2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00290

Large-scale Fake Click Detection for E-commerce Recommendation Systems

Jingdong Li*[1], ✉Zhao Li*[2], Jiaming Huang[2], ✉Ji Zhang[3], Xiaoling Wang[1], Xingjian Lu[1], and Jingren Zhou[2]

[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
Email: jdl@stu.ecnu.edu.cn,xlwang@sei.ecnu.edu.cn,xjlu@cs.ecnu.edu.cn
[2]Alibaba Group, China
Email: {lizhao.lz,jimmy.hjm,jingren.zhou}@alibaba-inc.com
[3]Zhejiang Lab, Zhejiang, China
Email: zhangji77@gmail.com

***Abstract*—With the development of e-commerce platforms, e-commerce recommendation systems are playing an increasingly important role for the purpose of product recommendation. As a new attack model against e-commerce recommendation systems, the "Ride Item's Coattails" attack creates fake click information to establish the deceptive correlation between popular products and low-quality products in order to mislead the recommendation system of e-commerce platform to boost the sales of low-quality products. This attack is characterized by high concealment and strong destructiveness, which can cause great damage to e-commerce recommendation systems, and adversely affect the usability of the e-commerce platform and users' shopping experience. It is therefore of great practical significance to study how to quickly and effectively identify the false click information and the corresponding "Ride Item's Coattails" attack to better safeguard e-commerce recommendation systems. At present, there is no previously reported relevant research work conducted specifically for addressing the detection of the "Ride Item's Coattails" attack. In this work, we carried out pioneering work in analyzing and summarizing the characteristics of the false click information produced by attackers on the target products in the "Ride Item's Coattails" attack and designed a set of attack detection techniques suitable for e-commerce recommendation systems. Experimental results on real e-commerce datasets show that our proposed techniques can quickly and effectively detect the large-scale fake click information as well as the associated "Ride Item's Coattails" attack in e-commerce recommendation systems.**

***Index Terms*—E-commerce Recommendations, Ride Item's Coattails Attack, Attack Detection**

I. INTRODUCTION

Online shopping websites, such as TaoBao and Amazon, have become popular platforms for people to find and purchase products. Compared with traditional shopping, e-commerce provides customers with the opportunity of browsing voluminous product catalogs, comparing prices, and creating their wish lists with great convenience, which at the same time generates a large amount of user behavior data. By collecting and analyzing these user behavior data, e-commerce recommendation systems can provide customers with a better service based on their interests [1]. Specifically, in an item-to-user recommendation scenario, once the user clicks an item A,

*, These authors contributed to this work equally

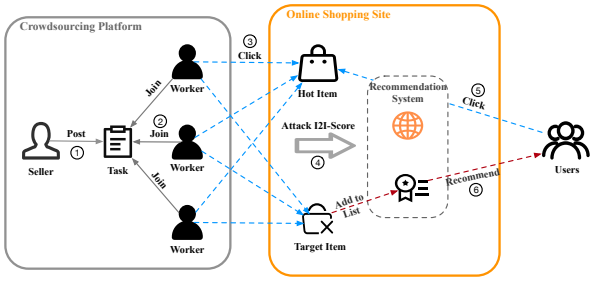


Figure 1. An Example of "Ride Item's Coattails" Attack

recommendation systems will figure out other items that are "similar" to A, then recommend them to the user. In TaoBao's recommendation system, item-to-item relevance score (abbreviated as I2I-score) is used to describe and measure the similar relationship between items. It is impacted by not only the similarity of the items' attributes but also the probability of the two items appearing in users' click lists at the same time. Of course, recommendation systems will consider other factors for a more comprehensive judgment [2], but the I2I-score turns out to be the most valuable one.

Such I2I recommendation strategy allows users to select highly relevant products together conveniently (e.g., beer and diapers). However, some malicious online sellers exploit it to increase the exposure of their low-quality products by producing a large-scale of fake clicks to affect the evaluation of the I2I-score. More specifically, they maliciously forge common clicks between the low-quality items and hot items to raise the I2I-score between them. So when normal users are browsing through these hot items, the target low-quality items can also have a higher probability of occurrence in the resulted item recommendations list, thereby increasing the exposure of these items. What's more, malicious online sellers always try to associate multiple hot items with target items to maximize the exposure opportunities. We name this kind of attack as the **"Ride Item's Coattails" attack**.

The "Ride Item's Coattails" attack can cause serious negative impacts on the I2I-score leading to higher I2I-score be-

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00290

tween hot items and target low-quality items, thereby reducing the recommendation accuracy of the online shopping sites' recommendation system and misleading normal users to buy low-quality or even fake items from those malicious online sellers. This not only damages the user's shopping experience, but also affects the reputation of the e-commerce platform. Therefore, in this paper, we aim to study how to effectively detect the large-scale fake click information as well as the newly-emerged "Ride Item's Coattails" attack described above in e-commerce recommendation systems for minimizing its negative impact on normal users and online shopping sites.

To increase their items' exposure, some online sellers manipulate the ranking of the items in the recommendation list with the help of crowd workers (As shown in Fig. 1). First, a malicious online seller from an online shopping site posts a task describing the target items, expectations, and remunerations on the crowdsourcing platform. Then, some experienced workers accept this task and launch a large-scale "Ride Item's Coattails" attack on the online shopping site. After the attack, target items gain higher probabilities of appearing at the top of the item recommendation list.

These attacks conducted by crowd workers instead of sellers bring many challenges: (1) Compared with sellers, crowd workers are less relevant to the target item, so it is hard to detect them directly; (2) Crowd workers with some hot items click records by themselves are very similar to normal ones and difficult to be detected even with manual efforts; (3) Experienced workers will follow some particularly designed guidelines to disguise themselves as normal users [1], such as clicking on other irrelevant items or increasing the number of clicks on hot items; (4) With the increasing popularity of deceptive items, some normal users may also be attracted by them and contribute clicks. In other words, both attackers and target items largely behave like normal records.

To tackle these challenges, we first extract a large-scale user-item bipartite graph from Taobao and investigate the corresponding user and item behavior that are highly likely to be participants in a "Ride Item's Coattails" attack. Then, we analyze the characteristics of the "Ride Item's Coattails" attack from the perspective of users and items. By integrating structural features and correlations (both user-based and item-based), we design a discriminative framework to detect the "Ride Item's Coattails" attack. Through experimental comparisons with competitive baselines, we show that our framework is efficient and effective.

We summarize the major contributions of this paper as follows:

- We discovered and defined a special attack mode against e-commerce recommendation systems and named it the "Ride Item's Coattails" attack. To our best knowledge, ours is the first reported work to investigate this type of attack;
- We provide a comprehensive analysis of the "Ride Item's Coattails" attack from the perspective of users and items;
- We propose a novel detection framework that can effectively detect the large-scale fake click information as well as the associated "Ride Item's Coattails" attack in e-commerce recommendation systems;
- We have conducted experiments on a large-scale real-world e-commerce dataset. The experimental results show the effectiveness and efficiency of our attack detection framework and algorithms.

II. RELATED WORKS

The detection of the "Ride Items Coattails" attack is related to the research works in the following areas: dense subgraph mining, false information detection, and community detection in bipartite graphs. However, they are not applicable to directly solve the detection problem of the "Ride Items Coattails" attack.

A. Dense Subgraph Mining Algorithms

The "Ride Item's Coattails" attack model determines that the subgraph formed by hot items, target items, and attackers must be a cohesive bipartite graph. The densest structure in a bipartite graph is called the complete bipartite graph (also known as *biclique*, as defined in Definition 1) which has been widely used to capture cohesive bipartite subgraphs in a broad spectrum of bipartite graph applications [3], [4].

Definition 1. *Given a bipartite graph $G = (U, V, E)$, a biclique $C = (L \cup R)$ is a complete bipartite subgraph of G induced by a pair of two disjoint subsets $L \subseteq U$, $R \subseteq V$, such that $\forall u \in L, \forall v \in R, (u, v) \in E$.*

However, enumerating maximal bicliques in a bipartite graph is #P-complete [5], and it is impossible to directly enumerate all the maximal bicliques in a large-scale e-commerce bipartite graph [6]. Moreover, the cohesive subgraph, composed of users, target items, and hot items, may not be a perfect biclique. The definition of quasi-biclique can tackle this problem [7], but as far as we know, the existing quasi-biclique mining algorithms [7], [8] all aim at solving the maximum (vertex) quasi-biclique search problem. This problem is proved to be NP-hard [9] and can only output one near biclique.

B. False Information Detection

The existing false information detection methods can broadly be categorized into three categories: propagation-modeling based, feature-based, and graph-based. Propagation-modeling algorithms use information spread models to identify false information [10], [11]. Feature-based algorithms leverage the unique characteristics for detection by using them as features in a machine learning model or rule-based framework [12], [13]. However, both of them require side information, such as true information or labeled data.

Graph-based algorithms detect attackers by identifying suspicious dense regions of the graph. Such methods are potentially harder to evade because creating fake clicks unavoidably generates edges in the graph. COPYCATCH [4] detects temporally bipartite cores that are ill-gotten "Likes" of Facebook, but without timestamps, it degenerates into enumerating all

the maximal bicliques. CATCHSYNC [14] looks for synchronized behavior in the graph without side information, which forms dense bipartite subgraphs. However, this method is not robust against experienced adversaries and lacks performance guarantees on the detection algorithm. FRAUDAR [15] proposes suspiciousness metrics to catch fraudulent blocks (dense subgraphs), which are robust to camouflage. However, without determining the number of blocks in advance, the algorithm can't find multiple attack groups.

C. Community Detection in Bipartite Graphs

Newman et al. [16] first propose the modularity metric to identify communities in general graphs. Guimera et al. [17] modify the definition given by Newman [16] and introduce a modularity measurement for bipartite graphs.

Raghavan et al. [18] propose a famous label propagation algorithms, which uses the network structure to guide its progress for finding communities in general graphs. In addition, some variants of community detection algorithms have been proposed to discover well-connected communities [19] and dense clusters [20]. Unfortunately, community detection algorithms often require a priori information such as density or number of clusters as input. This preliminary information is hard to acquire ahead from large-scale e-commerce networks, making the above algorithms unable to be directly applied to "Ride Item's Coattails" attack detection.

III. Problem Definition

We assume that there are a set of m users $U = \{u_1, ..., u_m\}$ and a set of n items $V = \{v_1, ..., v_n\}$ connected according to a bipartite graph $G = (U \cup V, E)$, where $(u_i, v_j, p) \in E$ is user u_i has clicked item v_j p times. We now describe our attack model and then our problem definition.

A. Attack Model

We assume that crowd workers are hired to use accounts they control to add a large number of fake clicks/edges between their account and items, inducing a dense subgraph. This general characteristic of the "Ride Item's Coattails" attack was found to be true in a real Taobao dataset. Many other papers use dense blocks to detect fraud [4], [14], [15], [21]. However, different from the "who-Likes-what" graph on Facebook [4], each user can only "Like" a page once. In the "Ride Item's Coattails" attack, an account can click an item multiple times, making the size of the attack group smaller and more challenging to be detected. Also, the clicked items include not only target items but also hot items and a small part of random normal items. Therefore, it is unreasonable to use a non-discriminatory metric [4], [15] to distinguish suspicious nodes.

In real e-commerce recommendation systems, there are always adversarial challenges. Experienced crowd workers will add arbitrary "camouflage" to disguise their fraud, i.e., edges pointing from their user accounts to random normal items. This work assumes that attackers have complete knowledge of how the recommendation system works and the attack detection mechanisms. Under such a strict attack model, the false click information created by the attacker is more purposeful and brings more significant challenges to our detection approach. In Section IV, we conducted a deeper analysis of the strategies that the attacker might adopt.

B. Desired Properties of Detection Approach

Our goal is to detect suspicious nodes typically indicative of groups of users and items participating in "Ride Item's Coattails" attacks, and the problem is defined as:

Given a bipartite graph $G = (U \cup V, E)$ of clicks between users U and items V, we aim to find a set of suspicious users U_{sus} and a set of suspicious target items V_{sus} that are involved in a set of "Ride Item's Coattails" attack groups $g = \{g_1, g_2, ..., g_n\}$ in the graph G, $U_{sus} = \{g_1.u_{sus} \cup g_2.u_{sus} ... \cup g_n.u_{sus}\}$ and $V_{sus} = \{g_1.v_{sus} \cup g_2.v_{sus} ... \cup g_n.v_{sus}\}$.

The detection approach should satisfy the following properties:

(1) *Effectiveness and efficiency:* It accurately detects groups of "Ride Item's Coattails" attack with suspicious behaviors and is applicable to large e-commerce graphs;
(2) *Side information oblivious:* Neither the labeled node nor the attribute information of the node is required. It only defines suspicious behavior using a topological structure;
(3) *Camouflage restriction:* It can restrict the maximum number of false clicks/edges (i.e., an upper bound) that attackers can add without being detected;
(4) *Easy of use for end-users:* a) It should support scoring for each abnormal node, convenient for business experts to select the top-k nodes for analysis and punishment; b) It can explicitly limit the detected group's size to avoid the misjudgment of group-buying phenomenon.

IV. "Ride Item's Coattails" Attack Analysis

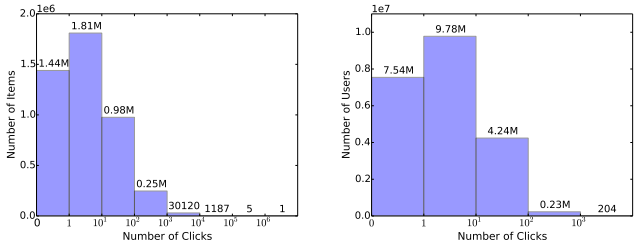
To have a more comprehensive understanding of the "Ride Item's Coattails" attack, we first extract a large-scale user-item bipartite graph from Taobao's historical data within the past year, as the basis for analyzing the underlying data characteristics of the attack. Without loss of generality, we conduct stratified sampling on various items to generate a representative bipartite graph. Then, we store it in a table (named TaoBao_UI_Clicks) with three columns: *User_ID*, *Item_ID* and *Click*. Specifically, one record (1, 1, 3) in TaoBao_UI_Clicks means that user with $id = 1$ has clicked the item with $id = 1$ three times.

Table I
Data Scale of TaoBao_UI_Clicks

User	Item	Edge	Total_click
20M	4M	90M	200M

Table II
Data Statistics of TaoBao_UI_Clicks

	Avg_clk	Avg_cnt	Stdev
User	11.35	4.32	33.34
Item	54.94	20.49	992.78



(a) Distribution of Items' Clicks (b) Distribution of Users' Clicks

Figure 2. Distribution of Clicks

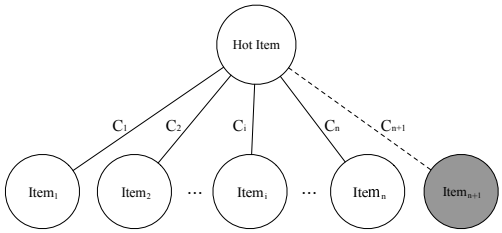


Figure 3. I2I-score Calculation Model

The scale and statistics information of the dataset are shown in Table I and II respectively. The *Edge* represents the number of click records in the dataset, and *Total_click* represents the sum of *Click*. The statistics data include the average number of clicks (*Avg_clk*), the average number of item clicks (*Avg_cnt*), and the standard deviation of the total number of clicks (*Stdev*). Moreover, the distributions of the clicks are shown in Fig. 2a and Fig. 2b. Based on this real-world dataset, we conduct a comparative analysis of the "Ride Item's Coattails" attack which covers two aspects: user behavior and item behavior. A summary of the behavior of "Ride Item's Coattails" attack is presented at the end of this section.

A. User Behavior

The reason why the "Ride Item's Coattails" attack can work is that, by clicking on hot items and target items, crowd workers can increase the I2I-scores between hot items and target items. Then, normal users who have clicked on these hot items will be misled by the e-commerce recommendation system to browse low-quality target items.

To give a clear explanation of this process, we introduce the calculation model of the I2I-score as shown in Fig. 3. Assuming that there are co-click records of a hot item and an ordinary item sets $\{item_1, item_2, ..., item_n\}$. The C_i between the $hot\ item$ and an ordinary item $item_i$ represents the number of clicks on item $item_i$ under the condition that the $hot\ item$ has been clicked before. So the I2I-score S_i between the hot item and the ordinary item $item_i$ can be calculated as follows:

$$S_i = \frac{C_i}{C_1 + C_2 + ... + C_n} \tag{1}$$

Next, we give three basic assumptions on the behavior of crowd workers (abnormal users) in a "Ride Item's Coattails" attack:

Assumption 1. *Crowd workers hope to get the remuneration as much as possible without being detected by the e-commerce risk control system;*

Assumption 2. *Crowd workers have some specifically designed strategies to realize Assumption 1;*

Assumption 3. *All crowd workers follow similar strategies (as mentioned in Assumption 2).*

Based on the above I2I-score model and three assumptions, we focus on analyzing the strategies that crowd workers will adopt to discover some distinctive anomalous patterns.

Suppose that an online seller posts a "Ride Item's Coattails" attack request and the target item assigned by he/she is $item_{n+1}$. Then, the crowd worker who has accepted this request will click the hot item and the target item $item_{n+1}$ to establish a link at first. At this time, C_{n+1} is initialized to 1.

Because of the purpose to study the click behavior of abnormal users and Assumption 3, we can ignore other user's influence during this attack and the value of $C_1, C_2, ..., C_n$ are fixed. For Assumption 1, crowd workers' goal is to increase the I2I-scores between a target item and hot items as much as possible (because the I2I-score is positively correlated with the target item's exposure and the compensation they can receive), while keeping their click behavior similar to normal users.

One of the simplest and effective strategies is to set a click budget for each attack because the risk control system can easily detect excessive clicks on a item from a user. Crowd workers need an effective click strategy to maximize the I2I-score with limited number of clicks. Assuming that the click budget assigned to this attack task is C_b, the number of additional clicks to maximize the I2I-score is C, the number of additional clicks on the target item C_{n+1} is C' and the number of additional clicks on other items is $C - C'$, we can get the I2I-score calculation function as follow:

$$S_{n+1} = \frac{C_{n+1} + C'}{C_1 + C_2 + ... + C_n + (C_{n+1} + C') + C - C'} \tag{2}$$

where $0 \le C' \le C \le C_b - 2$. For establishing a link between a hot item and a target item, two clicks have to be consumed.

Note that given constants m and n, $f(x) = \frac{m+x}{n+x}$ is a strictly monotone increasing function for $n \ge m > 0, x \ge 0$. So, $f(x) \le f(C_b - 2)$ for $0 \le x \le C_b - 2$, and we can derive the following inequality where $C_1, C_2, ..., C_{n+1}$ are constants:

$$\begin{aligned} S_{n+1} &= \frac{C_{n+1} + C'}{C_1 + C_2 + ... + C_n + (C_{n+1} + C') + C - C'} \\ &\le \frac{C_{n+1} + C}{C_1 + C_2 + ... + C_n + C_{n+1} + C} \\ &\le \frac{C_{n+1} + C_b - 2}{C_1 + C_2 + ... + C_n + C_{n+1} + C_b - 2} \end{aligned} \tag{3}$$

S_{n+1} can get the maximum value iff $C' = C = C_b - 2$. Therefore, the best strategy for crowd workers is to click the hot item once and then click the target item as much as possible. In a realistic scenario, to counter the risk control system of the shopping sites, crowd workers will not only

click the hot item and the target item, but they will also click other uncorrelated items to disguise themselves. Next, based on the above behavior analysis results, we will look for users with such click patterns from the dataset.

In the first step, we classify items in the dataset into hot items and ordinary items. As shown in Fig. 2, the distribution of user clicks on the items exhibits a heavy-tailed distribution. It follows Pareto's principle [22], which means that the clicks on a few hot items (about 20%) account for most of the total number of clicks (about 80%). We rank the items by the number of clicks and get the hot items set by summing clicks up to 80% of the total number of clicks. The threshold T_{hot} equals to the number of clicks of the last hot items which is 1,320 in the dataset. In other words, items with the number of clicks greater than or equal to 1,320 are hot items.

In the second step, based on the above analysis, we screen out the suspicious users who have clicked both hot and ordinary items at the same time and have clicked ordinary items more than a certain number of times from the dataset. In order to determine this threshold T_{click}, we assume that the average total number of clicks (*Avg_clk* in Table II) is the most reasonable total number of clicks that the crowd users will use to disguise themselves. The distribution of the crowd works' clicks on the items also exhibits a heavy-tailed distribution, that is, the number of clicks on a few target items (about 20%) account for most of the total number of clicks (about 80%). So we can figure out the T_{click} by Equation 4,

$$T_{click} = \frac{Avg\_clk \times 80\%}{Avg\_cnt \times 20\%} \tag{4}$$

where $Avg\_clk = 11.35$ and $Avg\_cnt = 4.23$ in the table. Then, an ordinary item whose number of clicks greater than or equal to 12 is an abnormal click record.

Table III
PART OF THE CLICK RECORD OF A SUSPECT

ID	Click	Total_click	Hot
1	2	16,853	1
2	1	9,928	1
3	1	6,178	1
4	1	1,912	0
5	2	1,558	0
6	**13**	1,319	**0**
7	1	632	0
8	**13**	368	**0**

In this way, we roughly screen out more than 1.4 million users ($\geq 7\%$ of all users) with abnormal click records. Here we show a part of the click records of a representative and suspected user in Table III for analysis. We replace the real *Item_ID* with the sequence *ID* for data privacy, and the *Click* column represents how many times this user have clicked this item. The *Total_click* column represents the number of clicks from all users. If the total number of clicks of the item is higher than the threshold T_{hot}, then the value of *Hot* is 1, otherwise it is 0.

As shown in Table III, the suspicious user intentionally clicks hot items (*ID*=1,2,3) as few as possible, and frequently clicks the target items (*ID*=6,8). The suspicious user also clicks some other items (*ID*=4,5,7) to confuse the risk control system. To increase the I2I-score efficiently, the number of clicks of such non-target items is relatively small. These click behaviors are consistent with the results of attack intention analysis.

Here we show a part of the click records of a normal user in Table IV as a comparison. It can be noticed that the normal user clicks hot items (*ID*=1,2) more frequently. For ordinary items (*ID*=3,4,5,6) the number of clicks is relatively small.

Table IV
PART OF THE CLICK RECORD OF AN ORDINARY USER

ID	Click	Total_click	Hot
1	**19**	9,206	**1**
2	4	3,816	1
3	1	1,013	1
4	1	974	0
5	1	894	0
6	1	632	0

In conclusion, we summarize the typical behavioral characteristics of the abnormal users involved in a "Ride Item's Coattails" attack as follows (in order of significance):

(1) They click ordinary items more frequently, and the number of clicks of some ordinary items (suspicious target items) is greater than a certain threshold T_{click};
(2) The number of clicks of hot items is far less than that of ordinary items, and the average number of clicks of hot items is extremely small (< 4);
(3) Because of the need to launch an attack and disguise themselves at the same time, it causes significant differences in their number of clicks on different ordinary items.

B. Item Behavior

A typical scenario of online sellers launching a "Ride Item's Coattails" attack is that the target items cannot attract users' clicks due to their low quality or poor design. Sellers hope to increase the exposure of their items in this way artificially. In this scenario, the total number of clicks on the target item and the number of users who have clicked it should be small at the beginning. After the "Ride Item's Coattails" attack, the total number of clicks of the target items will increase. Because those additional clicks are almost from the crowd workers, the click behavior of the target items will show the characteristics of a high number of clicks from few users.

The suspicious items selected here come from the ordinary item sets that appear in the abnormal click records introduced in the previous section, and we get more than 600,000 suspicious items ($\geq 15\%$ of all items). In order to eliminate the influence of the existing "Ride Item's Coattails" attack on normal items, we select the items that newly appear in item tables (from the business department) in the past month. Table V shows the statistics of a suspicious item with a total number of clicks of 368 and a normal item with a total number of clicks of 404.

It can be found that when the total number of clicks is not much different ($< 10\%$), the number of users (*User_num*) who have clicked normal items is twice as that of suspicious items. The mean (*Mean*) and standard deviation (*Stdev*) of

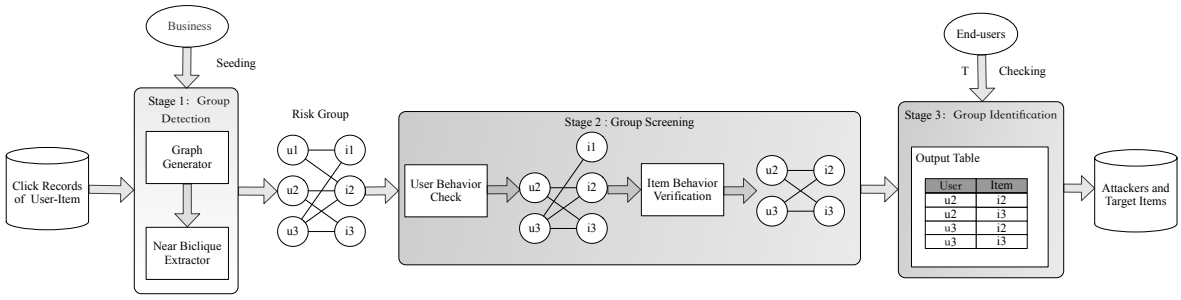


Figure 4. "Ride Item's Coattails" Attack Detection Framework

Table V
STATISTIC DATA OF SUSPICIOUS ITEM AND NORMAL ITEM

Total_click	Mean	Stdev	User_num	Max	Min
368	3.64	7.36	101	40	1
404	1.99	2.52	203	17	1

the number of clicks of normal items are lower than those of suspicious items as well. We also count the frequency that suspicious users appear in the click lists of the normal items and suspicious items, and the results are 0.49% for the normal items and 1.98% for the suspicious items, respectively.

In conclusion, we summarize the typical behavioral characteristics of the target items involved in a "Ride Item's Coattails" attack as follows (in order of significance):

(1) When the total number of clicks are close, the number of users who have clicked the target item is far less than that of a normal item;
(2) The proportion of abnormal users in the click list of a target item is much higher than that of a normal item.

C. Summary

From the above analysis results, it can conclude that crowd workers and target items have unique behavioral characteristics that are significantly different from those of normal users and items. These unique behavioral characteristics can be used as an effective additional filtering condition to help us detect crowd workers and target items. However, the screening techniques used in the above analysis are very rough and inaccurate. It leaves more than 7% of all users and 15% of all items in the results. In order to detect the "Ride Item's Coattails" attack in e-commerce recommendation systems quickly and effectively, we need more systematic approaches.

V. "RIDE ITEM'S COATTAILS" ATTACK DETECTION

In this section, based on the analysis results of the behavior of abnormal users and items, we first present a naive algorithm that can judge whether an item or user is abnormal independently. Then, we propose a novel "**R**ide **I**tem's **C**oattails" Attack **D**etection framework (RICD) for systematically detecting the "Ride Item's Coattails" attack groups in e-commerce recommendation systems.

A. Naive Algorithm

The basic idea can be summarized as follows. If most of the users who click an ordinary item have clicked a large number of hot items, it is very likely that this ordinary item is a target item and the users are suspicious users. The pseudocode of the algorithm is shown in Algorithm 1.

Algorithm 1 Naive Algorithm

Input: User List U, Item List L, Hot Item Threshold T_{hot}, Risk Threshold T_{risk}.
Output: Abnormal Item Set S.

1: $NewItem, HotItem \leftarrow \emptyset$;
2: **for** each item $l \in L$ **do**
3: **if** $l.total\_click < T_{hot}$ **then**
4: $NewItem.append(l)$;
5: **else**
6: $HotItem.append(l)$;
7: **for** each user $u \in U$ **do**
8: $u.Alpha \leftarrow$ GETALPHA$(u, HotItem)$;
9: **for** each item $l \in L$ **do**
10: $l.RiskScore \leftarrow$ sum $Aplha$ of $l's$ neighbors;
11: **if** $l.RiskScore > T_{risk}$ **then**
12: $S.append(l)$;
13: **return** S;
14: **function** GETALPHA$(u, HotItem)$
15: $Alpha \leftarrow 0$;
16: **for** each item l adjacent to u **do**
17: **if** $l \in HotItem$ **then**
18: $Alpha \leftarrow Alpha + l.click$;
19: **return** $Alpha$

The input of the algorithm includes the user list U, item list L, hot item threshold T_{hot}, and risk threshold T_{risk}. Here, the risk threshold is used to balance the trade-off between precision and recall. The algorithm classifies items into hot items and new items in Line 2 to 6. New items are treated as potential target items in this algorithm. The algorithm invokes function *getAlpha* to calculate the $Alpha$ for every user in Line 8. Then, we can get $riskscore$ of each item by summing their neighbors' $Alpha$ and append items with $riskscore > T_{risk}$ to abnormal item set S in Line 10 to 12. Utilizing the obtained abnormal item set S for classification, we can figure out abnormal users in the same way as Algorithm 1. The advantages of the naive algorithm is that it is very intuitive and can efficiently detect some suspicious users and items, but it has the following disadvantages: (1) The calculation of

$riskscore$ is independent and does not consider the impact between items; (2) Thresholds are hard to set in advance. Therefore, we need a more systematic solution.

B. Detection Framework

The framework of RICD is shown in Fig. 4, which can be divided into the following three sequential modules from left to right:

(1) **Suspicious Group Detection Module.** This module aims to quickly locate suspicious "Ride Item's Coattails" attack groups in large-scale e-commerce networks. In such networks, abnormal nodes only take up a tiny portion, so it is unreasonable to apply a filtering algorithm directly in original large-scale networks.

As we mentioned in Section IV, creating fake clicks between attackers and target items unavoidably generates edges, which will form dense subgraphs. So the intuitional idea is that we can regard the suspicious "Ride Item's Coattails" attack group as a near biclique and design a near biclique extraction algorithm (Algorithm 2 Line 3) for the "Ride Item's Coattails" attack detection. In Section V-C, we will introduce related definitions and algorithm (Line 3) in details.

Algorithm 2 Suspicious Group Detection Algorithm

Input: User-Item Click Table CT, Known Attacker $Seed$, Parameters $Para$.
Output: Suspicious Attack Group Sets B.

1: $BiGraph \leftarrow \emptyset$; ▷ Store bipartite graph
2: $BiGraph \leftarrow$ GRAPHGENERATOR($CT, Seed$);
3: $B \leftarrow$ NearBicliqueExtract($BiGraph, Para$);
4: **function** GRAPHGENERATOR($CT, Seed$)
5: **if** size of $Seed$ is not null **then**
6: **for** each node in $Seed$ **do**
7: $BiGraph.append$(MaxBiGraph($node$));
8: $Seed.remove(node)$;
9: **else**
10: $BiGraph \leftarrow$ TableToBiGraph(CT);
11: **return** $BiGraph$

The input of this module includes the user-item click table, some known abnormal items or users provided by the business department, and the parameters used in the near biclique extraction algorithm. We store the intermediate results in the *Bigraph* in the form of a bipartite graph in Line 2 and use the near biclique extraction algorithm to detect suspicious groups in Line 3. It should be noted that in the suspicious group detection algorithm, the known *Seed* provided by the business department is only utilized as an auxiliary input in Lines 5 to 8. Its main purpose is to help prune the input bipartite graph and locate the suspicious groups quickly. So in an actual attack detection scenario, even if there are no known abnormal items or users, this module can still work properly.

(2) **Suspicious Group Screening Module.** The main function of this module is to further screen the output suspicious groups by the previous module based on the relevant conclusions of the "Ride Item's Coattails" attack behavior analysis in Section IV and output more accurate attack groups. We divide the group screen process into the following two steps: user behavior check and item behavior verification.

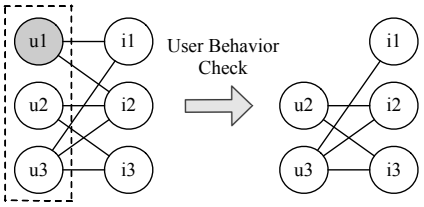


Figure 5. Example of User Behavior Check

We first introduce the user behavior check step. For the suspicious group $G = \{u_1, u_2, u_3, i_1, i_2, i_3\}$ produced by the suspicious group detection module and their click records $R = \{C_1^1, C_1^2, C_2^2, C_2^3, C_3^1, C_3^2, C_3^3\}$ (C_1^2 representing the number of times user u_1 clicked on the item i_2), shown in Fig. 5. We check the corresponding click list of each user in the user list $U = \{u_1, u_2, u_3\}$. For example, for suspicious user u_1, its click list is $I_1 = \{i_1, i_2\}$. We will check whether they are hot items (clicks greater than T_{hot}). The same method is applied to u_2 and u_3. Supposed that we find I_1 is a hot item, while I_2 and I_3 are not, which we have $C_2^2, C_3^2 > T_{click} > C_1^2$. Then, items I_2 and I_3 clicked by two users u_2 and u_3 at the same time are more likely to be involved in a "Ride Item's Coattails" attack, because neither I_1 or I_2 is a hot item and C_1^1 and C_1^2 are so small. Then, we can remove u_1 from G and get a more accurate attack group $G' = \{u_2, u_3, i_1, i_2, i_3\}$. However, i_1 cannot be removed from G at this time, because it is still possible that I_1 and I_2 launch a "Ride Item's Coattails" attack together.

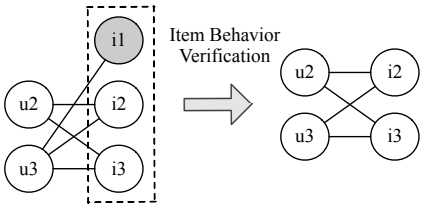


Figure 6. Example of Item Behavior Verification

Next, we will conduct the item behavior verification step, as shown in Fig. 6. We will check the coincidence degree of the clicked user sets of $I = \{i_1, i_2, i_3\}$ in turn. The clicked sets of i_2 and i_3 are exactly the same. If $C_3^2 \gg C_3^1$ is also met at the same time, then the link between u_3 and I_1 is likely to be a disguise, which means that the item I_1 is an ordinary item and can be removed from the attack group G'. We finally get the output suspicious attack group $G' = \{u_2, u_3, i_2, i_3\}$.

(3) **Suspicious Group Identification Module.** The main function of this module is to convert the output suspicious attack groups into the form of a user-item table ordered by the degree of suspicion and make the framework easy to use for end-users (e.g., business experts and downstream applications). To achieve these goals, we propose two optimization strategies:

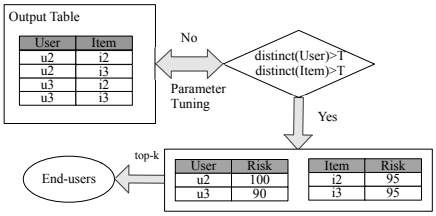


Figure 7. Bussiness-oriented Group Identification Strategy

Ranking strategy based on the risk score. The core idea of this strategy is to efficiently assign a risk score to each user and item in the user-item table returned. The user's risk score refers to the number of suspicious items clicked by the user. The item's risk score can be obtained by calculating the average risk score of the clicked users. In this way, we can sort users and items according to their risk scores. Then, end-users can conveniently select the top-k users and items to punish and amend the manipulated I2I-score.

Parameter adjustment strategy based on feedback. In extreme situations, the scale of suspicious groups output by the first two modules is far less than expected, which means that some parameters in the parameter space may not be properly specified. Because these parameters are highly interpretable, we can quickly adjust the parameters based on our experience (e.g., decrease T_{click}) to increase the recall of the result. This strategy can improve the scalability and robustness of the framework.

These two strategies can be coordinated, as shown in Fig. 7. Each time we will check whether the number of users and items in the output table meets the expectation T given by the end-users. If not, we will adjust the parameters of the suspicious group detection module and suspicious group screen module according to the feedback until the output result meets the expectation T. Then, with the help of the ranking strategy based on risk scores, the end-users can choose what to evaluate.

C. Near Biclique Extraction Algorithm

In Section II-A, we have pinpointed two main shortcomings of utilizing the biclique mining algorithms to detect the "Ride Item's Coattails" attack in large-scale e-commerce bipartite graphs as being (1) the high latency brought by MBE algorithm with a high complexity; (2) the exceedingly strict structural requirements of the biclique (Definition 1). In order to solve these two problems, we define a (k_1, k_2, α)-extension of biclique and design a (k_1, k_2, α)-extension biclique extraction algorithm to detect the "Ride Item's Coattails" attack. Here, we first give the definition of the α-extension of biclique [23].

Definition 2. *Given a bipartite graph $G = (U, V, E)$, an α-extension of a biclique $C = (L \cup R)$ is an ordered pair (L', R') where $L' \subseteq (U - L)$ and $R' \subseteq (V - R)$ such that at least $\alpha \times 100\%$ of the nodes in each of the node sets L and R are connected through edges to all nodes in R' and L', respectively.*

By adjusting the parameter α properly, we can extract attack groups in various scales. This definition reflects the click behavior of the crowd workers as well as target items in reality and solves the problem caused by the strict definition of biclique. To improve computational efficiency and find dense subgraphs containing more fake click records, we introduce two parameters k_1 and k_2 to limit the minimum size of the results as given in Definition 3.

Definition 3. *Given a bipartite graph $G = (U, V, E)$, an (α, k_1, k_2)-extension biclique is an ordered pair $(L \cup L', R \cup R')$ where (L, R) is a biclique and (L', R') its α-extension such that $|L| \geq k_1, |R| \geq k_2$ and at least $\alpha \times 100\%$ of the nodes in each of the node sets L and R are connected through edges to all nodes in R' and L', respectively.*

Algorithm 3 (k_1, k_2, α)-extension Biclique Extraction Algorithm

Input: User-Item Graph $BiGraph$, Parameters $Para$.
Output: Suspicious Attack Group Sets B.

```
1: BiGraph ← CorePruning(BiGraph, Para);
2: B ← SquarePruning(BiGraph, Para);
3: function CorePruning(BiGraph, Para)
4:     for each user u ∈ BiGraph do
5:         if u.degree < ⌈Para.α × Para.k2⌉ then
6:             BiGraph.remove(u);
7:     for each item i ∈ BiGraph do
8:         if i.degree < ⌈Para.α × Para.k1⌉ then
9:             BiGraph.remove(i);
10:    return BiGraph
11: function SquarePruning(BiGraph, Para)
12:    for each user u_i ∈ BiGraph do
13:        num ← 0;
14:        for each user u_j ∈ BiGraph|u_i do
15:            if |u_i.adj ∩ u_j.adj| ≥ ⌈Para.k2 × Para.α⌉
       then
16:                num ← num + 1;
17:        if num < k1 then                          ▷ α > 0
18:            BiGraph.remove(u_i);
19:    for each item l_i ∈ BiGraph do
20:        num ← 0;
21:        for each item l_j ∈ BiGraph|l_i do
22:            if |l_i.adj∩l_j.adj| ≥ ⌈Para.k1×Para.α⌉ then
23:                num ← num + 1;
24:        if num < k2 then
25:            BiGraph.remove(l_i);
26:    B ← the rest of items and users in BiGraph;
27:    return BiGraph
```

Lemma 1. *Given an (α, k_1, k_2)-extension biclique $C = (U, V)$ composes of a biclique $C_b = (L, R)$ and its α-extension $C' = (L', R')$, where $U = L \cup L'$, $V = R \cup R'$ and $L \cap L' = R \cap R' = \emptyset$, we have:*
(1) $\forall u \in U(C), u.degree \geq \alpha \times k_2$;

(2) $\forall v \in V(C), v.degree \geq \alpha \times k_1$;

Definition 4. *Given an* (α, k_1, k_2)*-extension biclique* $C = (U, V)$ *composes of a biclique* $C_b = (L, R)$ *and its* α*-extension* $C' = (L', R')$*, for any* $u \in U(C)$ *and* $u' \in U(C)$*,* u' *is a* (α, k)*-neighbor of* u *iff*

$$|u.adj \cap u'.adj| \geq \lceil k \times \alpha \rceil$$

For any $u \in U(C)$*, the set of* (α, k)*-neighbor of* u *can be defined as;*

$$S_{\alpha,k}(u, C) = \{u', s.t. |u.adj \cap u'.adj| \geq \lceil k \times \alpha \rceil\}$$

Similarly, we can define $S_{\alpha,k}(v, C)$ *for any* $v \in V(C)$.

Lemma 2. *Given an* (α, k_1, k_2)*-extension biclique* $C = (U, V)$ *composes of a biclique* $C_b = (L, R)$ *and its* α*-extension* $C' = (L', R')$*, where* $U = L \cup L'$*,* $V = R \cup R'$*,* $L \cap L' = R \cap R' = \emptyset$*, we have:*
(1) $\forall u \in U(C), |S_{\alpha,k_2}(u, C)| \geq k_1$;
(2) $\forall v \in V(C), |S_{\alpha,k_1}(u, C)| \geq k_2$;

The pseudocode of the (α, k_1, k_2)-extension biclique extraction algorithm (Line 3 in Algorithm 1) is shown in Algorithm 3 in details. The input of the algorithm includes the bipartite graph $BiGraph$ and parameters $k1, k2$ and α. Our algorithm contains two pruning strategies based on the input parameters in Line 1 to 2. The idea of *Core Pruning* is that we can directly eliminate vertices without sufficient degree (neighbors) in Line 4 to 9 based on Lemma 1. The *Square Pruning* function takes the number of the two-hop neighbors of the vertex with common adjacencies into consideration in Line 15 to 17 based on Lemma 2. We remove a vertex in $BiGraph$ and all its adjacent edges from G. Theoretically, after performing these two pruning strategies, the remaining vertices should appear in specific (α, k_1, k_2)-extension bicliques. The selection order of candidate vertices will affect the number of intermediates, thereby affecting the computational efficiency of the algorithm [3]. We select candidates in a non-decreasing order based on the size of common two-hop neighborhood in Line 14 and 21, like *reduce2Hop* in [6].

Note that the reason why our framework can restrict camouflage is that each (α, k_1, k_2)-extension biclique extracted by Algorithm 3 must contains a biclique, if the attacker wants not to be detected by the algorithm, the new edges he adds can't create a new biclique. This problem is known as Zarankiewicz problem [24] and Füredi [25] provides the best general upper bound. In other words, for every attacker who is not detected by RICD, the false clicks he can create have an upper bound.

D. Complexity Analysis

Given a bipartite graph $G = (U, V, E)$, obviously the time complexity of the naive algorithm is $O(|U||V|)$. The time complexity of the (k_1, k_2, α)-extension biclique extraction algorithm is dominated by the $CorePruning$ procedure and the $SquarePruning$ procedure. Because the $remove$ operation (in Line 6 and 9) not only needs to remove a node but all its adjacent edges, the complexity of the $CorePruning$ procedure is $O(|U| + |V| + |E|)$. The time complexity of the $SquarePruning$ procedure is dominated by the operations from Line 14 to 16 (from Line 21 to 23) and the complexity of the intersection operations (in Line 15 and 22) is $O(|V|)$ or $O(|U|)$. So the time complexity of the $SquarePruning$ procedure is $O(|U|^2|V| + |V|^2|U|)$. The total complexity of the (k_1, k_2, α)-extension biclique extraction algorithm is $O((|U| + |V|)(|V||U| + 1) + |E|)$.

VI. Experiments

A. Experimental Setup

Dataset. The real-world e-commerce user-item click data used in our experiments is the large-scale TaoBao_UI_ table (introduced in Section IV), and its statistical information is shown in Table I and Table II. To get the ground truth, we first sample 4,000 nodes from the results of the naive algorithm and ask business experts to label them as suspicious or normal. Then, we intersect these suspicious nodes with attackers already known in the dataset to produce a list of about 2,000 known abnormal nodes.

Platform. All experiments and most of the algorithms (except COPYCATCH and FRAUDAR) are conducted on the Grape [26] and MaxCompute [27]. Grape is a parallel graph engine for graph computations, and MaxCompute is a data processing platform widely used in Alibaba. The Grape has been integrated into MaxCompute so that we can use the graph mining algorithms offered by Grape API by inputting data and setting the number of workers (16 by default). Each worker represents a machine with 48 cores and 512G memory.

Baseline Methods. Since we conduct the first reported research on investigating the "Ride Item's Coattails" attack in online shopping, there is a lack of effective detection methods for this problem that we can directly compare with. Therefore, we compare our proposed detection framework RICD with three widely-used methods for community detection in many fields and two state-of-the-art graph-based anomaly detection algorithms. Meanwhile, to verify the effectiveness of our proposed techniques in the framework, we also add some simplified framework variants as our baseline methods. Details are given below:

- **Label Propagation Algorithm (LPA):** LPA is a fast algorithm for finding communities in a graph, and it does not require prior information about the communities. In the experiment, we use the LPA [18] implemented in Grape for community detection in bipartite graphs with the default input of maximal iteration round (max_round = 20) and every node is initialized with a unique label.
- **Common Neighbors Algorithm (CN):** CN algorithm is widely used to determine the closeness of a pair of nodes [28], which is similar to the idea of RICD. We use the implementation of the CN algorithm for bipartite graphs in Grape and adjust the parameter cn_threshold (minimum number of common neighbors) to 10 to make it consistent with the k_1, k_2 in RICD.
- **Louvain:** It is a heuristic method based on modularity optimization to extract the community structure of large

networks. In the experiment, we use the Louvain algorithm [29] implemented in Grape with the default input of tolerance and minimal progress threshold (tolerance = 2, min_progress = 1,000).

- **COPYCATCH:** This work is the most closely related to ours. Because there is no timestamp in the dataset, the algorithm degenerates to enumerate (near) biclique cores, which is a #P-hard problem. So we refer to the imbea [3] for the implementation and take the result of running the algorithm in a limited time (about 600 seconds) as the final output. ρ, m and n are consistent with the α, k_1 and k_2 in RICD.
- **FRAUDAR:** FRAUDAR is designed to catch fraudulent blocks in graph datasets in a camouflage-resistant way. We refer to the source code provided by the authors (http://bhooi.github.io/code/camo.zip), and re-implement it in MaxCompute for detecting multiple blocks.
- **Naive:** The naive algorithm introduced in Section V-A and we implement it in MaxCompute.
- **RICD-UI:** Comparing to RICD, it removes the whole suspicious group screening module. We use this method to illustrate the necessity of the suspicious group screening module.
- **RICD-I:** Comparing to RICD-UI, it only removes the item behavior verification step in the suspicious group screening module. Through this method, we can analyze whether user behavior check step and item behavior verification step are effective.

Evaluation Metrics. We use four metrics, namely precision, recall, F1-score and elapsed time, to evaluate our detection framework RICD and compare RICD with baseline methods. Some of the experimental results (marked with $\star$) shown below come from the experts' judgments on the sampling results for large output. We define the precision and recall rate as follows:

$$precision = \frac{Number\ of\ detected\ abnormal\ nodes}{Number\ of\ output\ abnormal\ nodes} \quad (5)$$

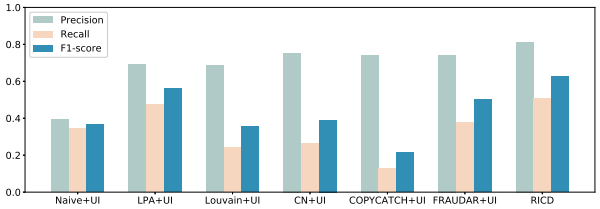
$$recall = \frac{Number\ of\ detected\ abnormal\ nodes}{Number\ of\ known\ abnormal\ nodes} \quad (6)$$

We record the end-to-end time of executing these modules as the whole elapsed time, and we need to ensure that the algorithm can output results in an acceptable time. Note that because the number of abnormal nodes in the dataset is more than the known abnormal nodes, the precision rate shown in the results will be lower than the true precision rate, but it is fair for all the algorithms in our experiments.

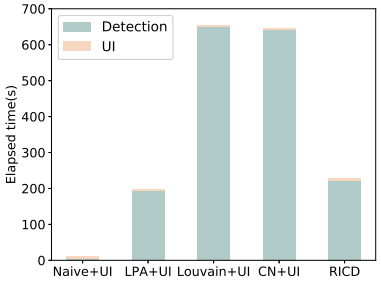
B. Comparing with Baseline Methods

In this section, we compare our framework with the baseline methods to see the effectiveness of suspicious group detection, user behavior check and item behavior verification techniques. We leave the parameters unchanged with $k_1 = 10, k_2 = 10, \alpha = 1.0, T_{hot} = 1,000$ and $T_{click} = 12$.

Effectiveness of Suspicious Group Detection. Because all baselines do not have suspicious group screening module, for the sake of fairness, we add the suspicious group screening module (**U**ser behavior check and **I**tem behavior verification, UI) to all baselines. For example, in the experiments, after the louvain algorithm outputs the partition results (into communities), we filter out communities that do not include enough users and items (less than k_1 and k_2), then perform user behavior check and item behavior verification in every remaining community. The effectiveness of suspicious group screening module can be verified by the performance gap between baselines in Fig. 8a and Fig. 8b.



(a) Comparisons of baselines by Precision, Recall and F1-score

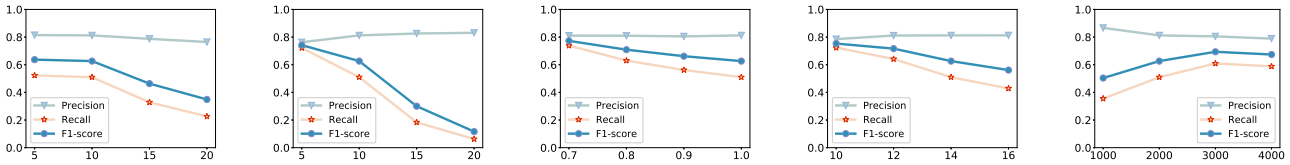


(b) Comparisons of baselines by elapsed time

Figure 8. Comparisons of baselines by Precision, Recall, F1-score, and elapsed time

In Fig. 8a, the best performer is RICD, followed by LPA, FRAUDAR, CN, Naive, Louvain, and COPYCATCH (for brevity, remove "+UI") in the order of increasingly worse performance. RICD consistently outperforms other baselines with respect to the precision, recall and F1-score. Specifically, compared with the two closest algorithms FRAUDAR and LPA, RICD achieves an improvement over 18% in the precision measure while maintaining the approximately same recall measure with LPA and achieves an improvement over 35% in the recall measure while maintaining a competitive precision measure with FRAUDAR. In general, the dense-graph-based methods can acquire a higher precision, while the recall of community-based methods is relatively high. Only RICD has advantages in both measures. The gap in recall measure between CN and RICD indicates that only considering neighbor information will cause many abnormal users or items to be erroneously undetected. The results also show that the performance of COPTCATCH is poor without timestamps, but attack detection against the dense structure can indeed improve the precision. FRAUDAR has a good performance in the dataset with camouflages, but there is still a gap between FRAUDAR and RICD in F1-score.

Fig. 8b shows the elapsed time of different baselines. Because Grape can't help accelerate the implements of COPY-



(a) Varing k_1 (b) Varing k_2 (c) Varing α (d) Varing T_{click} (e) Varing T_{hot}

Figure 9. Sensitivity analysis by Precision, Recall and F1-score

CATCH and FRAUDAR, they are not included in the comparison of the elapsed time for the sake of fairness. We can observe that compared with the UI process, the elapsed time of the detection algorithm occupies most of the time. The naive algorithm has achieved the best performance on the elapsed time for its simple logic, but its performance on other measures is not satisfactory (see Naive+UI in Fig. 8a). Compared with the CN+UI and Louvain+UI algorithms, RICD is 35% lower in the elapsed time but achieves a better F1-score. The elapsed time of LPA, a near-linear time algorithm, is slightly (less than 15%) better than that of RICD.

Effectiveness of Suspicious Group Screening. To evaluate the effectiveness of suspicious group screening module in screening suspicious items and users, we compare the precision, recall and F1-score produced by RICD-UI, RICD-I and RICD, which are presented in TableVI. We can see that, with user behavior check and item behavior verification, the precision rate of the framework gradually increases despite a decrease in recall. The results of the F1-score show that RICD has achieved the best balance between precision and recall. By comparing RICD-UI and RICD-I with RICD, we find that removing both steps will decrease the performance to some extent. More specifically, the user behavior check step can help screen out normal users who hardly click unpopular items (a small amount) and distinguish abnormal users. These results will help the item behavior verification step screen out a lot of items to produce a relatively clean click list. So compared to adding the user behavior check step (from RICD-UI to RICD-I), adding the item behavior verification step (from RICD-I to RICD) can further improve the precision.

Table VI
EFFECTIVENESS OF SUSPICIOUS GROUP SCREENING

	Precision	Recall	F1-score
RICD-UI⋆	0.03	0.82	0.06
RICD-I⋆	0.14	0.78	0.23
RICD	0.81	0.51	**0.63**

C. Sensitivity Analysis

We conduct a parameter sensitivity analysis to see how various parameters affect the performance of our "Ride Item's Coattails" attack detection framework. We vary five types of parameters in our framework:

- k_1 in Definition 2: $k_1 = 5, 10, 15, 20$.
- k_2 in Definition 2: $k_2 = 5, 10, 15, 20$.
- α in Definition 2: $\alpha = 0.7, 0.8, 0.9, 1.0$.
- abnormal click threshold: $T_{click} = 10, 12, 14, 16$.
- hot item threshold: $T_{hot} = 1,000, 2,000, 3,000, 4,000$.

The default parameter settings are $k_1 = 10, k_2 = 10, \alpha = 1.0, T_{click} = 12$ and $T_{hot} = 2,000$.

Fig. 9 shows the results for RICD. In general, the change of parameters has monotonic effect on the precision and recall measure. The only exception is the parameter T_{hot} (Fig. 9e). This is because that as T_{hot} increases from 1,000 to 4,000, a large number of abnormal items in this range are detected (positive) with the number of hot items drops, which will cause some attackers who have only clicked hot items in this range to be missed (negative). The experimental results show that setting T_{hot} to 3,000 can achieve the best recall. Significantly, the increase of k_1 and k_2 brings the opposite change to the precision measure, which indicates that in real scenarios, crowd workers tend to attack frequently (large k_2) on a small scale (small k_1).

VII. CASE STUDY

We take a "Ride Item's Coattails" attack group detected by RICD in a recent marketing campaign in TaoBao as an example to explain how our framework works in the real-life system. This attack group consists of 13 items (2 hot items and 11 target items) and 28 accounts. In Figure. 10, we show the historical traffic data of these target items. It can be found that before the start date of the marketing campaign (Day 6), the abnormal traffic of these target items had begun to increase, which means that sellers post attack missions before the campaign starts. Because of this, the normal traffic grew rapidly from Day 6 to Day 9, proving that the "Ride Item's Coattails" attack was launching. On Day 9, such a "Ride Item's Coattails" attack group was detected by RICD, and the system cleaned the false click information. These target items' traffic data was restored to the normal level (Day 10), and sellers soon removed these inferior items from their online store in TaoBao (Day 13). Based on the prediction result of the traffic model, our framework protects hundreds of thousands of users from incorrect recommendations in this campaign.

What's more, using the known affiliation of item and seller and the detection technology for account association, we find that more than 85% of these accounts have associations with each other and 11 target items are only sold by 3 sellers. All of this information provide additional strong evidence that the users and items caught are very suspicious and prove that RICD can effectively catch "Ride Item's Coattails" attack from large-scale click data.

VIII. CONCLUSIONS

In this paper, we investigate the "Ride Item's Coattails" attack in online shopping. To have a better understanding of

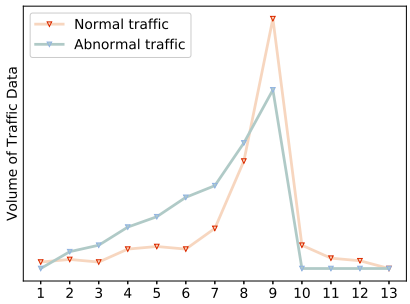


Figure 10. Historical traffic data of target items: the x-axis represents the timeline, and the y-axis represents the volume of traffic data. For security reasons, we don't display the specific event dates and the real volume of traffic data.

this type of newly-emerged malicious attack, we conduct a comprehensive analysis of the behavior of the users and items. Based on the analysis results, we propose a framework, named RICD, for detecting the "Ride Item's Coattails" attack in e-commerce applications. In RICD, the "Ride Item's Coattails" attack group is regarded as a dense near-biclique, and two suspicious node filtering strategies are proposed. The proposed dense bipartite graph structure composed of crowd workers and items are well interpretable and generic. The experimental results on the large-scale e-commerce data also show that the proposed methods are superior to the existing attack detection methods in terms of precision, recall and F1-score.

As the future work directions, it is important to study how to add an incremental data processing module to this framework so that it can be applied online to perform the "Ride Item's Coattails" attack detection in dynamic graphs. In a large-scale online shopping scenario like Taobao's "Double 11 Shopping Festival", the earlier these attacks are detected in real time, the more losses can be reduced.

ACKNOWLEDGMENT

The authors would like to thank the support from NSFC grants (No. 61972155), Zhejiang Lab under No.2019KB0AB04, Zhejiang Provincial Natural Science Foundation (LZ21F030001) and the Science and Technology Commission of Shanghai Municipality (20DZ1100300).

REFERENCES

[1] N. Su, Y. Liu, Z. Li, Y. Liu, M. Zhang, and S. Ma, "Detecting crowd-turfing" add to favorites" activities in online shopping," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1673–1682.

[2] C. Xu and J. Zhang, "Towards collusive fraud detection in online reviews," in *2015 IEEE international conference on data mining*. IEEE, 2015, pp. 1051–1056.

[3] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, and M. A. Langston, "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types," *BMC bioinformatics*, vol. 15, no. 1, p. 110, 2014.

[4] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copy-catch: stopping group attacks by spotting lockstep behavior in social networks," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 119–130.

[5] S. O. Kuznetsov, "On computing the size of a lattice and related decision problems," *Order*, vol. 18, no. 4, pp. 313–321, 2001.

[6] B. Lyu, L. Qin, X. Lin, Y. Zhang, Z. Qian, and J. Zhou, "Maximum biclique search at billion scale." *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1359–1372, 2020.

[7] L. Wang, "Near optimal solutions for maximum quasi-bicliques," *Journal of Combinatorial Optimization*, vol. 25, no. 3, pp. 481–497, 2013.

[8] D. I. Ignatov, P. Ivanova, and A. Zamaletdinova, "Mixed integer programming for searching maximum quasi-bicliques," in *International Conference on Network Analysis*. Springer, 2018, pp. 19–35.

[9] X. Liu, J. Li, and L. Wang, "Quasi-bicliques: Complexity and binding pairs," in *International Computing and Combinatorics Conference*. Springer, 2008, pp. 255–264.

[10] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sina weibo by propagation structures," in *2015 IEEE 31st international conference on data engineering*. IEEE, 2015, pp. 651–662.

[11] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, "Epidemiological modeling of news and rumors on twitter," in *Proceedings of the 7th workshop on social network mining and analysis*, 2013, pp. 1–9.

[12] S. Kumar, R. West, and J. Leskovec, "Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes," in *Proceedings of the 25th international conference on World Wide Web*, 2016, pp. 591–602.

[13] B. Horne and S. Adali, "This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 11, no. 1, 2017.

[14] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Catchsync: catching synchronized behavior in large directed graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 941–950.

[15] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 895–904.

[16] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[17] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, "Module identification in bipartite and directed networks," *Physical Review E*, vol. 76, no. 3, p. 036102, 2007.

[18] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[19] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "Eigenspokes: Surprising patterns and scalable community chipping in large graphs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 435–448.

[20] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[21] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 410–421.

[22] Wikipedia contributors, "Pareto principle — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Pareto_principle&oldid=967358099, 2020, [Online; accessed 26-August-2020].

[23] C. Yan, J. G. Burleigh, and O. Eulenstein, "Identifying optimal incomplete phylogenetic data sets from sequence databases," *Molecular phylogenetics and evolution*, vol. 35, no. 3, pp. 528–535, 2005.

[24] P. Kővári, V. T Sós, and P. Turán, "On a problem of zarankiewicz," in *Colloquium Mathematicum*, vol. 3. Polska Akademia Nauk, 1954, pp. 50–57.

[25] Z. Füredi, "An upper bound on zarankiewicz'problem," *Combinatorics, Probability and Computing*, vol. 5, no. 1, pp. 29–33, 1996.

[26] W. Fan, W. Yu, J. Xu, J. Zhou, X. Luo, Q. Yin, P. Lu, Y. Cao, and R. Xu, "Parallelizing sequential graph computations," *ACM Transactions on Database Systems (TODS)*, vol. 43, no. 4, pp. 1–39, 2018.

[27] Alibaba Group, "Maxcompute," https://www.alibabacloud.com/product/maxcompute, 2020, [Online; accessed 4-September-2020].

[28] S. Daminelli, J. M. Thomas, C. Durán, and C. V. Cannistraci, "Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks," *New Journal of Physics*, vol. 17, no. 11, p. 113037, 2015.

[29] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.