RESEARCH

Open Access

Enhancing fairness of trading environment: discovering overlapping spammer groups with dynamic co-review graph optimization

Chaoqun Wang[1], Ning Li[1], Shujuan Ji[1*], Xianwen Fang[2] and Zhen Wang[1]

Abstract
Within the thriving e-commerce landscape, some unscrupulous merchants hire spammer groups to post misleading reviews or ratings, aiming to manipulate public perception and disrupt fair market competition. This phenomenon has prompted a heightened research focus on spammer groups detection. In the e-commerce domain, current spammer group detection algorithms can be classified into three categories, i.e., Frequent Item Mining-based, graph-based, and burst-based algorithms. However, existing graph-based algorithms have limitations in that they did not adequately consider the redundant relationships within co-review graphs and neglected to detect overlapping members within spammer groups. To address these issues, we introduce an overlapping spammer group detection algorithm based on deep reinforcement learning named DRL-OSG. First, the algorithm filters out highly suspicious products and gets the set of reviewers who have reviewed these products. Secondly, taking these reviewers as nodes and their co-reviewing relationships as edges, we construct a homogeneous co-reviewing graph. Thirdly, to efficiently identify and handle the redundant relationships that are accidentally formed between ordinary users and spammer group members, we propose the Auto-Sim algorithm, which is a specifically tailored algorithm for dynamic optimization of the co-reviewing graph, allowing for adjustments to the reviewers' relationship network within the graph. Finally, candidate spammer groups are discovered by using the Ego-Splitting overlapping clustering algorithm, allowing overlapping members to exist in these groups. Then, these groups are refined and ranked to derive the final list of spammer groups. Experimental results based on real-life datasets show that our proposed DRL-OSG algorithm performs better than the baseline algorithms in Precision.

Keywords Spammer groups, Homogeneous network, Redundant relationships, Overlapping members, Deep reinforcement learning, Ego-splitting algorithm

Introduction
In recent years, online live streaming has amplified users' activities on e-commerce platforms, emphasizing the role of product reviews in influencing consumer choices (Dewang and Singh 2018; Chen et al. 2022). However, malicious sellers employ spammers to post deceptive reviews, compromising trust in e-commerce platforms and impacting the social environment (Luca and Zervas 2016). Motivated by profit, these sellers collaborate with coordinated spammer groups to systematically post fraudulent reviews, influencing product perceptions (Mukherjee et al. 2012). By controlling public sentiment,

*Correspondence:
Shujuan Ji
jsjsuzie@sina.com
[1] Shandong Provincial Key Laboratory of Wisdom Mine Information Technology, Shandong University of Science & Technology, Qingdao 266590, Shandong, China
[2] Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Anhui University of Science and Technology, Huainan, China

these groups considerably affect consumer behavior. Addressing the threats posed by these organized groups is crucial for the fairness of the trading environment on e-commerce platforms.

Given the increasing activities of spammer groups, numerous researchers have been motivated to develop algorithms to detect these groups to mitigate their detrimental impact (Mukherjee et al. 2012; Xu et al. 2013; Xu and Zhang 2015; Wang et al. 2016, 2018a; Choo et al. 2015; Hu et al. 2019; Zhang et al. 2020, 2022a, b, 2023; Akoglu et al. 2013; Ye and Akoglu 2015; Zheng et al. 2018; Zhu et al. 2019; Shehnepoor et al. 2021; Chao et al. 2022; Li et al. 2017; Ji et al. 2020; Liu et al. 2018). According to the approach of generating candidate groups, existing spammer group detection algorithms can be classified into Frequent Item Mining (FIM)-based, graph-based, and burst-based algorithms. The FIM-based algorithms (Mukherjee et al. 2012; Xu et al. 2013; Xu and Zhang 2015) aim to capture repetitive fraudulent actions of reviewers to identify spammer groups. The graph-based algorithms (Wang et al. 2016, 2018a; Choo et al. 2015; Hu et al. 2019; Zhang et al. 2020, 2022a, b; Akoglu et al. 2013; Ye and Akoglu 2015; Zheng et al. 2018; Zhu et al. 2019; Shehnepoor et al. 2021; Chao et al. 2022) leverage the network relationships of reviewers to construct either homogeneous or heterogeneous graphs and then, based on the relational or behavioral information of the graph, generate candidate groups through clustering or community detection techniques. The burst-based algorithms (Li et al. 2017; Ji et al. 2020; Liu et al. 2018) capture the burst intervals of reviews from either reviewers or products, using these intervals to determine candidate groups.

Though numerous existing graph-based algorithms have demonstrated impressive performance, they still exhibit two notable limitations. Firstly, with the widespread adoption of recommendation systems, genuine reviewers may inadvertently post reviews for the same products as members of spammer groups or provide similar ratings in their daily activities. This seemingly innocuous behavior introduces significant, redundant information into the co-reviewing graph. These superfluous relationships increase the co-reviewing graph's complexity and adversely impact the efficiency and the accuracy of detection algorithms. Previous research has primarily concentrated on the similarity between products co-reviewed by reviewers, employing these similarities as edge weights in the co-reviewing graph. However, such studies largely neglected the inherent redundant relationships and their potential implications on the performance of detection algorithms. For instance, Wang et al. (2016) exclusively relied on the volume of products co-reviewed by reviewers within a specific time frame to construct their graph. This approach overly simplified the criteria for establishing co-reviewing relationships, leading to an intricate co-reviewing graph. Wang et al. (2018a) considered factors like the co-reviewing time intervals and rating biases among reviewers when constructing their graphs. However, they merely adjusted the graph's structure using the similarity of co-reviewed products as edge weights, offering insufficient attention to the redundant relationships within the co-reviewing graph. Similarly, Zhang et al. (2020) formed a reviewer relationship graph based on analyzing review timings and product scores but overlooked the inherent redundant relationships in the co-reviewing graph.

Secondly, spammer groups within e-commerce platforms may operate independently or under coordinated management. Since they originate from the same source, spammer groups driven by specific organizations often exhibit pronounced behavioral consistency, resulting in heightened disruption (Shehnepoor et al. 2022). Moreover, given the limitations in the number of members within an organization, there may be overlapping members among different groups (Gabardo et al. 2019). Overlapping members within spammer groups might represent core members of the spammer organization, possibly orchestrating or influencing the activities of multiple spammer groups. Nevertheless, most existing algorithms (Wang et al. 2016, 2018a; Choo et al. 2015; Ye and Akoglu 2015; Ji et al. 2020; Liu et al. 2018; Zhang et al. 2023) have yet to adequately address situations where spammer groups have overlapping members, which might curtail these algorithms' comprehensive and profound understanding of spammer behaviors.

According to the above analysis, the designing of a novel detection algorithm that adequately considers and reduces redundant information in co-reviewing relationships while precisely identifying spammer groups with overlapping members is of great concern. To address this problem, we present the DRL-OSG algorithm, a spammer group detection technique grounded on homogeneous information networks and deep reinforcement learning, to address these challenges. First, we evaluate the suspicion level of each product by leveraging the Network Footprint Score (NFS) metric (Ye and Akoglu 2015), subsequently filtering out the set of products deemed suspicious. Secondly, from this refined product set, we extract co-reviewing relationships, constructing a homogeneous co-reviewing graph concerning reviewers of suspicious products. Thirdly, we introduce Auto-Sim, an algorithm explicitly tailored for the dynamic optimization of the co-reviewing graph. Through Auto-Sim, we dynamically restructure the reviewer relationship network within the graph, significantly diminishing redundant connections and thus enhancing the overall cohesiveness and structure of the co-reviewing graph. Fourthly, based on the optimized graph, the Ego-Splitting overlapping clustering algorithm (Epasto et al. 2017) is employed to segment the graph according to the relationship, yielding candidate spammer groups potentially featuring overlapping members. Fifthly, we incorporate five distinctive fraudulent behavioral features to refine these overlapping candidate groups, effectively weeding out members with low suspicion ratings. Sixthly, drawing upon five spammer group-centric features, we categorize the refined candidate groups, pinpointing the definitive spammer groups.

The contributions of this paper can be described as follows:

(1) *Proposes a novel optimization algorithm for co-reviewing graphs.* Existing graph-based algorithms predominantly focus on utilizing the similarity of co-reviewed products among reviewers as the edge weights in the co-reviewing graph to adapt its structure (Wang et al. 2018a; Hu et al. 2019; Zhang et al. 2020). Different from these literatures, our algorithm uses a behavioral consistency index that combines both product consistency and rating consistency in their scores to remove unnecessary co-reviewing connections within the graph. Additionally, current graph-based algorithms usually set the edge weight parameters in the co-reviewing graph by manually adjusting them (Wang et al. 2018a; Hu et al. 2019; Zhang et al. 2020). In contrast, our algorithm introduces deep reinforcement learning algorithms to adjust the behavioral consistency parameters automatically, aiming to create a more optimized co-reviewing graph.

(2) *Discovers overlapping groups.* In contrast to existing algorithms that assume a spam member can only belong to a single spammer group (Wang et al. 2016, 2018a; Choo et al. 2015; Ye and Akoglu 2015; Ji et al. 2020; Liu et al. 2018; Zhang et al. 2023), we acknowledge the real-world scenario where spam members may engage in various fraudulent activities across different spammer groups. We employed an overlapping clustering algorithm to cluster the optimized co-reviewing graph to address this. This process involved segmenting the neighbor network of each reviewer node to form multiple sub-candidate groups and then incorporating the central node into each sub-candidate group, thereby creating candidate groups with overlapping members. Detecting spammer groups with overlapping members facilitates the exploration of the evolutionary dynamics of these groups. This approach enables a deeper understanding of the development trajectories of spammer groups, thereby enhancing the comprehensiveness of spammer group detection methodologies.

(3) *Four sets of experiments were designed and implemented on real-world datasets.* The first set of experiments verifies the rationality of the parameter selection. The second set of the experiment suggests that our approach can effectively eliminate redundant relations within the co-reviewing graph, significantly enhancing the accuracy of the detection algorithm. The third set of experiments reveal that the spammer groups identified by our algorithm are characterized by their large size and tight interconnections among members, and it also provides preliminary detection of spammer groups with overlapping members. The outcomes of the fourth experiment underscore the pivotal role of the co-reviewing graph optimization module in our algorithm and highlight the importance of eliminating redundant relations in the co-reviewing graph.

The rest of the paper is organized as follows. Sect. "Related works" reviews the related research on spammer group detection. Sect. "Framework of the overlapped spammer group detection algorithm" elaborates on our DRL-OSG algorithm in detail. Sect. "Experiment and result analysis" provides experimental results and comparative analysis. Finally, we summarize the full text and propose future directions for development in Sect. "Conclusion".

Relatesd works

In recent years, as the number of users on e-commerce sites has grown quickly, the need to fight against harmful actions, especially those from spammer groups, has increased. As a result, the work of finding these spammer groups on e-commerce platforms has become very important and has also grown fast.

Existing methods for detecting spammer groups can be distinguished based on two classification criteria, i.e., the distinct methods of generating candidate groups and the features used in generating candidate groups. According to the distinct methods of generating candidate groups, existing detection algorithms can be classified into three categories such as the FIM-based algorithms (Mukherjee et al. 2012; Xu et al. 2013; Xu and Zhang 2015), the graph-based algorithms (Wang et al. 2016, 2018a; Choo et al. 2015; Hu et al. 2019; Zhang et al. 2020, 2022a, 2022b; Akoglu et al. 2013; Ye and Akoglu 2015; Zheng et al. 2018; Zhu et al. 2019; Shehnepoor et al. 2021; Chao et al. 2022), and the burst-based algorithms (Li et al. 2017; Ji et al. 2020; Liu et al. 2018). According to the features employed to generate candidate groups, existing detection algorithms can also be divided into three categories: the ones based on group behavioral features (Mukherjee et al. 2012; Xu et al. 2013; Xu and Zhang 2015; Li et al. 2017; Ji et al. 2020) such as review content, timing, and ratings (which emphasizes the analysis of behavior patterns of the group), the ones based on relationship features (also called structural features) of the group (Choo et al. 2015; Ye and Akoglu 2015; Zheng et al. 2018; Zhu et al. 2019; Shehnepoor et al. 2021; Chao et al. 2022), and algorithms that combine both group behavioral and structural features (Wang et al. 2016, 2018a; Hu et al. 2019; Zhang et al. 2020, 2022a, 2022b; Akoglu et al. 2013; Liu et al. 2018). Figure 1 illustrates the timeline of some existing Spammer Group detection efforts under the two classification criteria. In the following subsections, we focus on reviewing graph-based algorithms, as it is a prominent category within the first classification criterion and these algorithms have gained significant attention in spammer group detection in the e-commerce context. The graph-based approaches to spammer group detection can be further divided based on the construction of the graphs into two categories: homogeneous graph-based algorithms (Wang et al. 2016, 2018a; Choo et al. 2015; Hu et al. 2019; Zhang et al. 2020, 2022a) and heterogeneous graph-based algorithms (Akoglu et al. 2013; Ye and Akoglu 2015; Zheng et al. 2018; Zhu et al. 2019; Shehnepoor et al. 2021; Chao et al. 2022; Zhang et al. 2022b).

Homogeneous graph-based algorithms

The basic idea of Homogeneous graph-based spammer group detection algorithms is to first construct a homogeneous co-reviewing graph according to the characteristics and similar behavior of reviewers. In a general way

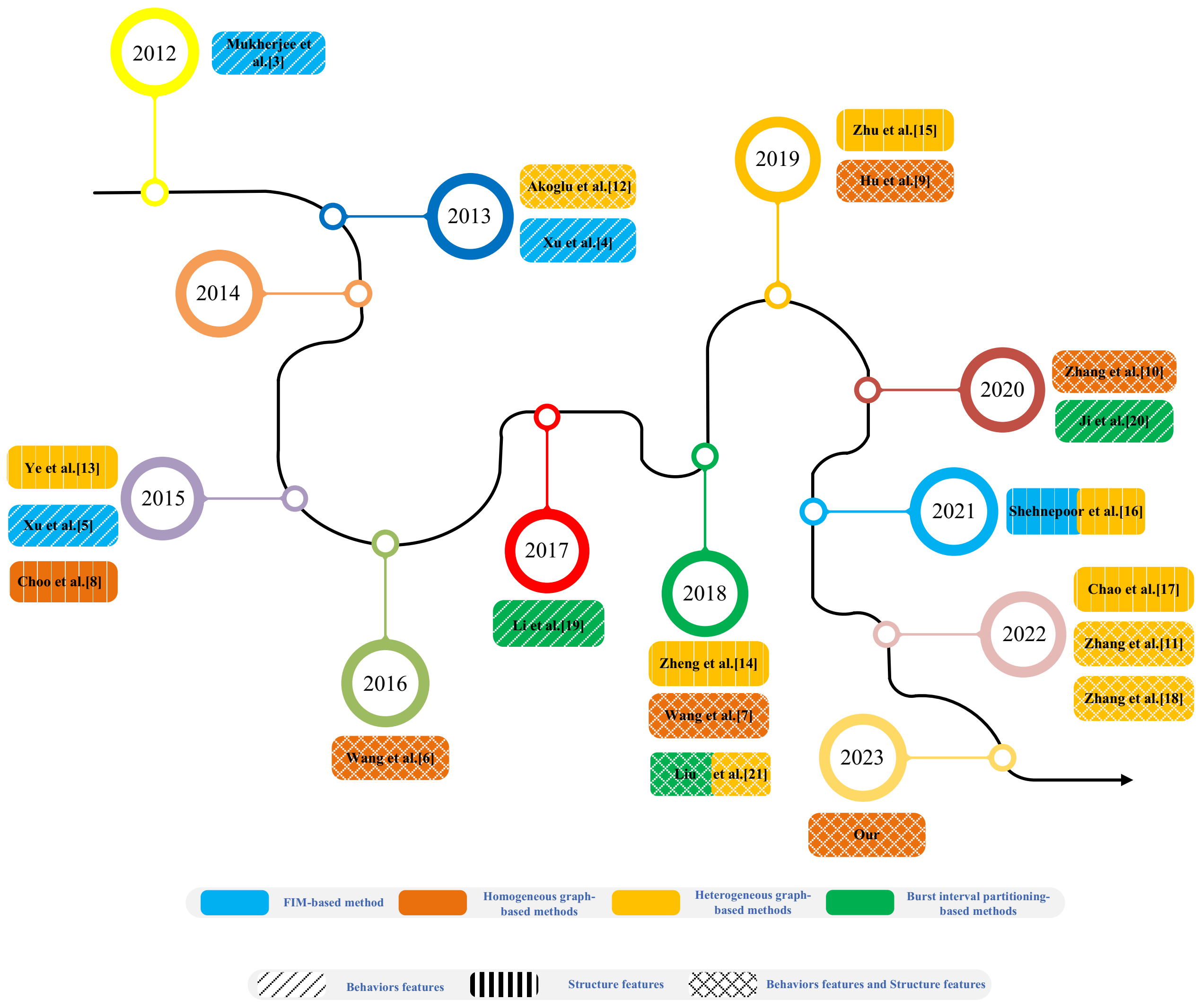


Fig. 1 Timeline of related work

in the homogeneous co-reviewing graph, nodes represent the reviewers, and the edges depict the co-reviewing relationships. For instance, within the homogeneous co-reviewing graph, edges connect reviewers who have reviewed the same product within a specific time frame.

Addressing the limitations of the FIM-based algorithm in detecting loosely coupled spammer groups, Wang et al. (Wang et al. 2016) innovated and developed a model for identifying such groups. After constructing the co-reviewing graph, they only adjusted the graph's structure based on the number of products that reviewers co-reviewed within a given time frame. They represented the reviewing data as a bipartite graph and used a divide-and-conquer strategy to produce loosely coupled spammer groups. While this algorithm could detect loosely coupled groups, it solely defined collusive behavior by the count of products co-reviewed within a specific time window, greatly simplifying the behavioral consistency metrics. To overcome the mentioned challenges, Wang et al. (2018a) introduced a new Homogeneous graph-based algorithm for spammer group detection. Their algorithm considered only the time interval between reviews and rating biases during graph construction. They adjusted the co-reviewing graph's structure using the similarity of products co-reviewed by the reviewers as a weight for the co-reviewing relationships. They introduced a top-down structure termed GGSpam and developed a spammer group identification approach named GSBC within this structure. The GSBC technique, by creating a bi-connected reviewer relationship co-reviewing graph, took into account factors such as the time interval and rating biases of co-reviewing on items. It implemented new spammer group markers, evaluating the dubiousness of potential groups from various angles. Choo et al. (2015) departed from the conventional content analysis-based spammer detection model, showcasing an unsupervised spammer group identification approach reliant on reviewer interactions and sentiment assessment. However, their algorithm did not consider adjustments to the co-reviewing graph structure. This algorithm, stemming from the viewpoint of reviewer behavior, moved away from solely emphasizing post content. Hu et al. (2019) unveiled an online two-stage framework, OGSpam, for identifying spammer groups in evolving reviewing networks. Like Wang et al. (2018a), their approach also focused on the review time interval and rating biases when constructing the co-reviewing graph. They initially utilized the Clique Percolation Method (CPM) to detect spammer groups on the original static reviewer network, then integrated updates based on networks and detections at subsequent moments. Zhang et al. (2020) began by examining reviewers' post timings and product ratings to construct a reviewer relationship co-reviewing graph. Their approach primarily adjusted the co-reviewing graph's structure using the similarity of products co-reviewed by the reviewers as the weight for the co-reviewing relationships. They then employed an enhanced label propagation algorithm to form potential groups, and finally, they used a ranking process to recognize spammer groups. Zhang et al. (2022a) redefined the detection challenge as one of distinguishing distribution variances between regular reviewer groups and spammer groups. They proposed a technique grounded in Generative Adversarial Networks (GAN), encompassing multiview sequence sampling of the dataset and leveraging the Word2Vec model for compact vector representation of reviewers. By forming neighbor reviewer relationship graphs based on vector likeness in the embedding domain and utilizing the DBSCAN algorithm, they crafted potential groups. In the end, they assessed the dubiousness of these groups through a combined loss of generator reconstruction and discriminator loss in their adversarial network model, ranking them based on suspiciousness to pinpoint spammer groups.

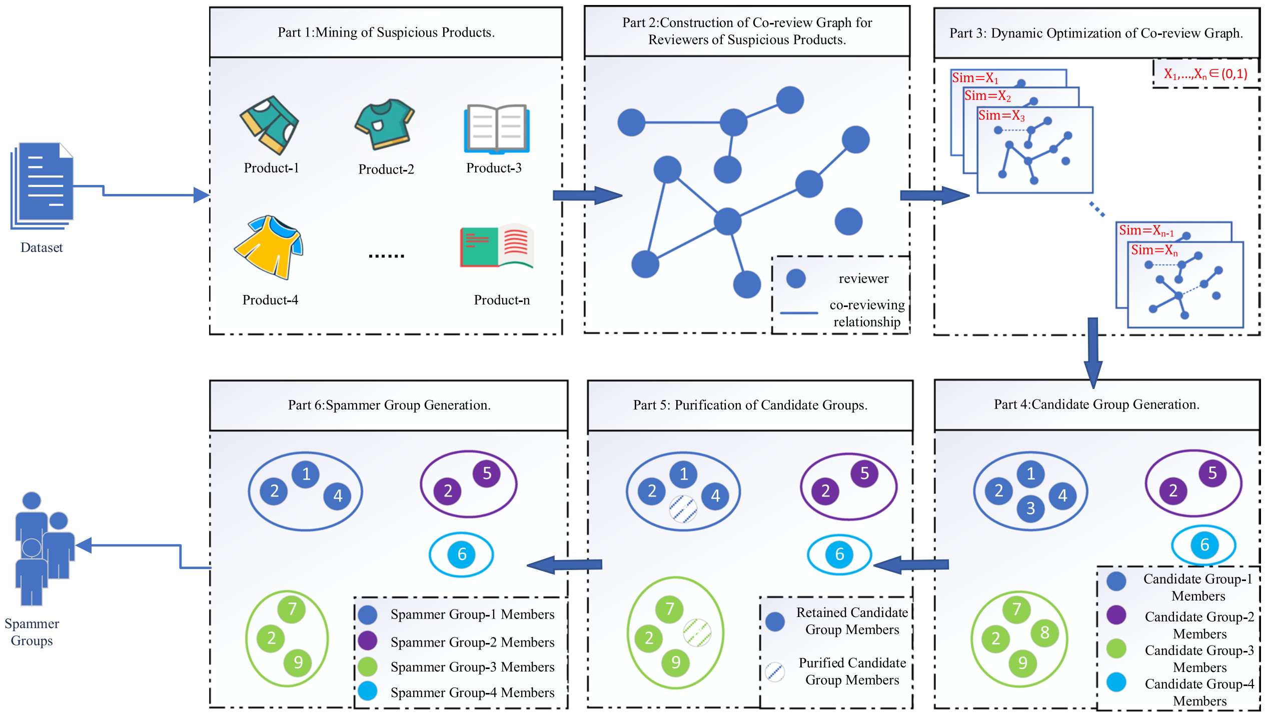
Heterogeneous graph-based algorithms

Heterogeneous graph-based approaches for spammer group detection take into account various factors, including reviewers and products. By constructing a heterogeneous network that comprises multiple types of nodes and edges, these approaches facilitate the mining of potential candidate groups.

Akoglu et al. (2013) approached the detection of spammer groups by considering highly suspicious reviewers and corresponding products as nodes and their interactive relationships as edges. By constructing heterogeneous induced subgraphs, they applied graph clustering techniques to identify spammer groups. Ye et al. (2015) introduced a novel two-step algorithm for detecting spammer groups and targeted products. They initially introduced a metric called NFS to quantify the likelihood of products being targeted by spammers, followed by the design of an algorithm named GroupStrainer. This algorithm applies agglomerative hierarchical clustering on the heterogeneous induced subgraph of suspicious products to generate spammer groups. Zheng et al. (2018) proposed an unsupervised detection algorithm called FraudNE, which employs deep network embedding to learn the latent representation of nodes. This algorithm simultaneously embeds reviewers and products into a common low-dimensional space, utilizing the DBSCAN clustering algorithm for spammer group detection. Zhu et al. (2019) introduced a heterogeneous network embedding-based algorithm for spammer group detection, which learns reviewers' feature representation through embedding explicit and implicit relationships in a bipartite network. This approach leverages a k-dimensional tree-based fast-density sub-graph mining algorithm to identify spammer groups. Shehnepoor et al. (2021) put forward a heterogeneous graph-based algorithm known as HIN-RNN. HIN-RNN adopts a structured approach, embedding the content of reviewers' reviews through SoWEs and then using an autoregressive model to encode non-local semantic dependencies between reviewers. It further employs a collaboration matrix to remove less connected "anomalous reviewers," thereby enabling precise spammer group detection. Chao et al. (2022) first constructed a heterogeneous information network using meta-graph concepts, followed by the application of an enhanced DeepWalk algorithm to learn the low-dimensional vector representation of nodes. They then utilized the Canopy and K-means clustering algorithms to generate candidate groups and finally ranked the candidates using fake indicators to obtain the spammer groups. Zhang et al. (2022b) proposed a spammer group detection algorithm called DCS-RLAA, which is based on reinforcement learning and adversarial autoencoders. They began by constructing a reviewer-product bipartite graph, then employed an enhanced reinforcement learning algorithm, Sarsa, to obtain candidate groups. Subsequently, they used the Doc2Vec model to generate embedded representations of the candidate groups, ultimately applying a single-classification model based on adversarial autoencoders to detect the spammer groups. Zhang et al. (2023) proposed a spammer group detection approach based on collaborative training. First, they constructed a Heterogeneous graph-based induced sub-network using a targeted product set. Subsequently, they employed collaborative training techniques to learn reviewer embeddings and utilized the DBSCAN clustering algorithm to produce candidate groups. Finally, a ranking approach was employed to delineate the definitive spammer groups.

Framework of the overlapped spammer group detection algorithm

In response to the shortcomings of existing detection algorithms, we introduce an overlapped spammer group detection algorithm, DRL-OSG, as outlined in Algorithm 1. Figure 2 depicts the comprehensive framework of our algorithm, which is divided into six distinct parts: (1) Mining of suspicious products; (2) Construction of co-review graph for reviewers of suspicious products; (3) Dynamic optimization of co-review graph; (4) Candidate group generation; (5) Purification of candidate groups; (6) Spammer group generation. Specifically, in the first part, we utilize NFS (Ye and Akoglu 2015) to detect suspicious products and build the suspicious product set (refer to Algorithm 2 for details). In the second part, we construct a homogeneous co-review graph for the suspicious products, with reviewers



Dataset
Part 1:Mining of Suspicious Products.
Product-1
Product-2
Product-3
Product-4
......
Product-n
Part 2:Construction of Co-review Graph for Reviewers of Suspicious Products.
reviewer
co-reviewing relationship
Part 3: Dynamic Optimization of Co-review Graph.
Sim=X₁
Sim=X₂
Sim=X₃
Sim=Xₙ₋₁
Sim=Xₙ
X₁,...,Xₙ∈(0,1)
Part 4:Candidate Group Generation.
Candidate Group-1 Members
Candidate Group-2 Members
Candidate Group-3 Members
Candidate Group-4 Members
Part 5: Purification of Candidate Groups.
Retained Candidate Group Members
Purified Candidate Group Members
Part 6:Spammer Group Generation.
Spammer Group-1 Members
Spammer Group-2 Members
Spammer Group-3 Members
Spammer Group-4 Members
Spammer Groups

Fig. 2 The overall framework of DRL-OSG

as nodes and their co-review relationships as edges (refer to Algorithm 3 for details). The third part proposes a deep reinforcement learning-based Auto-Sim algorithm, which dynamically optimizes the co-review graph structure by eliminating redundant co-review relationships through behavioral similarity among reviewers (refer to Algorithm 4 for details). In the fourth part, we apply the Ego-Splitting overlapping clustering algorithm (Epasto et al. 2017) on the optimized co-review graph, yielding candidate groups with overlapping members (refer to Algorithm 5 for details). The fifth part involves the utilization of five individual metrics (Ji et al. 2020; Zheng et al. 2012; Mukherjee et al. 2013a; Fei et al. 2013) based on fake review behavior to purify members with low suspicion from the candidate groups (refer to Algorithm 6 for details). Finally, in the sixth part, we employ five group metrics (Mukherjee et al. 2012, 2013b; Wang et al. 2016; Ji et al. 2020) centered around spammer group fraudulent features to categorize the purified candidate groups, culminating in identifying definitive spammer groups (refer to Algorithm 7 for details).

Algorithm 1 DRL-OSG ($V, \delta_{\dot{P}}, \dot{P}, \dot{R}, \dot{E}, G, \tilde{S}, OG, \delta_{\dot{I}}, \delta_{\dot{G}}, CG, PCG, SG$)

```
Input:
    $V$: Product set
    $\delta_{\overset{*}{P}}$: Suspicious product filtering threshold
    $\overset{*}{P}$: Suspicious product set
    $\overset{*}{R}$: Suspicious product reviewers set
    $\overset{*}{E}$: Co-review relationship set
    $G$: Homogeneous co-review graph
    $\tilde{S}$: Behavioral consistency parameter
    $OG$: Optimal co-review graph
    $\delta_{\overset{*}{I}}$: Individual spammer score ($ISS$) threshold
    $\delta_{\overset{*}{G}}$: Group spammer score ($GSS$) threshold
    $CG$: Candidate group set
    $PCG$: Purified candidate group set
Output:
    $SG$: Spammer group set
1. $\overset{*}{P}$ = SuspiciousProductMining($V$, $\delta_{\overset{*}{P}}$)
2. $G$ = HomogeneousCo-reviewGraphConstruction($\overset{*}{R}$, $\overset{*}{E}$)
3. $OG$ = DynamicOptimizationCo-reviewGraphs($G$, $\tilde{S}$)
4. $CG$ = CandidateGroupGeneration($OG$)
5. $PCG$ = CandidateGroupPurification($CG$, $\delta_{\overset{*}{I}}$)
6. $SG$ = SpammerGroupGeneration($PCG$, $\delta_{\overset{*}{G}}$)
```

The homogeneous co-review graph construction algorithm for reviewers of suspicious products

In the real world, members within spammer groups often engage in coordinated activities, directing public sentiment toward a specific product by reviewing it collectively. Similar to the approach taken by Zhang et al. (2023), we employ Network Footprint Scoring (NFS) (Ye and Akoglu 2015) to detect highly suspicious products from the entire product collection, as detailed in Algorithm 2. To further elucidate the interrelationships and interaction patterns among reviewers, we constructed a homogeneous co-reviewing graph for a set of suspicious products. Our process began with the creation of a bipartite graph comprising reviewers and products, followed by utilizing co-reviewing relationships to construct a homogeneous co-reviewing graph, effectively preserving the complex structural information among reviewers.

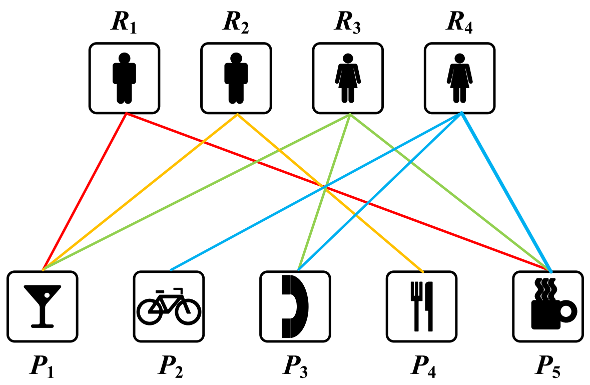


Fig. 3 The reviewer-product bipartite graph example

Definition 1 Reviewer-Product Bipartite Graph. The reviewer-product bipartite graph is defined as a network $\tilde{G} = (\tilde{V}, \tilde{E})$, where $\tilde{V}$ represents the set of nodes, comprising both reviewer nodes and product nodes. Reviewer nodes correspond to individuals who have reviewed the products, and product nodes correspond to the specific items that have been reviewed. $\tilde{E}$ represents the set of edges, encapsulating the relationships between reviewers and products. Each edge connects a reviewer node to a product node, symbolizing that the particular reviewer has reviewed on the corresponding product.

Figure 3 illustrates a sample representation of graph $\tilde{G}$, where, R_1, R_2, R_3, and R_4 denote four different reviewers, and P_1, P_2, P_3, P_4, and P_5 signify five different products. From Fig. 3, it can be observed that reviewer R_1 has reviewed on products P_1 and P_5; reviewer R_2 has reviewed products P_1 and P_4; reviewer R_3 has given feedback on products P_1,P_3, and P_5; and reviewer R_4 has expressed opinions on products P_2, P_3, and P_5. By constructing a reviewer-product bipartite graph, we can derive the co-review relationships among reviewers.

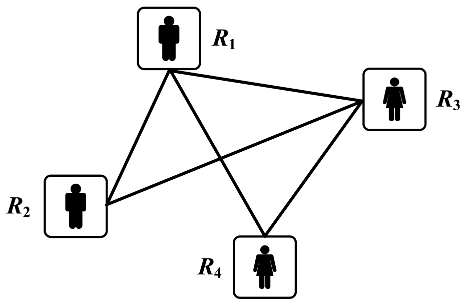


Fig. 4 The homogeneous co-review graph example

Definition 2 Homogeneous Co-review Graph. The homogeneous co-review graph is defined as a network $G = (\dot{R}, \dot{E})$,where $\dot{R}$ represents the set of nodes, consisting of reviewers who have reviewed on the target products within $\dot{P}$; $\dot{E}$ represents the set of edges, signifying the co-review relationships among reviewers obtained from the reviewer-product bipartite graph.

Figure 4 illustrates an example of a homogeneous co-review graph constructed utilizing the target product set and the set of nodes, where only reviewer nodes and co-review edges are included. For instance, in Fig. 3, reviewers R_3 and R_4 have both reviewed on product P_3. As a result, a co-review edge exists between reviewers R_3 and R_4 within the homogeneous co-review graph, indicating their shared involvement in reviewing this particular product.

The specific algorithm for constructing the homogeneous co-review graph concerning suspicious products is shown in Algorithm 3.

Algorithm 2 Suspicious Product Mining ($V, \delta_{\mathring{p}}$).

```
Input:
$V$: Product set
$\delta_{\overset{*}{p}}$: Suspicious product filtering threshold
Output:
$\overset{*}{P}$: Suspicious product set
1. $\overset{*}{P} = \emptyset$
2. for each $\overset{*}{p}$ in $V$ do
3.     if $NFS(\overset{*}{p}) \geq \delta_{\overset{*}{p}}$   # $NFS(\overset{*}{p})$ is calculated via Eq. (3)
4.         $\overset{*}{P} \cup = \overset{*}{p}$
5.     end if
6. end for
7. return $\overset{*}{P}$
```

Algorithm 3 Homogeneous Co-review GraphConstruction ($\dot{R}, \dot{E}$).

```
Input:
$\overset{*}{P}$: Suspicious product set
$\overset{*}{R}$: Suspicious product reviewers set
$\overset{*}{E}$: Co-review relationship set
Output:
$G$: Homogeneous co-review graph
1. $G = \emptyset$
2. for each pair of reviewer $R_i, R_j \in \overset{*}{R}$ do
3.     $P_i \leftarrow The\ product\ set\ reviewed\ by\ Reviewer\ i$
4.     $P_j \leftarrow The\ product\ set\ reviewed\ by\ Reviewer\ j$
5.     $P_i, P_j \in \overset{*}{P}$
6.     if $P_i \cap P_j \neq \emptyset$
7.         $\overset{*}{E} \cup = (R_i, R_j)$ # An edge representing co-reviewing is added to graph G, consisting of reviewers $R_i$ and $R_j$.
8.     end if
9. end for
10. return $G$
```

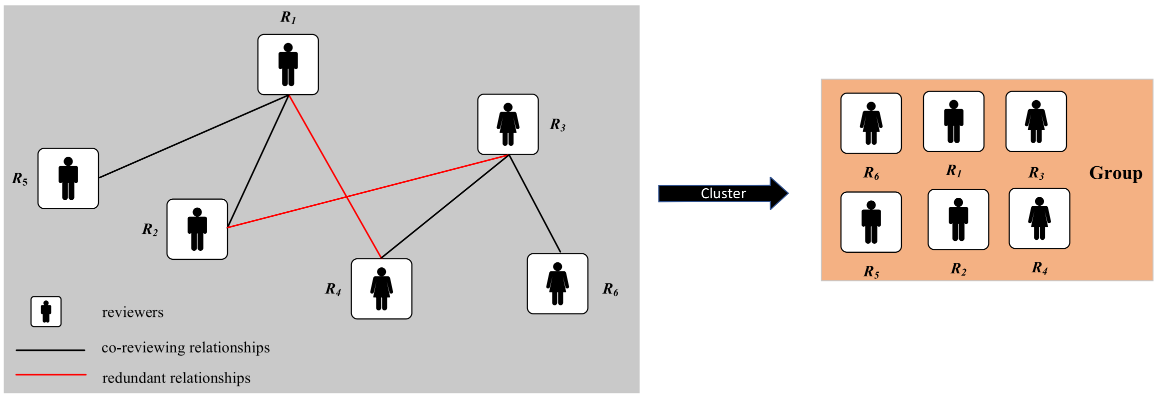
The dynamic optimization algorithm for co-review graphs

In spammer groups, there inevitably exists a common pattern of behavior among members, characterized by strong correlation. In contrast, the behaviors of genuine reviewers are independent and exhibit weaker correlations (Wang et al. 2018b). However, accidental connections may unavoidably arise between genuine reviewers and spammer members, these connections are referred to as "redundant relationships". These meaningless co-review connections can interfere with the analysis. Therefore, it is crucial to identify and eliminate these redundant relationships in order to recognize and analyze spammer groups. We introduced a deep reinforcement learning-based dynamic optimization algorithm to optimize the homogeneous co-review graph by calculating the behavioral consistency relationships between pairs of reviewers. This algorithm retains strong connections while improving the accuracy and time efficiency of the detection algorithm. Furthermore, it also autonomously searches for the optimal behavioral consistency parameter, avoiding subjective biases and enhancing detection performance and robustness.

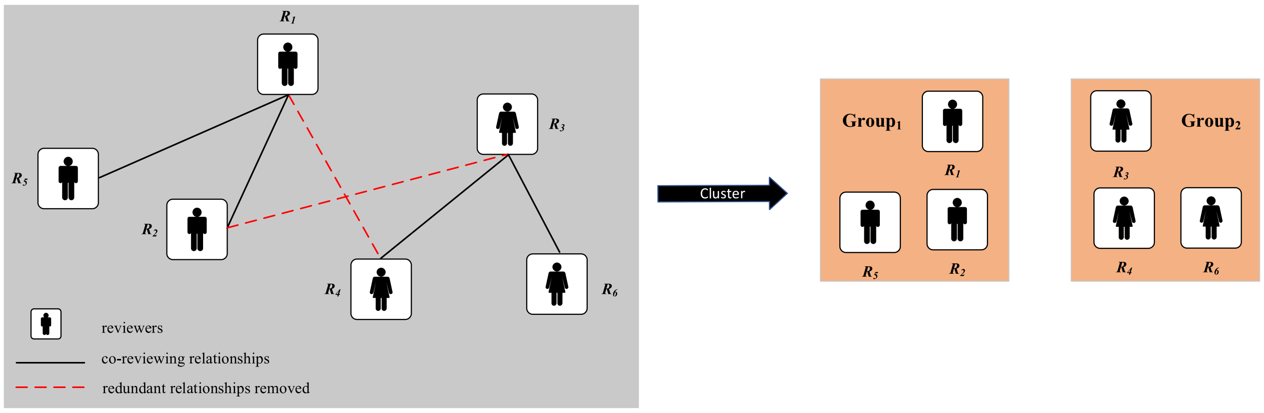
Redundant relationship

Definition 3 Redundant relationship. In the co-reviewing graph, a redundant relationship represents co-reviewing behaviors between two reviewers that arise from incidental or insignificant reasons rather than their association with a spammer group. Such relationships should not be equated with genuine co-reviewing connections, as they may adversely affect the precision of spammer group detection algorithms.

Figure 5 illustrates the implications of these redundant relationships on spammer group detection algorithms. In Fig. 5a, the co-reviewing graph retains redundant relationships. Due to the existence of redundant relationships



(a) Retain redundant relationships in co-reviewing graphs



(b) Removing redundant relationships in co-reviewing graphs

Fig. 5 Schematic representation of the impact of redundant relationships

between reviewers R_1 and R_4, as well as between R_2 and R_3, the clustering algorithm erroneously groups reviewers R_1–R_6 into a single candidate group, subsequently compromising the performance of the detection algorithm. Contrastingly, Fig. 5b showcases a co-reviewing graph with redundant relationships eliminated. With the removal of redundant connections between reviewers R_1 and R_4, and between R_2 and R_3, the clustering algorithm accurately clusters reviewers R_1, R_2, and R_5 into one candidate group, and reviewers R_3, R_4, and R_6 into another.

Behavioral consistency

Addressing the redundant relationships highlighted in Sect. "Redundant relationship", we propose a metric named "behavioral consistency." This metric assesses the extent of consistency in behavior between two co-reviewing reviewers, guiding the decision to either retain or remove their co-reviewing relationship.

Definition 4 Behavioral Consistency. In defining behavioral consistency, we incorporate two primary components: the correlation among products co-reviewed by reviewers and the correlation in ratings assigned to these co-reviewed products. Given two distinct reviewers R_i and R_j, which are nodes within a co-review graph, their behavioral consistency is shown in Eq. (1).

$$\text{Behavioral Consistency}_{ij} = \frac{\text{Product Consistency}_{ij} + \text{Rating Consistency}_{ij}}{2}$$

where Product Consistency$_{ij}$ measures the consistency of products co-reviewed by reviewers R_i and R_j, and Rating Consistency$_{ij}$ measures the consistency in the ratings assigned by reviewers R_i and R_j to these co-reviewed products, as illustrated in Eqs. (2) and (3).

$$\text{Product Consistency}_{ij} = \begin{cases} \frac{\left|\hat{P}_i \cap \hat{P}_j\right| + (n-1)}{\left|\hat{P}_i \cup \hat{P}_j\right| + n} & \left|\hat{P}_i \cap \hat{P}_j\right| = \left|\hat{P}_i \cup \hat{P}_j\right|, \\ \frac{\left|\hat{P}_i \cap \hat{P}_j\right|}{\left|\hat{P}_i \cup \hat{P}_j\right|} & \text{otherwise.} \end{cases} \tag{2}$$

where n represents the total number of products, $\hat{P}_i$ and $\hat{P}_j$ represents the sets of product nodes associated with reviewers R_i and R_j.

$$\text{Rating Consistency}_{ij} = \frac{1}{2} \cdot \left(\frac{\sum_{p=1}^{n} \left(S_{ip} - \overline{S}_i\right) \cdot \left(S_{jp} - \overline{S}_j\right)}{\sqrt{\sum_{p=1}^{n} \left(S_{ip} - \overline{S}_i\right)^2} \cdot \sqrt{\sum_{p=1}^{n} \left(S_{jp} - \overline{S}_j\right)^2}} + 1 \right) \tag{3}$$

where p represents the products co-reviewed by reviewers R_i and R_j, S_{ip} and S_{jp} denote the scores given by reviewers R_i and R_j respectively for the co-reviewed product p. Additionally, $\overline{S}_i$ and $\overline{S}_j$ elucidate the average scores rendered by reviewers R_i and R_j across all products they've co-reviewed.

To make the Product Consistency$_{ij}$ more robust, we have incorporated a smoothing term into it, $\left|\hat{P}_i \cap \hat{P}_j\right| = \left|\hat{P}_i \cup \hat{P}_j\right|$ represents the set of products that are entirely identical between $\hat{P}_i$ and $\hat{P}_j$, meaning that the products co-reviewed by reviewers R_i and R_j are exactly the same. The more products that different reviewer nodes have co-reviewed, the higher their Product Consistency$_{ij}$.

In our research, we introduced a behavioral consistency parameter threshold denoted as $\tilde{S}$. Our objective was to refine the co-reviewing graph by eliminating redundant relationships. To achieve an optimal co-reviewing graph, we excluded relationships between reviewers with a behavioral consistency less than $\tilde{S}$. Only relationships where the behavioral consistency exceeded $\tilde{S}$ were preserved, resulting in an optimized co-reviewing graph.

Deep reinforcement learning-based automatic search algorithm for behavioral consistency parameters

Behavioral consistency plays a pivotal role in DRL-OSG algorithms, offering an effective solution to the issue of meaningless co-review relationships that might otherwise interfere with or mislead detection. However, the selection of behavioral consistency parameters has a direct impact on the performance of the DRL-OSG algorithm. Achieving a delicate balance in behavioral consistency parameter $\tilde{S}$ selection is crucial: if the values are set too high, they may not only eliminate redundant relationships within the co-review graph but might also mistakenly remove relationships beneficial to the detection algorithm. Conversely, if the values are set too low, the algorithm may fail to completely remove the existing redundant relationships within the co-review graph. Traditional algorithms for parameter selection often rely on specialized knowledge and long-term accumulated experience, leading to a lack of flexibility and adaptability. Consequently, the design of an algorithm capable of automatically and effectively adjusting the values for behavioral consistency parameters, thereby substituting manual and experience-based algorithms, becomes vitally important.

In order to resolve this issue, we propose Auto-Sim, an automatic parameter search algorithm based on Deep Reinforcement Learning (DRL). In contrast to the conventional algorithms of parameter selection, the Auto-Sim algorithm casts the optimization of the behavioral consistency parameter as a Markov Decision Process and metaphorically maps the search into a maze navigation problem within the parameter space (Zhang et al. 2022c; Bom et al. 2013; Zheng et al. 2012). Employing a deep reinforcement learning *Agent* to continuously interact with the environment enables a step-by-step search from the starting position of the parameter to its endpoint within the maze. The endpoint value within the maze is then used as the final parameter value, which is then applied to downstream tasks of the algorithm, as depicted in Fig. 6. Specifically, the *Agent* considers the entire parameter space as the *environment*, the parameter search location as the *state*, and the manner of parameter adjustment as the *action*. Since the ultimate objective of our algorithm is to enhance the detection performance of spammer groups, we elect to use the precision of the detection results to formulate the *reward*. We choose the Twin Delayed Deep Deterministic strategy gradient algorithm (TD3) (Fujimoto et al. 2018) as the foundation of our algorithm. The search process for each episode $e(e=1,2,\ldots)$ is described as follows:

- State

The *state* at *i*-th step(i = 1,2,...) is represented as:

$$s^{(e)(i)} = \tilde{S}^{(e)(i)} \cup \mathcal{D}_b^{(e)(i)} \quad (4)$$

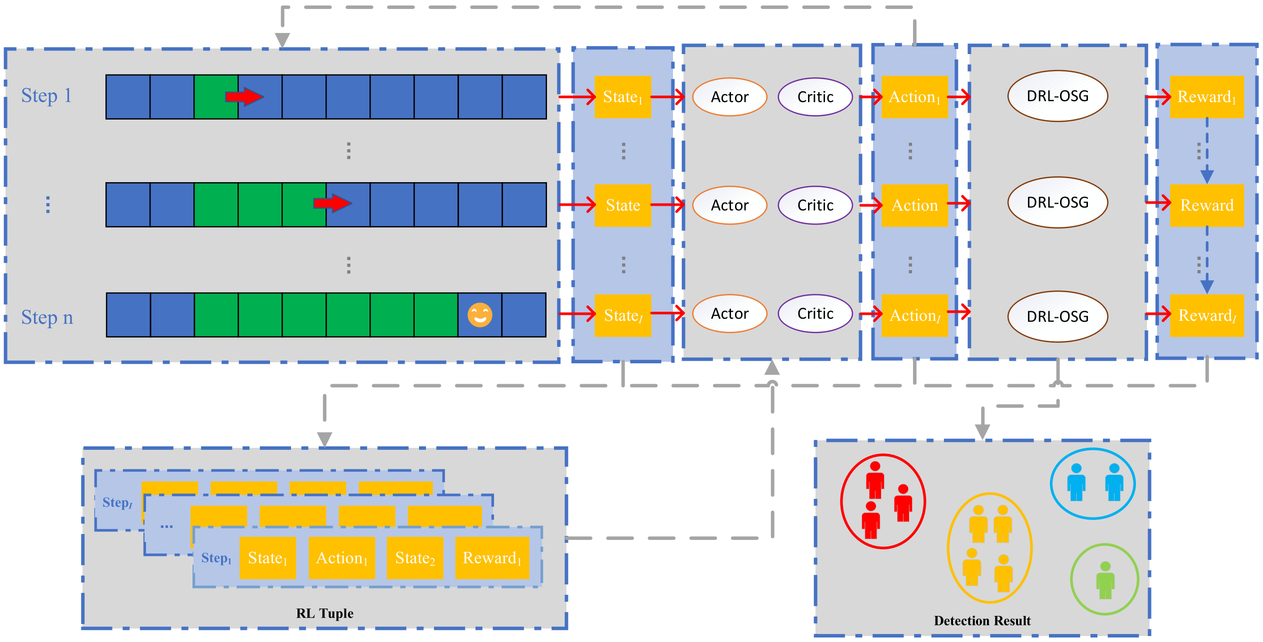


Fig. 6 Auto-Sim algorithm framework

where $\tilde{S}^{(e)(i)}$ denotes the current selection of parameters, while $\mathcal{D}_b^{(e)(i)}$ refers to the current distance settings, which include the distance of parameter $\tilde{S}^{(e)(i)}$ and its spatial boundaries $B_{\tilde{S},1}$ and $B_{\tilde{S},2}$. Specifically, $B_{\tilde{S},1}$ represents the minimum search boundary for the current parameter, and $B_{\tilde{S},2}$ signifies the maximum search boundary for the current parameter.

- Action

The *action* $a^{(e)(i)}$ at the i-th step represents the direction of the parameter search. We define the adjustable action space $\mathcal{A}$ for the parameters as $\{left, right, stop\}$.

where "left" represents moving the current parameter to the left (i.e., decreasing its value), "right" signifies moving the current parameter to the right (i.e., increasing its value), and "stop" means keeping the current parameter's position unchanged (i.e., maintaining its current value). Specifically, we have constructed an Actor (Konda and Tsitsiklis 1999) to serve as the policy network, determining the action $a^{(e)(i)}$ according to the current state $s^{(e)(i)}$:

$$a^{(e)(i)} = Actor\left(s^{(e)(i)}\right) \tag{5}$$

Additionally, the parameter transition process from the i-th step to the $i+1$-th step is illustrated as shown in Eq. (6):

$$\tilde{S}^{(e)(i)} \xrightarrow{a^{(e)(i)},\theta} \tilde{S}^{(e)(i-1)} \tag{6}$$

where $\tilde{S}^{(e)(i)}$ and $\tilde{S}^{(e)(i-1)}$ denote the values of the behavioral consistency parameter $\tilde{S}$ at the i-th and $i+1$-th steps, respectively; $a^{(e)(i)}$ represents the action chosen for $\tilde{S}$ in the action space $\mathcal{A}$ at the i-th step; and θ indicates the degree to which the action taken at the i-th step increases or decreases.

- State
- Reward

$$R\left(s^{(e)(i)}, a^{(e)(i)}\right) = \text{Precision}\left(\text{Group}\left(\tilde{S}^{(e)(i+1)}\right)\right) \tag{7}$$

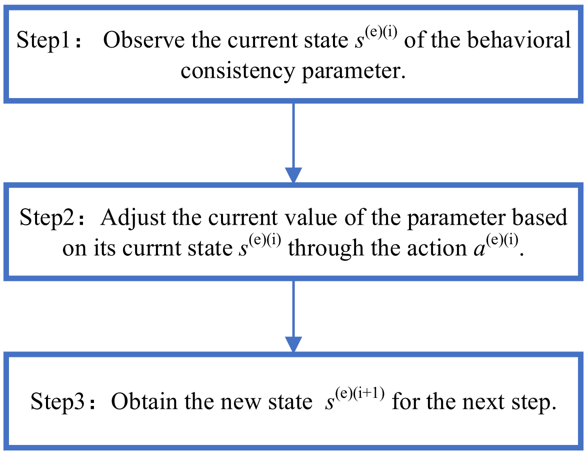

Step1 : Observe the current state $s^{(e)(i)}$ of the behavioral consistency parameter.
Step2 : Adjust the current value of the parameter based on its currnt state $s^{(e)(i)}$ through the action $a^{(e)(i)}$.
Step3 : Obtain the new state $s^{(e)(i+1)}$ for the next step.

Fig. 7 The search process for each episode

where $\tilde{S}^{(e)(i+1)}$ represents the value of the behavioral consistency parameter $\tilde{S}$ at step $i+1$-th, and Group signifies the spammer groups detected by the algorithm presented in this study.

- Termination

We define the termination conditions for a complete episode search process as follows:

$$\begin{cases} i >= I_{\max}, & \text{Timeout stop,} \\ \mathbf{a}^{(e)(i)} = \text{stop, where } i \geq 2, & \text{Active stop.} \end{cases} \tag{8}$$

where $I_{\max}$ is the maximum search step length within an episode.

Additionally, we have summarized the process of parameter searching in each episode, as illustrated in Fig. 7. Repeat the process depicted in this figure until the conclusion of each episode, where a reward $R\left(s^{(e)(i)}, a^{(e)(i)}\right)$ is obtained at every step.

In summary, we have accomplished dynamic optimization of the co-review graph by automatically adjusting the behavior consistency parameters with the Auto-Sim algorithm. This process effectively eliminates redundant relationships within the co-review graph. The specific details of this algorithm are elucidated in Algorithm 4.

Algorithm 4 Dynamic Optimization Co-review Graphs (G, $\tilde{S}$).

```
Input:
$G$ : Homogeneous co-review graph
$\widetilde{S}$ : Behavioral consistency parameter
Output:
$OG$ : Optimal co-review graph
1. $OG=\varnothing$
2. initialize paramenter space
3. for $e$=1,...,$E_{max}$ do
4.     initialize $\widetilde{S}^{(e)(0)}$
5.     for $i$=1,...,$I_{max}$ do
6.         obatin the current state $s^{(e)(i)}$ via Eq. (4)
7.         choose the action $a^{(e)(i)}$ via Eq. (5)
8.         spammer group detection using the current parameter
9.         get rewards $R(s^{(e)(i)},a^{(e)(i)})$ via Eq. (7)
10.         termination judgment via Eq.(8)
11.     end for
12.     update optimal parameter $\widetilde{S}$
13.     early stop judgment
14. end for
15. for each pair of reviewer $R_i,R_j \in \overset{*}{R}$ in $G$ do
16.     Calculate $BehavioralConsistency_{ij}$
17.     if $BehavioralConsistency_{ij} >= \widetilde{S}$
18.         $E_O \cup = (R_i,R_j)$, $E_O \to OG$ # Incorporate edges with high behavioral consistency from the G into the OG, $E_O \in \overset{*}{E}$
19.     end if
20. end for
```

The candidate group generation and purification algorithm

Upon generating the optimal co-reviewing graph through Algorithm 4, we employ the Ego-Splitting overlapping clustering algorithm (Epasto et al. 2017) to cluster the graph, thereby forming candidate spammer groups. Similar to the DBSCAN clustering algorithm used by Zhang et al. (2023), the Ego-Splitting algorithm does not require the pre-specification of the number of groups to be generated. This aligns with the premise of detecting spammer groups in the real world, and furthermore, the algorithm is capable of identifying overlapping relationships within groups. After generating candidate groups, there may still be some genuine reviewers intermingled within these groups. To enhance the precision of the detection algorithm, we utilize a candidate group purification algorithm founded on five distinct characteristics of individual spammer fake behaviors. This purification process aims to filter out the genuine reviewers within the candidate groups who do not conform to the criteria associated with spammer groups. The indicators and the calculation algorithms explained are as follows.

Definition 5 Account Duration (*AD*) (Mukherjee et al. 2013a). *AD* is a measure of the time interval between the date a reviewer posts their most recent review and the date when the same reviewer posted their earliest review on a particular website. According to the research conducted by Mukherjee et al. (Mukherjee et al. 2013a), spammers are usually not long-term users of a site, while legitimate reviewers tend to consistently use their accounts for product reviews. The calculation formula for *AD* is as follows:

$$AD(i) = 1 - \frac{t_n^i - t_e^i}{t_{data}} \tag{9}$$

where t_n^i represents the time when reviewer R_i posted their most recent review, t_e^i denotes the time when reviewer R_j posted their first review on the platform, and t_{data} signifies the time interval within the dataset. It is crucial to note that a higher AD value typically indicates a greater suspicion towards the reviewer in question.

Definition 6 Rating Deviation (RD) (Fei et al. 2013). RD is a metric that quantifies the degree of deviation between a reviewer's rating and the average rating of the product. The average product rating generally indicates the product's popularity. Legitimate reviewers' ratings should typically only show limited deviations from the product's average rating. However, spammers, with their motives of promoting or disparaging a product, are likely to give ratings significantly diverging from the product's average score. The calculation formula for RD is as follows:

$$RD(i) = \underset{p \in P_i}{\text{avg}} \frac{\left|s_{ip} - \overline{s_p}\right|}{c} \tag{10}$$

where P stands for the set of all products, P_i represents the target product set of reviewer R_i, s_{ip} signifies the score given by reviewer R_i to a specific product p ($p \in P_i, P_i \subset P_i$), and $\overline{s_p}$ indicates the average score of the product p. As e-commerce platforms typically adopt a 5-star rating system, the score deviation under this system will not exceed 4, hence we set the normalization constant c as 4. It is crucial to note that a higher RD value typically indicates a greater suspicion towards the reviewer in question.

Definition 7 Ratio of Extreme Rating (EXR) (Mukherjee et al. 2013a). Spammers typically employ extreme ratings as a tactic to either endorse or disparage products, rarely providing ratings in the 2–4 range (Mukherjee et al. 2013a). Thus, the EXR is a measure intended to quantify the fraction of extreme ratings within all the reviews for a particular product. Under the 5-star rating system, extreme ratings are identified as either 1-star or 5-star reviews issued by reviewers. The calculation formula for EXR is as follows:

$$EXR(i) = \frac{|\{s_i | s_i \in \{1, 5\}, s_i \in S_i\}|}{|S_i|} \tag{11}$$

where S_i represents the set of all ratings given by reviewer R_i, and s_i represents an element in the S_i set. It is crucial to note that a higher EXR value typically indicates a greater suspicion towards the reviewer in question.

Definition 8 The Most Reviews One-day (MRO) (Mukherjee et al. 2013a). The number of reviews that a genuine reviewer can make within a day is extremely limited, whereas spammer groups often concentrate on posting a large number of reviews for various products within a single day (Mukherjee et al. 2013a). MRO is a measure of the number of reviews a reviewer posts within a day and is normalized by the maximum number of reviews posted by any reviewer in a single day. The calculation formula for MRO is as follows:

$$MRO(i) = \frac{MaxComm(i)}{\max_{i \in R}(MaxComm(i))} \tag{12}$$

where $MaxComm(i)$ represents the maximum number of daily reviews generated by the reviewer identified as R_i, and R refers to the comprehensive collection of all reviewers. It is pertinent to mention that an increased MRO level often correlates with an augmented degree of suspicion surrounding the reviewer.

Definition 9 Review Time Interval (RTI) (Ji et al. 2020). The time interval for a genuine reviewer to post reviews on products is generally long, while spammer groups, due to efficiency concerns, typically post a large number of reviews on products in a short amount of time. RTI is a measure that reflects the length of the time interval at which a reviewer posts reviews and indicates the level of activity of a reviewer. The calculation formula for RTI is as follows:

$$RTI(i) = \frac{\sum_{x}^{m-1} f_{RTI}\left(i, t_x^i\right)}{m - 1} \tag{13}$$

$$f_{RTI}\left(i, t_x^i\right) = \begin{cases} 1, \left|t_{x+1}^i - t_x^i\right| \leq \rho, \\ 0, \left|t_{x+1}^i - t_x^i\right| > \rho \end{cases}, T^i = \left\{t_1^i, \cdots, t_x^i, t_{x+1}^i, \cdots, t_m^i\right\} \tag{14}$$

where T^i represents the review time series of reviewer R_i, t_x^i stands for the x-th element within the T^i series, and ρ encapsulates the threshold defining the permissible time interval between successive reviews. Similar to the approach of Ji et al. (Ji et al. 2020), we set the time interval threshold ρ to 28.

Definition 10 Individual Suspicious Score (*ISS*). The Spammer Individual suspicious score (*ISS*) is computed as the mean of the aforementioned five indicators, characterizing the deceptive practices of individual spammers. The calculation formula for *ISS* is as follows:

$$ISS(i) = \frac{AD(i) + RD(i) + EXR(i) + MRO(i) + RTI(i)}{5} \tag{15}$$

Additionally, we set a falsification threshold δ_I for the spammer individual falsification score *ISS*, choosing to purify the individuals in the candidate group whose *ISS* is less than δ_I, and obtain the purified candidate group collection *PCG*. Algorithm 6 describes the specific process of the algorithm.

Algorithm 5 Candidate Group Generation (*OG*).

```
Input:
$OG$: Optimal co-review graph
Output:
$CG$: Candidate group set
1. $CG = \varnothing$
2. initialize $ego\_group = \varnothing$
3. for each node $R_i$ in $OG$ do
4.     construct ego-network $EgoNet\_R_i$ of node $R_i$
5.     # $EgoNet\_R_i$ includes node $R_i$ and its neighbors
6.     for each node $R_j$ in $EgoNet\_R_i$ do
7.         initialize each node as a separate candidate group in $ego\_group$
8.     end for
9.     perform clustering in the ego-network $EgoNet\_R_i$ by maximizing modularity
10.    store the clustering result in $ego\_group$
11.    if size($ego\_group$) >= 2 then
12.        $CG = CG \cup ego\_group$
13.    end if
14. end for
15. return $CG$
```

Algorithm 6 Candidate Group Purification (CG, δ_I^*)

```
Input:
$CG$: Candidate group set
$\delta_I^*$: Individual spammer score ($ISS$) threshold
Output:
$PCG$: Purified candidate group set
1. $PCG = \varnothing$
2. for each group $g$ in $CG$ do
3.     for each member $i$ in $g$ do
4.         if $ISS(i) < \delta_I^*$ then
5.             Remove $i$
6.         end if
7.         $PCG = PCG \cup i$
8.     end for
9. end for
10. return $PCG$
```

The spammer group generation algorithm

Within the purified collection of candidate groups obtained through Algorithm 6, two distinct types of groups are identified: genuine reviewer groups and spammer groups. The goal of enhancing the precision of the detection algorithm necessitates a meticulous approach that avoids mistakenly categorizing genuine reviewer groups as spammer groups. To this end, we have employed five features indicative of spammer group fake behaviors. Based on these characteristics, we introduce a candidate group classification algorithm specifically designed to discriminate between genuine reviewer groups and spammer groups.

Definition 11 Group Rating Deviation (*GRD*). Mukherjee et al. 2012**).** Similar to the objective of individual rating bias metrics, spammer groups, when endeavoring to either promote or diminish a target product, frequently exhibit a pronounced discrepancy in their average rating compared to the product's overall average rating. The *GRD* quantifies this deviation, representing the degree to which the spammer group's rating diverges from that of the target product's mean score. The calculation formula for *GRD* is as follows:

$$RD_p(g) = \underset{i \in g}{avg} \frac{\left|s_{ip} - \overline{s_p}\right|}{c} \tag{16}$$

$$GRD(g) = \underset{p \in P_g}{\text{avg}}\, RD_p(g) \tag{17}$$

where s_{ip} represents the rating by members of group g for product p, and $\overline{s_p}$ is the average rating for product p. Similar to the individual rating bias indicator RD, we set the normalization constant c to 4. $RD_p(g)$ represents the rating deviation of group g for target product p. $GRD(g)$ represents the average rating deviation across all target products. It is crucial to note that a higher GRD value typically indicates a greater suspicion towards the group in question.

Definition 12 Group Extreme Rating Ratio (*GER*) (Ji et al. 2020). Consistent with the intent of individual extreme rating proportion metrics, the *GER* is characterized as the mean proportion of extreme ratings by members of a spammer group relative to the total ratings for a specific product. The calculation formula for *GER* is as follows:

$$GER(g) = \underset{i \in g}{\text{avg}} \frac{\left|\left\{r_i | r_i \in \{1, 5\}, r_i \in \tilde{R}_i\right\}\right|}{\left|\tilde{R}_i\right|} \tag{18}$$

where $\tilde{R}_i$ is the set of reviews from group member R_i, and r_i is an element of $\tilde{R}_i$.

Definition 13 Group Review Tightness (*GRT*) (Wang et al. 2016). Wang et al. (2016) first considered the group review tightness metric, *GRT* measures the closeness with which group members collaborate in crafting fake reviews. The calculation formula for *GRT* is as follows:

$$GRT(g) = \frac{\left|V_g\right|}{\left|R_g\right| \cdot \left|P_g\right|} \tag{19}$$

where V_g represents the collection of reviews posted by the members of the group g for the target product. R_g denotes the set of members in group g, and P_g stands for the set of target products reviewed by group g.

Definition 14 Group One Day Reviews (*GOR*) (Mukherjee et al. 2013b). By analyzing the number of reviews posted by a spammer group within a single day, one can assess the suspicion level of that group. If members of the group consistently post a significant number of reviews in a 24-h period, it renders the group increasingly suspicious. Based on the research by Mukherjee et al. (2013b), spammers tend to submit at least 6 reviews daily, in contrast to genuine reviewers who typically post only 1–2 reviews. The GOR calculates the days on which each group member posted more than 5 reviews and then averages this number across all group members. The calculation formula for *GOR* is as follows:

$$f_{GOR}\left(i,t^i\right) = \begin{cases} 1, & \text{if CountRev}\left(t^i\right) > 5 \\ 0, & \text{otherwise} \end{cases}, t^i \in T^i \tag{20}$$

$$GOR(g) = \underset{i\in g}{\text{avg}} \frac{\sum_{t^i\in T^i} f_{GOR}\left(t^i\right)}{\left|T^i\right|} \tag{21}$$

where T^i represents the comprehensive set of review dates for member R_i from a spammer group, t^i represents a specific review date within this set, and $\text{CountRev}\left(t^i\right)$ represents the number of reviews disseminated by member R_i on the date t^i.

Definition 15 Group Size (*GS*) (Wang et al. 2016). The GS metric represents the size of a spammer group. There exists a positive correlation between the size of the group and its level of suspiciousness and potential harm; as the group size increases, its potential harm intensifies. The calculation formula for *GS* is as follows:

$$GS(g) = \frac{1}{1 + e^{-(|R_g|-3)}\Big|} \tag{22}$$

where R_g represents the member set of the spammer group g. It is crucial to note that a higher *GS* value typically indicates a greater suspicion towards the group in question.

Definition 16 Group Suspicious Score (*GSS*). The Spammer Group Score (*GSS*) is calculated as the mean of the five aforementioned indicators, which characterize the deceptive practices of spammer groups. The calculation formula for *GSS* is as follows:

$$GSS(g) = \frac{GRD(g) + GER(g) + GRT(g) + GOR(g) + GS(g)}{5} \tag{23}$$

Furthermore, we establish a falsification threshold $\delta_{\dot{G}}$ for the *GSS*. This threshold is utilized to categorize the purified candidate groups (*PCG*). We retain groups in the candidate group collection with more than two members and the *GSS* exceeding $\delta_{\dot{G}}$. The final spammer groups (*SG*) is then obtained, encapsulating the resulting groups. The specific methodology is delineated in Algorithm 7.

Algorithm 7 Spammer Group Generation ($PCG, \delta_{\dot{G}}$).

```
Input:
$PCG$: Purified candidate group set
$\delta_{\dot{G}}^{*}$: Group spammer score ($GSS$) threshold
Output:
$SG$: Spammer group set
1. $SG=\varnothing$
2. for each group $g$ in $PCG$ do
3.    if size($g$)>2 then
4.       if $GSS(g) > \delta_{\dot{G}}^{*}$ then
5.          $SG = SG \cup g$
6.       end if
7.    end if
8. end for
9. return $SG$
```

Experiment and result analysis

Dataset and final group label getting algorithm

Dataset

Consistent with several published studies (Ji et al. 2020; Zhang et al. 2023), our experiments also employ the AmazonBooks real review dataset, which comes without genuine labels. This dataset encompasses AmazonBooks reviews from 1993 to 2014, amassing an impressive 22,507,155 reviews, given by 8,026,324 reviewers, for 2,330,066 products. Considering the enormity of the dataset, we elected to follow the methodology suggested by GSDB (Ji et al. 2020), where we restrict our examination solely to the reviews from 2013. This subset is comprised of 6,990,316 reviews penned by 2,998,380 reviewers, covering 1,079,741 different products. Detailed specifications of this dataset are illustrated in Table 1:

Table 1 Statistics of the dataset

Dataset	The whole AmazonBooks data	data in 2013
Reviews	22,507,155	6,990,316
Reviewers	8,026,324	2,998,380
Products	2,330,066	1,079,741

Final group label algorithm

Given the current lack of publicly available, genuinely labeled datasets for spammer group detection, we require group labels to evaluate the detection performance of our DRL-OSG algorithm. Therefore, we engaged three graduate experts in the field of e-commerce to manually annotate the top-300 spammer groups detected by all algorithms, and these annotations were used as ground truth.

We employ the five metrics for individual spammers, as defined in Sect. "The candidate group generation and purification algorithm", to label the spammer groups output by Algorithm 7. Initially, we assessed and scored each group member based on individual metrics tailored to gauge their level of deceptive activities: 1 point for identification as a spammer, 0.5 points for ambiguous cases, and 0 points for those judged as non-spammers. Subsequently, to acquire an overall understanding of the spamming behavior within the group, we calculate the total score for each group. The total score is represented by the sum of the individual scores of all members within the group. Lastly, we compute the average group member score by dividing the total group score by the number of group members. If the average group member score is greater than or equal to a threshold δ_g, we label that group as a spammer group.

It's imperative to note that the choice of the threshold δ_g is crucial. If δ_g is set too high, while it ensures that there are no innocent reviewers in the spammer groups, it might overlook some groups mainly composed of spammers, thereby increasing the rate of missed detection. Conversely, setting δ_g too low might lead to the misclassification of groups, where the majority are innocent reviewers, as spammer groups. Therefore, similar to the SGDCTH algorithm (Zhang et al. 2023), we have determined a δ_g value of 2/3 to efficiently balance the accuracy and scope of spammer identification.

Baseline, evaluation criteria, and experimental settings

Baselines

In order to validate the efficacy of our approach, we selected five outstanding comparative algorithms from the field of spammer group detection as baseline algorithms. The primary concepts of these algorithms are introduced as follows.

(1) GSDB (Ji et al. 2020): A classic algorithm for spammer group detection employs the burst-based algorithm, approaching the detection from the perspective of the product. Analogous to the methodology presented in this paper, it utilizes both individual spammer falsification metrics and group spammer falsification metrics for the purification and classification of candidate groups.
(2) GSBC (Wang et al. 2018a): A classic algorithm for spammer group detection based on the homogeneous graph. This algorithm solely considers the product similarity based on co-reviewing by reviewers to adjust the weight of the graph structure. Additionally, it utilizes a bi-connected reviewer relationship graph to detect spammer groups.
(3) GroupStrainer (Ye and Akoglu 2015): A classic algorithm for spammer group detection based on the heterogeneous graph. This approach constructs induced subgraphs and employs agglomerative hierarchical clustering techniques to generate spammer groups. Analogous to the approach presented in this paper, this algorithm also utilizes the NFS metric to filter suspicious products.
(4) HoloScope (Liu et al. 2018): A classic algorithm for spammer group detection based on the burst. This approach introduces an innovative "contrast suspicion" metric that dynamically accentuates the contrasting behavior between fraudsters and genuine users. By integrating graph topology, temporal bursts, and declines, as well as rating discrepancies, it systematically identifies fraudulent activities.
(5) SGDCTH (Zhang et al. 2023): A classic algorithm for spammer group detection based on the heterogeneous graph. This approach employs a collaborative training algorithm to learn the feature representations of nodes and then applies the DBSCAN clustering algorithm to generate candidate spammer groups within the vector space of reviewers. Analogous to the approach presented in this paper, this algorithm also utilizes the NFS metric to filter suspicious products.

Evaluation criteria

In the experiments, we utilized three standard metrics prevalent in the spammer group detection field (Wang et al. 2016, 2018a; Ji et al. 2020; Zhang et al. 2023) to evaluate the performance of our detection algorithm. These metrics are Precision, Recall, and the F1 score. The relevant formulas are provided as follows:

Table 2 Paramenter setting

Categories	Algorithm	Parameters	Physical interpretations	Value
Homogeneous graph-based method	DRL-OSG (Ours)	$\delta_{\dot{p}}$	Suspicious product filtering threshold	0.6
		$\tilde{S}$	Behavioral consistency parameter threshold	0.75
		$\delta_{\dot{I}}$	Individual spammer score (*ISS*) threshold	0.5
		$\delta_{\dot{G}}$	Group spammer score (*GSS*) threshold	0.5
	GSBC (Wang et al. 2018a)	T	Co-review time window size	30
		δ	Edge weight threshold	0.1
		MP	User-specified parameter	1000
		$MINSPAM$	Minimum spam threshold for a group	0.49
Review burst-based method	GSDB (Ji et al. 2020)	δ_{TP}	Target product filtering threshold	0.1
		δ_I	Individual spammer score (*ISS*) threshold	0.43
		δ_G	Group spammer score (*GSS*) threshold	0.54
Heterogeneous graph-based method	HoloScope (Ye and Akoglu 2015)	b	Scaling base	32
	GroupStrainer (Liu et al. 2018)	η	Degree threshold	20
		S_L	similarity lower bound	0.5
		$\delta_{\tilde{p}}$	Suspicious product filtering threshold	0.65
	SGDCTH (Zhang et al. 2023)	$\in$	Neighborhood radius threshold	0.6
		ϕ	Minimum number of sample points threshold	2

$$\text{Precision} = \frac{TP}{TP + FP} \tag{24}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{25}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{26}$$

In these formulas, True Positive (*TP*) represents the count of actual positive samples correctly identified by the algorithm, False Positive (*FP*) stands for the number of negative samples mistakenly identified as positive by the algorithm, and False Negative (*FN*) signifies the number of actual positive samples erroneously classified as negative by the algorithm.

These three fundamental metrics assist in a comprehensive evaluation of algorithm performance from various perspectives. Precision indicates the correctness of positive samples predicted by the algorithm, with a higher precision implying fewer mistakes within those predictions. Recall demonstrates the algorithm's capability to identify actual positive samples; a higher recall suggests that the algorithm is capable of identifying more true positive samples. The F1 score is the harmonic mean of precision and recall. It synthesizes both precision and recall, playing a critical role in balancing the demands for accuracy and comprehensiveness.

Experimental settings

Based on the 2013 Amazon books review dataset, we designed five sets of experiments to comprehensively evaluate our proposed algorithm. The first set of experiments was aimed at understanding the influence of parameter selection on our spammer group detection approach. We experimented with various parameter values, and through a meticulous analysis of the results, we unveiled the impact of these parameters on the overall performance of our algorithm. In our second set of experiments, the primary objective was to benchmark the efficacy of our algorithm by contrasting it with several baseline algorithms in the spammer group detection domain. By analyzing and comparing the performance of these algorithms across the Precision, Recall, and F1 score metrics, we assessed the accuracy and comprehensiveness of our approach. In the third set of experiments, which focused on the quality analysis of generated groups, we assessed the quality of groups produced by our algorithm from three perspectives: group size, the clustering coefficient of group members, and the overlap ratio of the groups. In the fourth set of experiments, we conducted a comparative analysis of the time complexity between our algorithm and the baseline algorithms. In the fifth set of experiments, we introduced three variants of the DRL-OSG algorithm to validate the necessity of considering all available information. We conducted thorough experiments on these variants and delved into a comprehensive analysis of their outcomes. The aim was

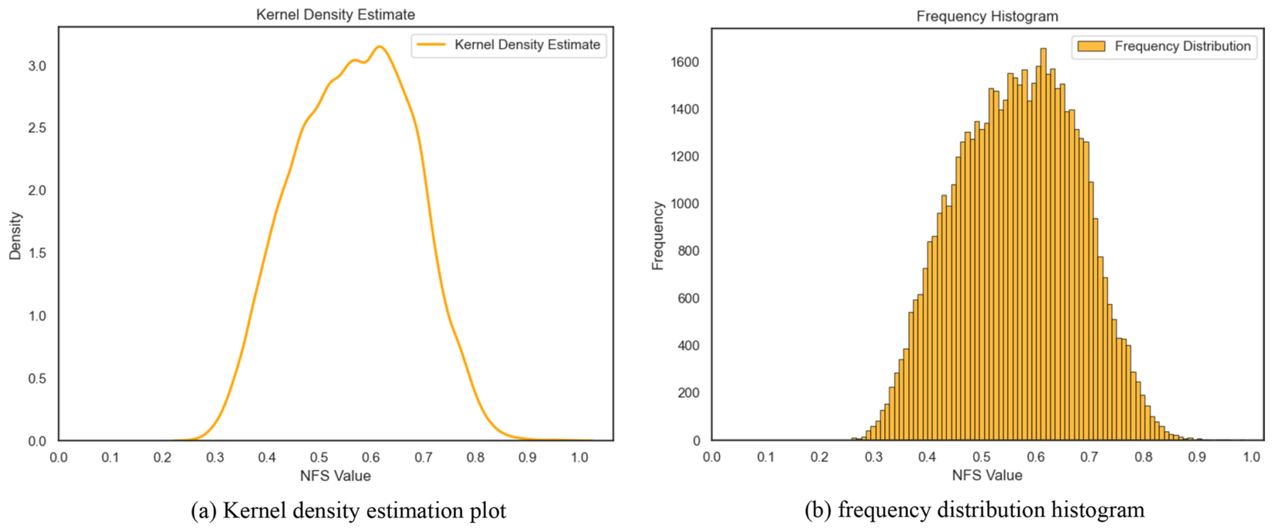


Fig. 8 Kernel density estimation plot and frequency distribution histogram for NFS values

to understand their performance under various conditions, as well as to discern their potential advantages and limitations.

In the DRL-OSG algorithm presented in this paper, four parameters are involved: the suspicious product filtering threshold $\delta_{\dot{p}}$, the individual spammer score threshold $\delta_{\dot{I}}$, the group spammer score threshold $\delta_{\dot{G}}$, and the behavioral consistency parameter $\tilde{S}$. The thresholds for individual spammer score $\delta_{\dot{I}}$ and group spammer score $\delta_{\dot{G}}$ are benchmarked against the GSDB algorithm (Ji et al. 2020), the details of which are provided in Table 2. We set the individual spammer score threshold $\delta_{\dot{I}}$ higher than that of the GSDB algorithm to further refine the groupings. Conversely, the group spammer score threshold $\delta_{\dot{G}}$ is configured to be lower than GSDB's standard, expanding the scope of potentially suspicious groups. Table 2 comprehensively lists the physical interpretations and set values of the parameters for both our algorithm and the baseline approach.

Results and analysis of parameter selection

Based on the parameter configurations outlined in Table 2, we conducted the first set of experiments focused on parameter selection. In this study's DRL-OSG algorithm, there are two parameters that require validation: the suspicious product filtering threshold $\delta_{\dot{p}}$ and the behavioral consistency parameter threshold $\tilde{S}$.

Results and analysis of the suspicious product filtration threshold

To further investigate the impact of the suspicious product selection threshold $\delta_{\dot{p}}$, we visualized the NFS values of products from the Amazonbooks review dataset using kernel density plots and frequency distribution histograms. As illustrated in Fig. 8, the distribution of the NFS values for products displays significant variability, with a prominent peak around 0.6. The majority of NFS values for products primarily fall within the 0.5–0.7 range. Given the vast number of reviews in the Amazon dataset, the selection of the suspicious product selection threshold $\delta_{\dot{p}}$ is crucial. Setting a threshold that's too low would increase the computational time of the detection algorithm, while a threshold that's too high might result in the loss of substantial product information, leading to a decrease in the performance of the detection algorithm due to the omission of co-review relationships between reviewers. Therefore, after balancing time efficiency and algorithmic performance, we chose the peak value of 0.6 from both the NFS kernel density plot and the NFS frequency distribution histogram as the suspicious product selection threshold $\delta_{\dot{p}}$. This choice enabled us to retain the majority of suspicious products, while also moderately reducing the algorithm's computational time, thus ensuring the efficacy of the DRL-OSG algorithm.

Conclusion 1 Following the aforementioned analysis, we set the threshold for target product selection, denoted as $\delta_{\dot{p}}$, at 0.6. This led to a final set of 23,318 products for subsequent experimental evaluation.

Results and analysis of the behavioral consistency parameter threshold

To optimize the value for the behavioral consistency parameter $\tilde{S}$, we applied the deep reinforcement learning-based Auto-Sim algorithm for automatic adjustments. Considering that the baseline approach in spammer group

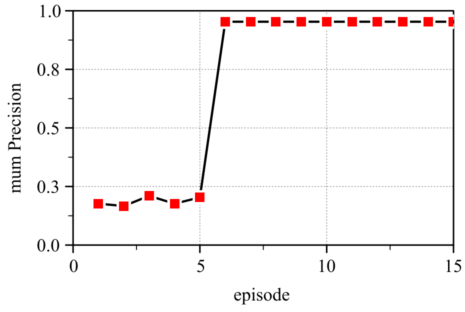


Fig. 9 The process of parameter searching

detection relies on assessing the performance against the top-300 detected spammer groups, we opted to evaluate the precision of our algorithm's top-300 groups as the reward function for the Auto-Sim experiments. Figure 9 charts the evolution of the top-300 groups' Precision across increasing episodes. Our results reveal that the precision had some fluctuations and mild increases in the initial episodes, with values ranging from 0.1667 to 0.21. It is evident that until the 5th episode, the precision saw minor oscillations suggesting our agent was still refining its experience and searching for a potential optimal parameter. However, from the 6th episode onwards, a significant surge in precision was observed, with the $\tilde{S}$ value settling at 0.9533. This remarkable stability in precision for the subsequent episodes implies that our agent successfully pinpointed an optimal value for the behavioral consistency parameter $\tilde{S}$.

Conclusion 2 Through the application of the Auto-Sim algorithm, we optimized the behavioral consistency parameter $\tilde{S}$. As a result, the precision of the DRL-OSG algorithm stabilized at 0.9533, indicating the successful identification of the optimal behavioral consistency parameter $\tilde{S}$.

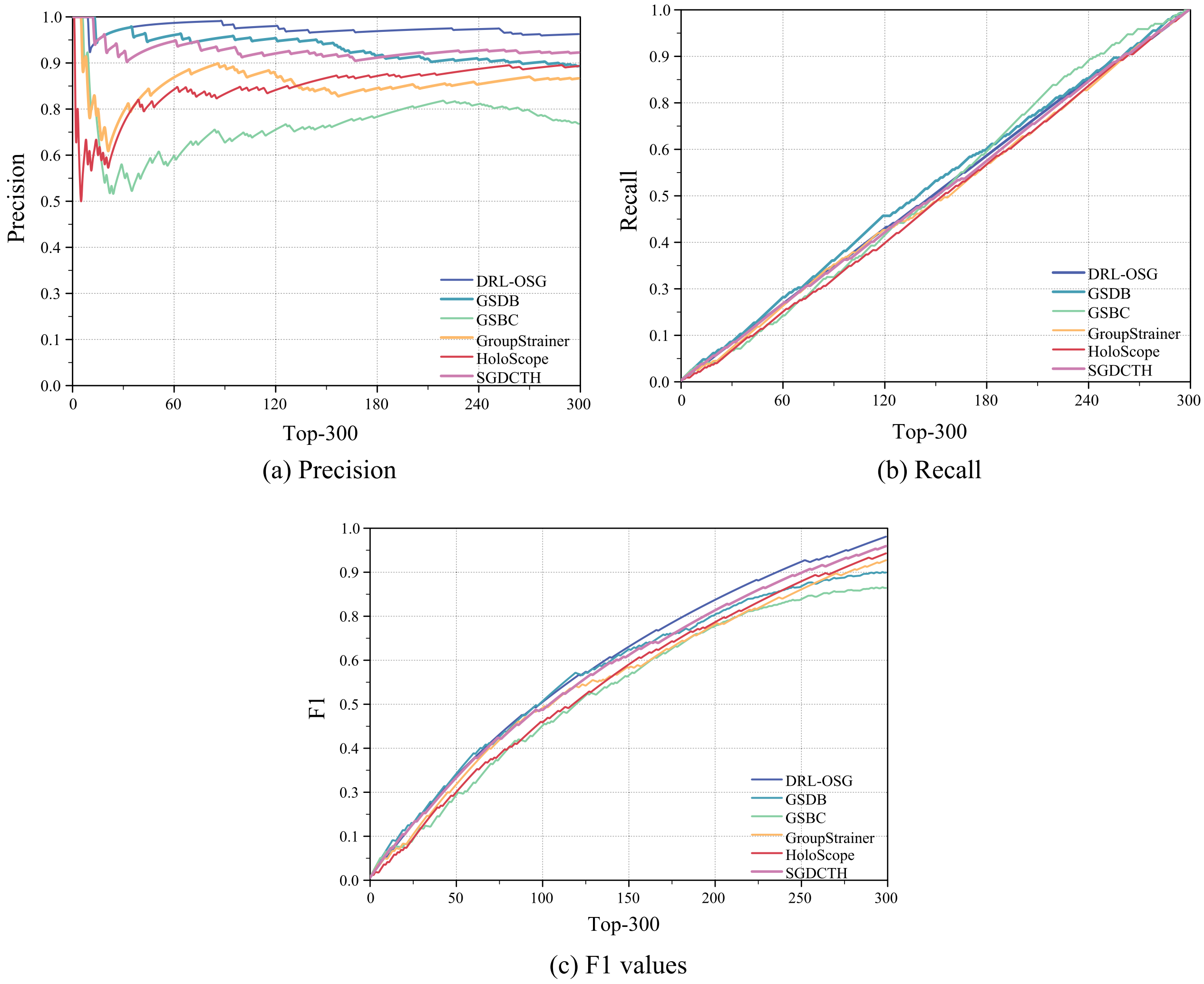


Fig. 10 The precision, recall, and F1 values of the top-300 groups for DRL-OSG with baseline algorithms

Results and comparison analysis for the group detection algorithm

Figure 10a demonstrates that among the detection results for the top-300 groups generated by each algorithm, the precision curve for our DRL-OSG algorithm consistently outperforms both the GroupStrainer and HoloScope algorithms. Moreover, it outperforms the SGDCTH algorithm after approximately the 19th group and the GSDB algorithm after approximately the 36th group. These results suggest that constructing a homogeneous co-review graph can significantly mine implicit relationships among reviewers to detect spammer groups. Simultaneously, our DRL-OSG algorithm consistently outperforms the GSBC algorithm, which similarly employs a homogeneous co-reviewing graph for spammer group detection. However, the GSBC algorithm only considers product similarity based on co-reviewing by reviewers to adjust the weight of the graph structure, neglecting other redundant relationships present in the co-reviewing graph. This highlights the importance of eliminating such redundancies and underscores the superiority of our proposed dynamic optimization for the co-reviewing graph. The initial precision curve for our DRL-OSG algorithm mirrors that of the GSBD and SGDCTH algorithm, exhibiting a slight decline but not a significant one. In contrast, the precision curves for the GSBC, GroupStrainer, and HoloScope algorithms decline sharply at first and then gradually rise. This indicates that the precision performance of our DRL-OSG algorithm is less dependent on sample size.

Figure 10b presents the recall results for each algorithm. The recall curve for our DRL-OSG algorithm is roughly on par with the GroupStrainer, HoloScope, and SGDCTH algorithms but slightly lower than the GSBC and GSDB algorithms. This discrepancy might be due to our algorithm overlooking some spammer groups that are adept at concealing their activity. Notably, compared to other baseline algorithms, the recall curve for our DRL-OSG algorithm is smoother.

Figure 10c illustrates the F1 scores derived from precision and recall, where our DRL-OSG algorithm's performance initially parallels that of the GSDB and SGDCTH algorithms, surpassing both the GSBC and GroupStrainer, as well as the HoloScope methods. However, subsequent to the 74th and approximately the 140th groups, the DRL-OSG algorithm exceeds the performance of SGDCTH and GSDB, respectively. Notably, between the 140th and 300th groups, the DRL-OSG algorithm consistently outperforms the GSDB, GSBC, GroupStrainer, HoloScope, and SGDCTH algorithms. This enhancement is likely due to the emergence of spammer groups with overlapping members around the 140th group, underscoring the superiority of the DRL-OSG algorithm in detecting such overlapping groups.

Conclusion 3 In summary, DRL-OSG algorithm exhibits a precision curve in the detection of the top-300 spammer groups that is significantly superior to baseline algorithms. The recall curve is smoother compared to other algorithms, and the F1 score curve is also superior to that of the baseline algorithms.

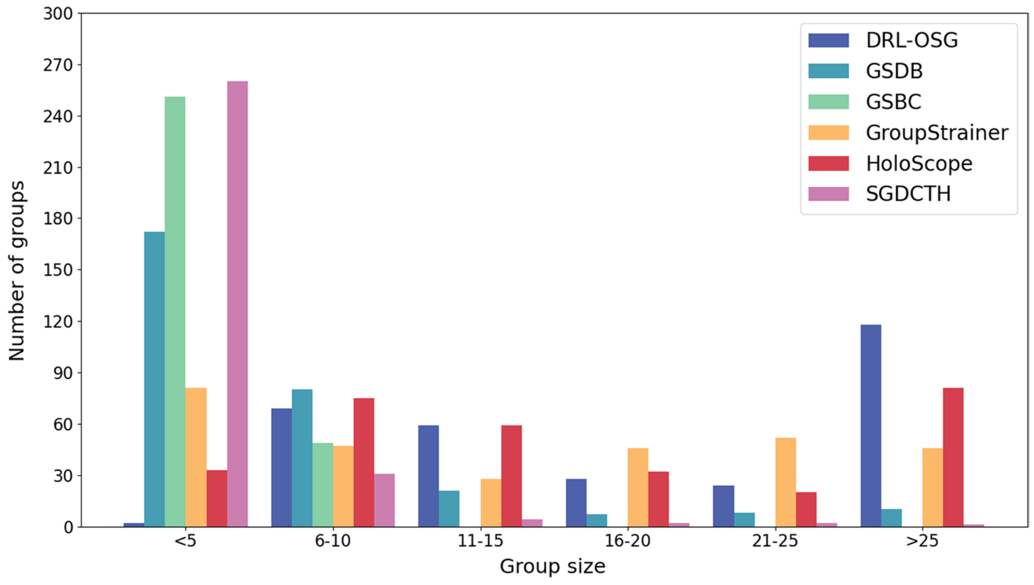


Fig. 11 Size distribution of groups

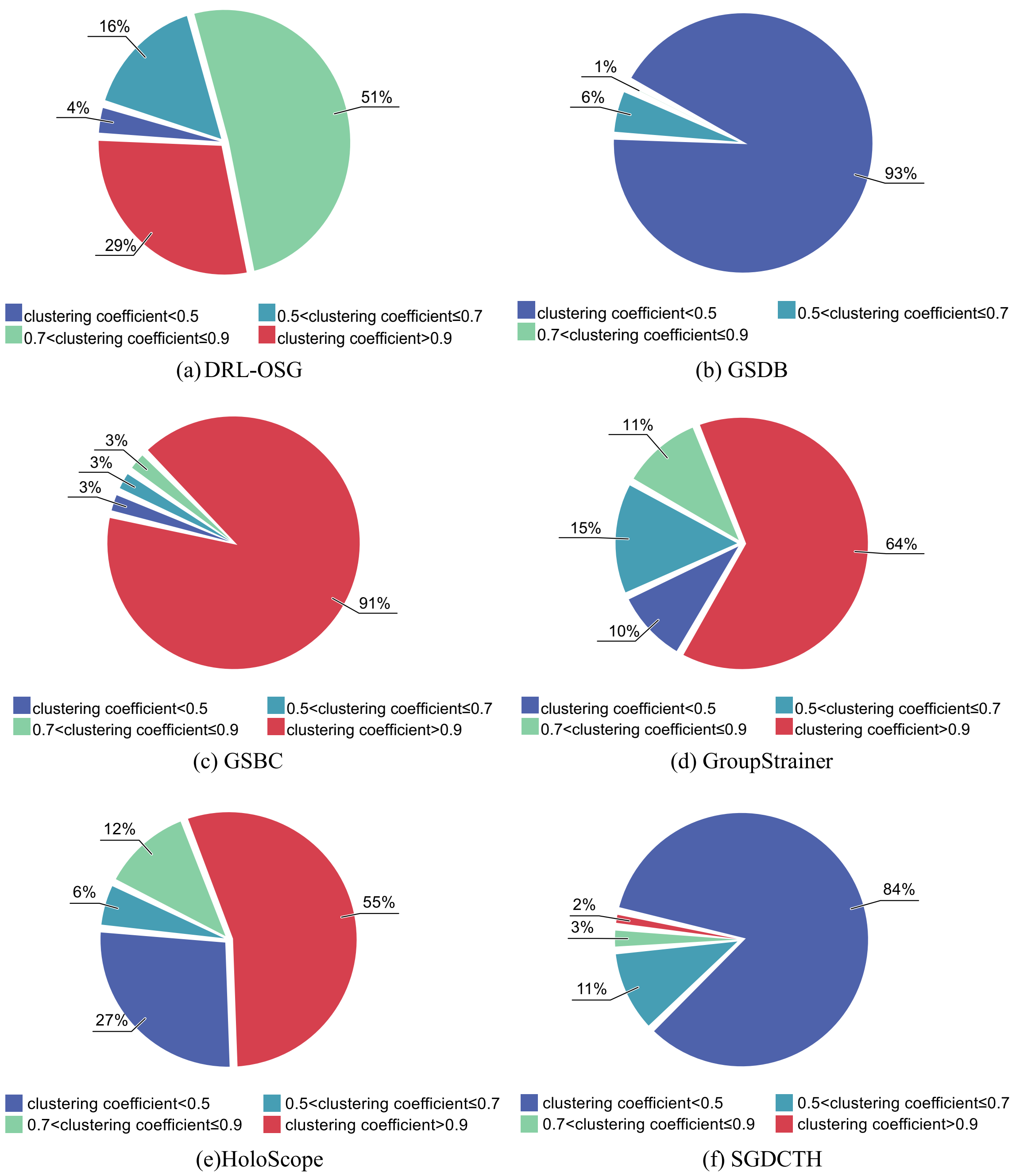


Fig. 12 Clustering coefficient of groups

Results and analysis of group quality

In this set of experiments, we assessed the quality of groups generated by the DRL-OSG algorithm through statistical feature analysis. Figures 11 and 12 presents present the size distribution and the clustering coefficient distribution of the top-300 spammer groups generated by the DRL-OSG algorithm and the baseline algorithms on the Amazon Books review dataset. Tables 3 and 4 illustrate the distribution of overlapping members in the top-300 and top-1000 spammer groups identified by the DRL-OSG algorithm.

Figure 11 presents the detection of larger group sizes by the DRL-OSG, GroupStrainer, and HoloScope algorithms, with the DRL-OSG algorithm particularly identifying spammer groups larger than 25 members, which comprise about 40% of all detected groups. This is due to

Table 3 Distribution of overlapping members in the top-300 groups

Reviewer overlaps 1 time	Reviewer overlaps 2 time
15,920	73

Table 4 Distribution of overlapping members in the top-1000 groups

Reviewer overlaps 1 time	Reviewer overlaps 2 time
30,548	1166

the DRL-OSG algorithm's capability for detecting overlapping members within groups. Larger spammer groups, armed with abundant resources and complex structures, can more rapidly spread malicious reviews and exert a greater impact on consumer purchasing decisions, thus posing a relatively higher threat (Wang et al. 2018a). Conversely, the GSDB, GSBC, and SGDCTH algorithms demonstrate a proficiency in detecting smaller groups, typically with sizes not exceeding five members. While this is beneficial for identifying small-scale coordinated activities, these algorithms may display limitations in addressing large-scale spammer campaigns.

Figure 12 presents a comparison of the clustering coefficient distributions between the DRL-OSG algorithm and baseline algorithms. The GSBC algorithm, employing a min-cut approach to partition the biconnected co-reviewing graph, resulted in a collection of candidate groups that were small in size distribution but highly interconnected internally. Consequently, the GSBC algorithm exhibited the highest proportion of higher-order clustering coefficients (greater than 0.7), reaching 94%. The DRL-OSG algorithm's proportion of higher-order clustering coefficients was approximately 80%, which is higher than that of the GSDB, GroupStrainer, HoloScope, and SGDCTH algorithms. The high-quality and closely interconnected groups produced by the DRL-OSG algorithm can be attributed to its effective graph optimization. This optimization process eradicates redundant and misleading ties among reviewers, accentuating genuine and closely bound relationships among spammers.

We observed that previous algorithms, such as GSDB, GSBC, GroupStrainer, HoloScope, and SGDCTH did not account for the overlapping of members within spammer groups. In reality, spammer groups with overlapping members tend to be more active and potentially more harmful. Tables 3 and 4 provide the overlapping percentages for the top-300 and top-1000 spammer groups, respectively. These figures underscore that the groups identified by the DRL-OSG algorithm exhibit significant overlap. Notably, certain suspicious reviewers are active across multiple distinct groups, suggesting they might be the leaders or core members of these spammer groups. Identifying overlapping members within these groups can shed light on the intricate structure and operational modus operandi of spammer organizations. Consequently, the DRL-OSG algorithm proves to be of significant value in practical applications.

Conclusion 4 The DRL-OSG algorithm outperforms other baseline algorithms in detecting large and highly interconnected spammer groups and effectively identifies overlapping members within these groups, uncovering the complex structures and operational methods of spammer groups.

Results and analysis of time complexity

To analyze the operational efficiency of various algorithms, we conducted a comparative analysis of the time complexity between the DRL-OSG algorithm and baseline algorithms. The results are presented in Table 5.

Table 5 details the time complexity comparison for each stage of the DRL-OSG algorithm and baseline algorithms. The DRL-OSG, GSDB, GroupStrainer, HoloScope, and SGDCTH algorithms have a time complexity

Table 5 The time complexity of DRL-OSG algorithm and baseline methods

algorithm	Target product filtration	Construct graph	Graph optimization	Feature representation learning	Candidate groups generation	Spammer groups generation	Total of time complexity
GSDB (Ji et al. 2020)	$O(n)$	⊗	⊗	⊗	$O(n^2)$	$O(n)$	$O(n^2)$
GSBC (Wang et al. 2018a)	⊗	$O(n^3)$	⊗	⊗	$O(n^3)$	$O(n)$	$O(n^3)$
GroupStrainer (Ye and Akoglu 2015)	$O(n^2)$	$O(n^2)$	⊗	⊗	⊗	$O(n^2)$	$O(n^2)$
HoloScope (Liu et al. 2018)	⊗	$O(n^2)$	⊗	⊗	⊗	$O(n \log n)$	$O(n^2)$
SGDCTH (Zhang et al. 2023)	$O(n^2)$	$O(n^2)$	⊗	$O(n \log n)$	$O(n^2)$	$O(n)$	$O(n^2)$
DRL-OSG	$O(n^2)$	$O(n^2)$	$O(n^2)$	⊗	$O(n^2)$	$O(n)$	$O(n^2)$

of $O(n^2)$. The GSBC algorithm, due to its three-level loop method for building co-reviewing graphs, has a time complexity of $O(n^3)$.

Conclusion 5 Each stage of the DRL-OSG algorithm has a relatively low time complexity.

Results and analysis of ablation

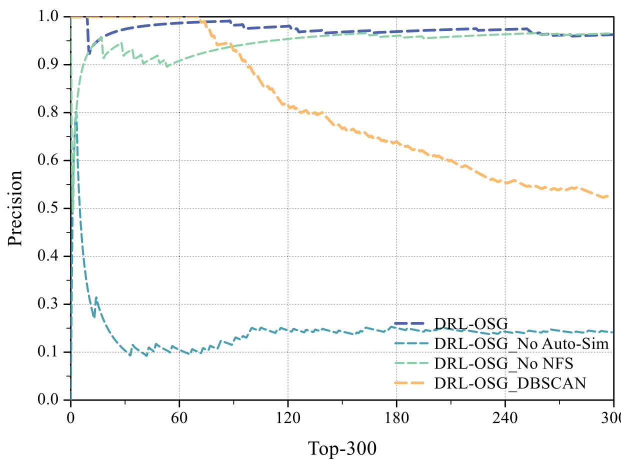
To delve deeper into the performance of the DRL-OSG algorithm, we conducted a series of ablation studies. Specifically, we devised the following variations of the DRL-OSG algorithm:

(1) DRL-OSG_No NFS: This variant omits the suspicious product filtering module inherent in the original DRL-OSG algorithm. Instead of filtering products with high suspicion levels based on the NFS threshold, it leverages all product reviewers to create a co-review graph, which then proceeds to subsequent tasks.

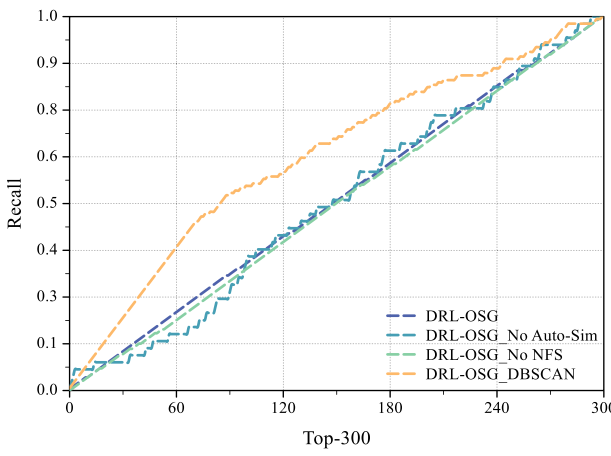
(2) DRL-OSG_No Auto-Sim: This variant excludes the dynamic optimization module for the co-review graph present in the DRL-OSG algorithm. Instead of refining the relationships within the co-review graph, it directly feeds this graph into the Ego-Splitting algorithm for clustering, followed by subsequent tasks.

(3) DRL-OSG_DBSCAN: This variant switches the clustering algorithm from the Ego-Splitting method to the DBSCAN algorithm (Ester et al. 1996) used by Zhang et al. (Zhang et al. 2023) for the generation of candidate spammer groups.

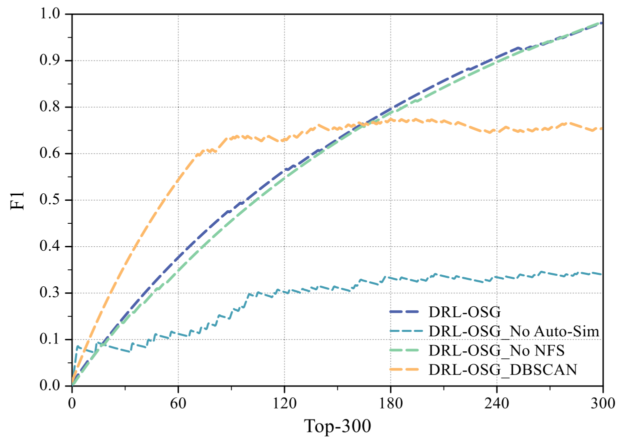
We also evaluated the performance disparities of the DRL-OSG algorithm and its three variants against three assessment metrics: Precision, Recall, and F1 score. The results are illustrated in Fig. 13.



(a) Precision



(b) Recall



(c) F1 values

Fig. 13 The precision, recall, and F1 values of the top-300 groups for DRL-OSG algorithm with its variants

The Precision curve in Fig. 13a illustrates that among the four algorithms compared, the DRL-OSG algorithm exhibits the most prominent results. Before the 136th group, the precision of the DRL-OSG_No NFS algorithm was inferior to that of the DRL-OSG algorithm, yet post this point, it demonstrated comparable precision with the DRL-OSG algorithm. This can potentially be attributed to the lack of filtering high-suspicion products, whereby a considerable amount of noise influenced the algorithm's performance due to the presence of reviews generated by low-suspicion products in the groups preceding the 136th one. In contrast, the DRL-OSG_No Auto-Sim algorithm consistently lagged behind the DRL-OSG algorithm, underlining the critical role of the dynamic optimization module of the co-reviewing graph in our algorithm. The DRL-OSG_DBSCAN algorithm outperformed the DRL-OSG algorithm until reaching the 74th group, after which it gradually fell behind. This indicates that before the 74th group, data in the co-reviewing graph might have had higher density, and post this group, the data potentially became sparser. Under these circumstances, the Ego-Splitting algorithm could better capture the structure in the graph, whereas the DBSCAN algorithm might overlook some crucial relationships or erroneously group unrelated reviewers into the same cluster.

The Recall curve in Fig. 13b illustrates that, on a general note, the DRL-OSG_DBSCAN algorithm outperforms the DRL-OSG, DRL-OSG_No NFS, and DRL-OSG_No Auto-Sim algorithms. This superior performance could possibly be ascribed to the DBSCAN leveraging a density-based clustering algorithm to better utilize spatial information, which might enable it to more adeptly identify and classify groups that have strong spatial correlations. However, when taking into consideration the Precision curve, it becomes evident that this advantage comes at the expense of a reduced precision rate. Consequently, even though the DRL-OSG_No NFS algorithm might detect a larger number of spammers, it also incurs a substantial number of false positives.

The F1 curve depicted in Fig. 13c illustrates that the F1 curve of the DRL-OSG algorithm is substantially more stable compared to those of the DRL-OSG_No NFS, DRL-OSG_No Auto-Sim, and DRL-OSG_DBSCAN algorithms. Following approximately the 165th group, the F1 score of the DRL-OSG algorithm consistently outperforms that of the DRL-OSG_No NFS, DRL-OSG_No Auto-Sim, and DRL-OSG_DBSCAN algorithms.

Conclusion 6 To encapsulate, although variants of the DRL-OSG algorithm may surpass the original in certain aspects, none achieve the comprehensive performance of the complete DRL-OSG algorithm when considered as a whole.

To further analyze the impact of individual modules within the DRL-OSG algorithm on its time efficiency, we conducted a series of runtime analysis experiments for both the DRL-OSG algorithm and its variants. These experiments were carried out on a computer equipped with a 2.4 GHz Intel Core i7 CPU and an NVIDIA RTX 3060 GPU. The outcomes of these experiments are illustrated in Fig. 14.

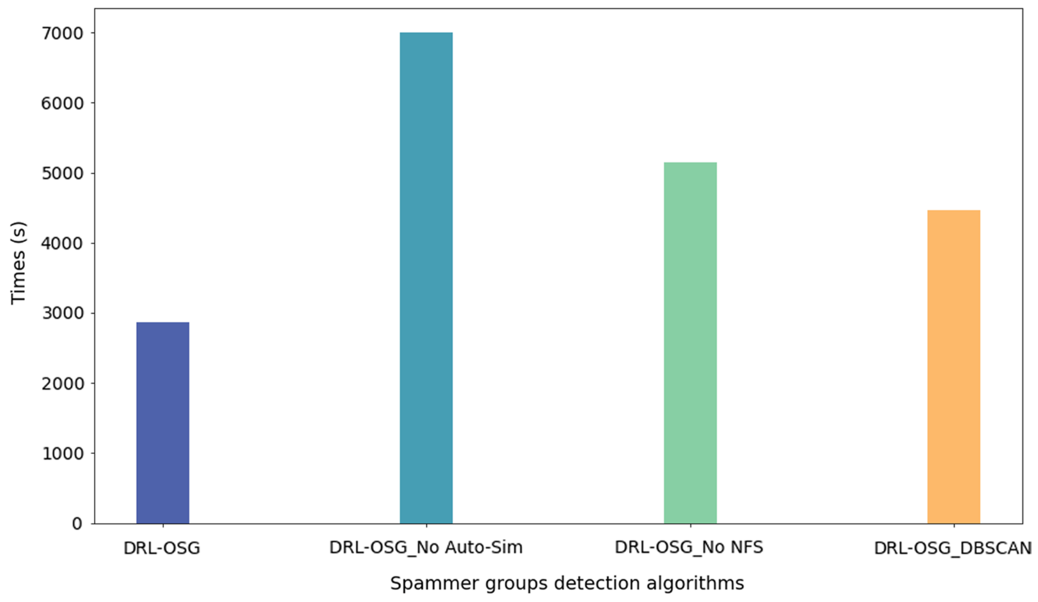


Fig. 14 The runtime of the DRL-OSG algorithm and its variants

Figure 14 presents that the DRL-OSG algorithm consumes the shortest runtime. In contrast, the DRL-OSG_No Auto-Sim variant exhibits the longest runtime, primarily due to the removal of the Auto-Sim module. This leads to an abundance of redundant relations in the co-reviewing graph, thereby increasing its complexity. Similarly, the DRL-OSG_No NFS variant also shows a higher runtime. This is attributed to the exclusion of the suspicious product detection module, which results in the inclusion of reviewers of normal products in the co-reviewing graph, thus escalating the algorithm's computational complexity. Additionally, the runtime for the DRL-OSG_DBSCAN variant is also higher compared to the DRL-OSG algorithm, indicating that our employed Ego-Splitting clustering algorithm is more efficient than the DBSCAN algorithm.

Conclusion 7 The DRL-OSG algorithm significantly enhances its operational efficiency by utilizing the NFS metric to filter out suspicious products, employing the behavioral consistency index to eliminate redundant relations, and generating candidate groups using the Ego-Splitting algorithm.

Conclusion

Online fraudulent activities have increasingly become a significant challenge in the e-commerce sector. Consumers are more reliant than ever on online reviews when purchasing products. Spammer groups, by disseminating deceptive reviews, mislead consumers regarding product quality, leading them to make purchasing decisions that may not align with their genuine needs. In this context, we introduce an spammer group detection algorithm based on deep reinforcement learning: DRL-OSG, tailored for detecting spammer groups within homogeneous information networks. The DRL-OSG algorithm, focusing on products, employs the NFS metric to filter products that are highly susceptible to spam attacks, thereby streamlining the data to be processed and enhancing the algorithm's overall efficiency. Subsequently, a homogeneous co-review graph is constructed using the filtered set of suspicious products. We then introduce the Auto-Sim algorithm, which dynamically adjusts the structure of the co-review graph to attain an optimal configuration. To identify overlapping entities within spammer groups, the Ego-Splitting algorithm clusters the optimized co-review graph, resulting in overlapping candidate groups. To further boost the algorithm's performance, DRL-OSG algorithm employs five metrics based on individual reviewers' fraudulent behavior characteristics and another five metrics rooted in the deceptive behavior patterns of spammer groups. These metrics refine and categorize the candidate groups, thereby enhancing the algorithm's efficacy.

While our proposed algorithm has achieved notable success, there remains room for enhancement. For instance, our current implementation of the Auto-Sim algorithm addresses the issue of consistent behavior parameter selection. In future iterations, we plan to harness deep reinforcement learning to concurrently adjust multiple parameters, aiming for a fully automated operation without manual intervention. Additionally, although our DRL-OSG algorithm excels in generating accurate spammer groups, the overlap rate of group members is relatively low. We aim to explore other overlapping clustering algorithms to uncover more overlapping entities, facilitating a better understanding and tracking of the activities of group members. Furthermore, we've observed that, when determining fraudulent behavior metrics for individuals and groups, we've employed mean values, which might not aptly represent the influence of each characteristic on deceptive actions. Hence, we intend to assign distinct weights to each feature, capturing the fraudulent behaviors of individuals and groups more precisely. Our future research will delve into these potential improvements to further elevate our algorithm's performance.

Abbreviations

FIM	Frequent item mining
NFS	Network footprint score
AD	Account duration
RD	Rating deviation
EXR	Ratio of extreme rating
MRO	Most reviews one-day
RTI	Review time interval
ISS	Individual suspicious score
GRD	Group rating deviation
GER	Group extreme rating ratio
GRT	Group review tightness
GOR	Group one day reviews
GS	Group size
GSS	Group suspicious score

Acknowledgements
The authors would like to thank the editor and anonymous referees for their constructive comments.

Author contributions
CW completed the writing and experiments for the manuscript, ZW and XF examined and validated experiments, and SJ and NL provided guidance and suggestions for revision. All authors read and approved the final manuscript.

Funding
This paper is supported by the Natural Science Foundation of China (71772107), the Natural Science Foundation of Shandong Province of China (ZR2023MF070, ZR2020MF044, ZR202102230289), Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety (NO. CSBD2022-ZD01), Shandong Education Quality Improvement Plan for Postgraduate (2021), the SDUST Research Fund.

Availability of data and materials
When certain data sharing requirements are met, the data is available upon request. Such requests should be sent to the first author of this paper.

Declarations

Competing interests
The authors declare that they have no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 15 November 2023 Accepted: 5 March 2024
Published: 4 June 2024

References

Akoglu L, Chandy R, Faloutsos C (2013) Opinion fraud detection in online reviews by network effects. In: Proceedings of the international AAAI conference on web and social media, Vol 7, No. 1, pp 2–11
Bom L, Henken R, Wiering M (2013) Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In: 2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL), pp 156–163
Chao J, Zhao C, Zhang F (2022) Network embedding-based approach for detecting collusive spamming groups on E-commerce platforms. Secur Commun Netw. https://doi.org/10.1155/2022/4354086
Chen T, Samaranayake P, Cen X, Qi M, Lan YC (2022) The impact of online reviews on consumers' purchasing decisions: Evidence from an eye-tracking study. Front Psychol. https://doi.org/10.3389/fpsyg.2022.865702
Choo E, Yu T, Chi M (2015) Detecting opinion spammer groups through community discovery and sentiment analysis. In: Data and applications security and privacy XXIX: 29th annual IFIP WG 11.3 working conference, DBSec 2015, Fairfax, Proceedings 29, pp 170–187
Dewang RK, Singh AK (2018) State-of-art approaches for review spammer detection: a survey. J Intell Inf Syst 50:231–264
Epasto A, Lattanzi S, Paes Leme R (2017) Ego-splitting framework: from non-overlapping to overlapping clusters. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 145–154
Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol 96, 34th edn. pp 226–231
Fei G, Mukherjee A, Liu B, Hsu M, Castellanos M, Ghosh R (2013) Exploiting burstiness in reviews for review spammer detection. In: Proceedings of the international AAAI conference on web and social media, pp 175–184
Fujimoto S, Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. In: International conference on machine learning, pp 1587–1596
Gabardo A, Berretta R, Moscato P (2019) Overlapping communities in co-purchasing and social interaction graphs: a memetic approach. Bus Consum Anal New Ideas. https://doi.org/10.1007/978-3-030-06222-4_9
Hu M, Xu G, Ma C, Daneshmand M (2019) Detecting review spammer groups in dynamic review networks. In: Proceedings of the ACM turing celebration conference-China, pp 1–6
Ji SJ, Zhang Q, Li J, Chiu DK, Xu S, Yi L, Gong M (2020) A burst-based unsupervised method for detecting review spammer groups. Inf Sci 536:454–469
Konda V, Tsitsiklis J (1999) Actor-critic algorithms. Adv Neural Inf Process Syst 12:1008–1014
Li H, Fei G, Wang S, Liu B, Shao W, Mukherjee A, Shao J (2017) Bimodal distribution and co-bursting in review spam detection. In: Proceedings of the 26th international conference on World Wide Web, pp 1063–1072
Liu S, Hooi B, Faloutsos C (2018) A contrast metric for fraud detection in rich graphs. IEEE Trans Knowl Data Eng 31(12):2235–2248
Luca M, Zervas G (2016) Fake it till you make it: reputation, competition, and Yelp review fraud. Manage Sci 62(12):3412–3427
Mukherjee A, Liu B, Glance N (2012) Spotting fake reviewer groups in consumer reviews. In: Proceedings of the 21st international conference on World Wide Web, pp 191–200
Mukherjee A, Kumar A, Liu B, Wang J, Hsu M, Castellanos M, Ghosh R (2013a) Spotting opinion spammers using behavioral footprints. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 632–640
Mukherjee A, Venkataraman V, Liu B, Glance N (2013b) What yelp fake review filter might be doing?. In: Proceedings of the international AAAI conference on web and social media, pp 409–418
Shehnepoor S, Togneri R, Liu W, Bennamoun M (2021) HIN-RNN: a graph representation learning neural network for fraudster group detection with no handcrafted features. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3123876
Shehnepoor S, Togneri R, Liu W, Bennamoun M (2022) Spatio-temporal graph representation learning for fraudster group detection. IEEE Trans Neural Netw Learn Syst 99:1–15
Wang Z, Hou T, Song D, Li Z, Kong T (2016) Detecting review spammer groups via bipartite graph projection. Comput J 59(6):861–874
Wang Z, Gu S, Zhao X, Xu X (2018a) Graph-based review spammer group detection. Knowl Inf Syst 55(3):571–597
Wang H, Zhou C, Wu J, Dang W, Zhu X, Wang J (2018) Deep structure learning for fraud detection. In: 2018 IEEE international conference on data mining (ICDM), pp 567–576
Xu C, Zhang J (2015) Towards collusive fraud detection in online reviews. In: 2015 IEEE international conference on data mining, pp 1051–1056
Xu C, Zhang J, Chang K, Long C (2013) Uncovering collusive spammers in Chinese review websites. In: Proceedings of the 22nd ACM international conference on information & knowledge management, pp 979–988
Ye J, Akoglu L (2015) Discovering opinion spammer groups by network footprints. In: Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2015, Porto, Portugal, Proceedings, Part I 15. pp 267–282
Zhang F, Hao X, Chao J, Yuan S (2020) Label propagation-based approach for detecting review spammer groups on e-commerce websites. Knowl-Based Syst 193:105520
Zhang F, Yuan S, Zhang P, Chao J, Yu H (2022a) Detecting review spammer groups based on generative adversarial networks. Inf Sci 606:819–836
Zhang F, Yuan S, Wu J, Zhang P, Chao J (2022b) Detecting collusive spammers on e-commerce websites based on reinforcement learning and adversarial autoencoder. Expert Syst Appl 203:117482
Zhang Q, Liang Z, Ji S, Xing B, Chiu DK (2023) Detecting fake reviewers in heterogeneous networks of buyers and sellers: a collaborative training-based spammer group algorithm. Cybersecurity 6(1):26
Zhang R, Peng H, Dou Y, Wu J, Sun Q, Li Y, Yu P S (2022) Automating DBSCAN via deep reinforcement learning. In: Proceedings of the 31st acm international conference on information & knowledge management, pp. 2620–2630
Zheng K, Li H, Qiu RC, Gong S (2012) Multi-objective reinforcement learning based routing in cognitive radio networks: Walking in a random maze. In: 2012 international conference on computing, networking and communications (ICNC), pp 359–363
Zheng M, Zhou C, Wu J, Pan S, Shi J, Guo L (2018) Fraudne: a joint embedding approach for fraud detection. In: 2018 international joint conference on neural networks (IJCNN), IEEE, pp. 1–8
Zhu C, Zhao W, Li Q, Li P, Da Q (2019) Network embedding-based anomalous density searching for multi-group collaborative fraudsters detection in social media. Comput Mater Contin 60(1):317–333

Publisher's Note