8098
		GAGE	0.8584	0.7445	0.8913	0.7624
		TP-GBF	0.9944	0.9870	1.0000	0.9989
		KC-GCN	0.9925	0.9932	0.9918	0.9986
	Recall	CBS	0.9797	0.9815	0.9865	0.9838
		GAGE	0.9876	0.9856	0.8518	0.9748
		TP-GBF	0.9383	0.9159	0.8941	0.9782
		KC-GCN	0.9500	0.9735	0.9871	0.9914
	F1-measure	CBS	0.8835	0.8861	0.8911	0.8884
		GAGE	0.9185	0.8483	0.8711	0.8556
		TP-GBF	0.9655	0.9501	0.9441	0.9884
		KC-GCN	0.9708	0.9833	0.9894	0.9950
$GSAGen_l$ AOP	Precision	CBS	0.6915	0.7035	0.7242	0.7214
		GAGE	0.7051	0.7479	0.806	0.7507
		TP-GBF	0.9764	0.9837	0.9854	0.9914
		KC-GCN	0.9740	0.9875	0.9773	0.9886
	Recall	CBS	0.8512	0.8657	0.8866	0.8843
		GAGE	0.9806	0.9322	0.9725	0.9679
		TP-GBF	0.8118	0.8025	0.8477	0.7744
		KC-GCN	0.9615	0.9080	0.9556	0.9255
	F1-measure	CBS	0.7631	0.7762	0.7972	0.7946
		GAGE	0.8203	0.8299	0.8815	0.8456
		TP-GBF	0.8865	0.8839	0.9114	0.8696
		KC-GCN	0.9677	0.9461	0.9663	0.9560

weakening user relationship in the group, its spatial structure changes, resulting in obvious changes in the initial user embedding and a significant decrease in the detection performance. The precision, recall, and F1-measure of TP-GBF for detecting tightly coupled shilling groups are 0.7209, 0.8204, and 0.7674, respectively, while those for loosely coupled shilling groups are 0.6620, 0.6141, and 0.6372, respectively. TP-GBF is less effective on the Netflix dataset under the $GSAGen_l$ Mixed attack. The precision, recall, and F1-measure of KC-GCN for identifying the tightly and loosely coupled shilling groups are 0.9978, 0.9430, and 0.9696 and 0.9583, 0.9705, and 0.9644, respectively. These findings demonstrate that KC-GCN is effective and outperforms the three baseline methods in terms of precision and F1-measure under the $GSAGen_l$ Mixed attack on the Netflix dataset.

Figure 5 shows the results of the four detection methods on the sampled Amazon dataset. The detection performance of KC-GCN is superior to that of the baseline methods on this dataset, yielding precision, recall, and F1-measure of

Table 4: Comparison between KC-GCN and other detection methods for detecting group shilling attacks with loosely coupled shilling groups at various attack sizes on the Netflix dataset.

Attack type	Metrics	Method	Attack size			
			2.5%	5%	7.5%	10%
$GSAGen_l$ Ran	Precision	CBS	0.7993	0.6441	0.6422	0.6479
		GAGE	0.8749	0.8102	0.8749	0.9078
		TP-GBF	1.0000	0.9985	0.8247	0.6856
		KC-GCN	0.9683	0.9852	0.9858	0.9593
	Recall	CBS	0.9730	0.7835	0.7796	0.7866
		GAGE	0.9459	0.9868	0.9509	0.9933
		TP-GBF	0.8555	0.8242	0.8451	0.7768
		KC-GCN	0.9313	0.8965	0.8910	0.9940
	F1-measure	CBS	0.8777	0.7070	0.7042	0.7105
		GAGE	0.9090	0.8898	0.9113	0.9486
		TP-GBF	0.9221	0.9030	0.8348	0.7284
		KC-GCN	0.9494	0.9399	0.9360	0.9763
$GSAGen_l$ Ave	Precision	CBS	0.7972	0.6439	0.6439	0.6487
		GAGE	0.6345	0.8544	0.8491	0.9070
		TP-GBF	0.4497	0.5362	0.5939	0.9137
		KC-GCN	0.9542	0.9783	0.9800	0.9600
	Recall	CBS	0.9723	0.7823	0.7823	0.7879
		GAGE	0.9884	0.9482	0.9900	0.9138
		TP-GBF	0.7055	0.7977	0.7687	0.7704
		KC-GCN	0.9843	0.9184	0.9188	0.9941
	F1-measure	CBS	0.8761	0.7064	0.7064	0.7115
		GAGE	0.7729	0.8988	0.9142	0.9104
		TP-GBF	0.5493	0.6413	0.6701	0.8360
		KC-GCN	0.9690	0.9474	0.9484	0.9767
$GSAGen_l$ AOP	Precision	CBS	0.5401	0.5624	0.5655	0.5569
		GAGE	0.5501	0.6043	0.4545	0.7517
		TP-GBF	0.8733	0.5643	0.7359	0.6868
		KC-GCN	0.9589	0.9294	0.9620	0.9540
	Recall	CBS	0.6664	0.6906	0.6944	0.6824
		GAGE	0.9151	0.9170	0.9444	0.9372
		TP-GBF	0.7773	0.7208	0.7655	0.7361
		KC-GCN	0.9211	0.9875	0.9048	0.9432
	F1-measure	CBS	0.5967	0.6200	0.6233	0.6133
		GAGE	0.6871	0.7285	0.6134	0.8343
		TP-GBF	0.8225	0.6330	0.7504	0.7106
		KC-GCN	0.9396	0.9576	0.9325	0.9486

0.9179, 0.8407, and 0.8776, respectively. This indicates that KC-GCN can effectively combine user node and graph structure features to construct new user features by using GCN, which can distinguish genuine and attack users on the sampled Amazon dataset. The precision, recall, and F1-measure of CBS are 0.6836, 0.8323, and 0.7507, respectively. This means that CBS can detect attack users on the Amazon dataset but that its detection performance is determined by the number of seed users. Meanwhile, GAGE exhibits 0.8004, 0.9277, and 0.8594 of precision, recall, and F1-measure, respectively. The result indicates that GAGE has a certain randomness when sampling with Node2vec, which leads to a bias in the division of the candidate groups, and a precision measurement performance is lower than that of KC-GCN. The precision, recall, and F1-measure of TP-GBF are 0.9283, 0.6467, and 0.7623, respectively. This precision is not much higher than that of KC-GCN, but its recall is lower than that of KC-GCN, indicating that TP-GBF may have filtered out some attack groups with a low density. In summary, KC-GCN shows a superior detection performance over GAGE, CBS, and TP-GBF on the sampled Amazon dataset.

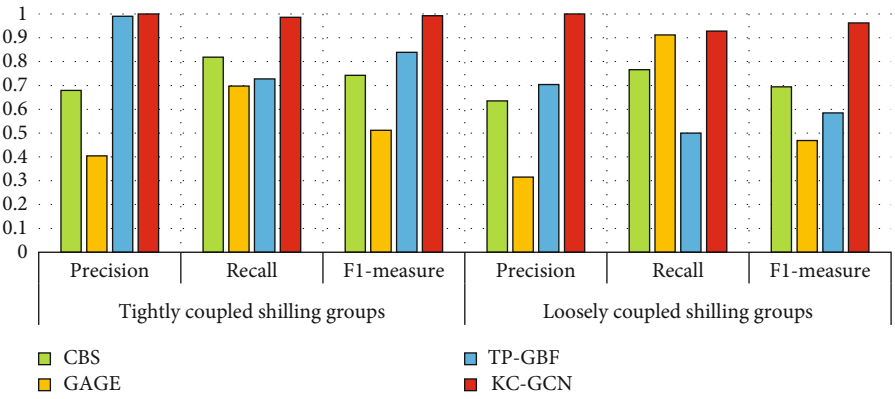


Figure 3: Comparison of the detection results of the four detection methods on the Netflix dataset under the $GSAGen_l$ GOAT attack.

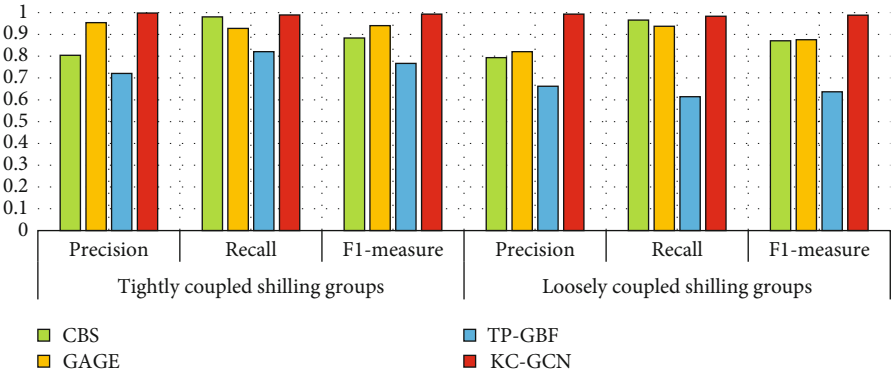


Figure 4: Comparison of the detection results of the four detection methods on the Netflix dataset under the $GSAGen_l$ Mixed attack.

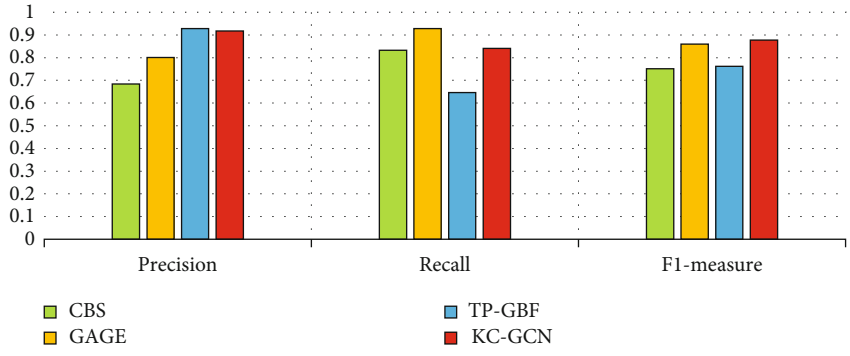


Figure 5: Comparison of the detection results of the four detection methods on the Amazon dataset.

5. Conclusions and Future Work

In this work, we put forward a two-stage semi-supervised model to validly detect various types of group shilling attacks on recommender systems. First, we construct a user relationship graph and spot the influential users. In the graph, the edge weight is calculated by analyzing the user similarity over suspicious time intervals on each item. Next, we generate the initial user embeddings based on the proposed four indicators describing the behavior difference between attack and genuine users. A GCN-based classifier is trained, and the attack users are detected based on the influential user labels. The experimental results prove the effectiveness and the generality of KC-GCN.

In the future work, we will automatically determine the labels of most influential users by further analyzing the structural properties of the weighted user relationship graph. We will also study the multiaspect data [37] to further help identify users of group shilling attack.

Data Availability

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

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Science and Technology Project of the Hebei Education Department (ZD2022105), the Natural Science Foundation of Hebei Province, China (F2020201023), and the High-Level Personnel Starting Project of Hebei University (521100221089).

References

[1] H. Li, K. Wang, Y. Sun, and X. Mou, "Application of recommendation systems based on deep learning," in *Recent Challenges in Intelligent Information and Database Systems. ACIIDS 2021*, Communications in Computer and Information Science, T. P. Hong, K. Wojtkiewicz, R. Chawuthai, and P. Sitek, Eds., pp. 85–97, Springer, Singapore, 2021.

[2] Q. Shambour, "A deep learning based algorithm for multi-criteria recommender systems," *Knowledge-Based Systems*, vol. 211, article 106545, 2021.

[3] N. Nassar, A. Jafar, and Y. Rahhal, "A novel deep multi-criteria collaborative filtering model for recommendation system," *Knowledge-Based Systems*, vol. 187, no. 7, pp. 104811.1–104811.7, 2020.

[4] Y. Feng, F. Lv, W. Shen et al., "Deep session interest network for click-through rate prediction," 2019, https://arxiv.org/abs/1905.06482.

[5] F. Lv, T. Jin, C. Yu et al., "SDM: sequential deep matching model for online large-scale recommender system," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2635–2643, Beijing, China, 2019.

[6] S. Zhang, H. Liu, J. He, S. Han, and X. Du, "Deep sequential model for anchor recommendation on live streaming platforms," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 173–182, 2021.

[7] P. Nitu, J. Coelho, and P. Madiraju, "Improvising personalized travel recommendation system with recency effects," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 139–154, 2021.

[8] T. Li, C. Li, J. Luo, and L. Song, "Wireless recommendations for internet of vehicles: recent advances, challenges, and opportunities," *Intelligent and Converged Networks*, vol. 1, no. 1, pp. 1–17, 2020.

[9] 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 103051, 2021.

[10] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pp. 149–156, New York, USA, 2009.

[11] 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.

[12] X. Su, H. Zeng, and Z. Chen, "Finding group shilling in recommender system," in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 960-961, China, 2005.

[13] L. Yang and X. Niu, "A genre trust model for defending shilling attacks in recommender systems," *Complex & Intelligent Systems*, pp. 1–14, 2021.

[14] S. Rani, M. Kaur, M. Kumar, V. Ravi, U. Ghosh, and J. R. Mohanty, "Detection of shilling attack in recommender system for YouTube video statistics using machine learning techniques," *Soft Computing*, vol. 27, pp. 377–389, 2023.

[15] 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.

[16] W. Zhou, Y. S. Koh, J. Wen, S. Alam, and G. Dobbie, "Detection of abnormal profiles on group attacks in recommender systems," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, Gold Coast Queensland, Australia, 2014.

[17] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems," *Neurocomputing*, vol. 210, no. 19, pp. 197–205, 2016.

[18] 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.

[19] 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 105984, 2020.

[20] 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 *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Melbourne, Australia, 2021.

[21] 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. 2021, Article ID 2907691, 18 pages, 2021.

[22] Y. Gao, X. Yu, and H. Zhang, "Uncovering overlapping community structure in static and dynamic networks," *Knowledge-Based Systems*, vol. 201-202, p. 106060, 2020.

[23] J. Zhou, G. Cui, S. Hu et al., "Graph neural networks: a review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[24] Y. Wang, Z. Wu, J. Cao, and C. Fang, "Towards a tricksy group shilling attack model against recommender systems," in *Advanced Data Mining and Applications. ADMA 2012*, S. Zhou, S. Zhang, and G. Karypis, Eds., vol. 7713 of Lecture Notes in Computer Science, pp. 675–688, Springer, Berlin, Heidelberg, 2012.

[25] P. A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pp. 67–74, New York, USA, 2005.

[26] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Detecting profile injection attacks in collaborative recommender systems," in *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, p. 23, San Francisco, USA, 2006.

[27] Y. Zhang, Y. Tan, M. Zhang, and Y. Liu, "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, Bunos Aires, Argentina, 2015.

[28] F. Zhang, Z. Zhang, P. Zhang, and S. Wang, "UD-HMM: an unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering," *Knowledge-Based Systems*, vol. 148, pp. 146–166, 2018.

[29] Z. Yang, Q. Sun, and Y. Zhang, "Probabilistic inference and trustworthiness evaluation of associative links toward malicious attack detection for online recommendations," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 1–896, 2020.

[30] 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.

[31] M. Ebrahimian and R. Kashef, "Detecting shilling attacks using hybrid deep learning models," *Symmetry*, vol. 12, no. 11, pp. 1805–1821, 2020.

[32] Q. Zhou and L. Duan, "Semi-supervised recommendation attack detection based on co-forest," *Computers & Security*, vol. 109, article 102390, 2021.

[33] S. Zhang, H. Yin, and T. Chen, "GCN-based user representation learning for unifying robust recommendation and fraudster detection," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 689–698, Xi'an, China, 2020.

[34] Y. Li, J. Fan, Y. Wang, and K. L. Tan, "Influence maximization on social graphs: a survey," *IEEE Transactions on Knowledge & Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.

[35] X. Liu, S. Wu, C. Liu, and Y. Zhang, "Social network node influence maximization method combined with degree discount and local node optimization," *Social Network Analysis and Mining*, vol. 11, no. 1, article 31, 2021.

[36] C. Xu, J. Zhang, and C. Long, "Uncovering collusive spammers in Chinese review websites," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pp. 979–988, Burlingame, USA, 2013.

[37] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.