BadLink: Combining Graph and Information-Theoretical Features for Online Fraud Group Detection

Yikun Ban∗
Peking University

Xin Liu
Tsinghua University

TianYi Zhang
Tsinghua University

Ling Huang
Fintec.ai

Yitao Duan
Netease Youdao Inc.

Xue Liu
McGill University

Wei Xu
Tsinghua University

ABSTRACT
Frauds severely hurt many kinds of Internet businesses. Group-based fraud detection is a popular methodology to catch fraudsters who unavoidably exhibit synchronized behaviors. We combine both graph-based features (e.g. cluster density) and information-theoretical features (e.g. probability for the similarity) of fraud groups into two intuitive metrics. Based on these metrics, we build an extensible fraud detection framework, BadLink, to support multimodal datasets with different data types and distributions in a scalable way. Experiments on real production workload, as well as extensive comparison with existing solutions demonstrate the state-of-the-art performance of BadLink, even with sophisticated camouflage traffic.

KEYWORDS
Fraud detection, Graph clustering, Information theory

1 INTRODUCTION
Fraud is a serious threat to the modern Internet [4, 22]. People conduct fraud for a variety of reasons: from gaining publicity to conducting a credit card scam. For example, on a social network or media sharing website, people want to increase their account value by adding more followers [23]. On e-commerce websites, fraudsters register for many accounts to take advantage of the site’s new user promotions.

Fraud detection is an essential topic in both computer security and data mining communities. While there are many kinds of methods to catch frauds on the Internet, we focus on analyzing the user logs (eg. the click stream) to distinguish fraudulent users from legit ones. The logs are multi-modal and may contain two kinds of data, 1) the user profile, such as geographical location, age, phone number and gender, and 2) user actions, such as registration, login, page viewing and purchasing, as well as social actions such as following other users. Depending on the application, most logs only contain a subset of these properties (i.e. modes).

There are basically two types of methods to detect frauds from the logs. The first kind is to model single-user actions and detect abnormal action sequences either with rule-based or machine-learning-based methods[1, 7]. There are two drawbacks of these approaches: 1) we need to wait for multiple actions, and the damage may have occurred at the time of detection, and 2) the fraudsters change behavior to avoid detection, and thus the sequence patterns do not last.

The second and more popular type of detection techniques focuses on detecting group behavior across multiple users [13, 14, 28, 29]. The basic assumption is that normal users act quite differently and fraudsters have unusually synchronized behavior. There are two reasons for this synchronized behavior. First, it takes a full black-market to supply resources like proxy servers, phone numbers, fake accounts and fraud scheme designs. To reduce the cost of these resources, fraudsters multiplex them, and thus many frauds share certain properties like source IP addresses or phone numbers. Secondly, to achieve the “economy of scale”, fraudsters often use many (fake) accounts to conduct the same fraud (e.g., paid followers), resulting in synchronized actions among these accounts.

Group-based fraud detection is challenging. First, as fraud schemes change quickly over time, people do not usually have many labeled fraud cases for model training. Plus, fraudsters often perform camouflage actions to hide their similarities. Thus, we can only rely on unsupervised methods. Second, logs are intrinsically multimodal, with different data types for each mode. Many existing methods encode these data types into feature vectors. Given many discrete data types (e.g., IP addresses and phone numbers), the typical one-hot encoding results in very high dimensional and sparse feature vectors, making it hard for unsupervised algorithms. Thus we cannot use regular clustering algorithms to analyze the data.

Third, different modes make different contributions to detecting fraud group, since different fraudster groups exhibit synchronized behavior on different subsets of modes. However, many existing works [14, 26] regard each mode equally or rely on heuristics to manually select and weight them. For example, it is more suspicious that two users use the same IP address than they are from the same city. Forth, different modes exhibit drastically different distributions across the dataset. For example, IP addresses usually have a Poisson distribution [14] while items purchased follow a
power-law distribution, further complicating the detection [9]. Last but not least, as false detection significantly affects user experience, it is vital to make the detection results explainable to operators so that they can make informed decisions on a fraud case.

We design and implement BadLink to address these challenges. BadLink is a graph-based fraud group detection framework. BadLink captures the similarities between two users across multiple modes using the information, which captures how unusual they are similar on a particular value (e.g. an IP address). We combine this information property with graph-level properties, such as the subgraph size and its density, to measure the suspiciousness of a user group.

We have deployed BadLink onto a proprietary production system and achieved state-of-the-art accuracy. All algorithms are straightforward to implement on Apache Spark[30] and provide production scalability and reliability. We perform extensive evaluations using a production dataset, an open dataset, as well as two synthetic datasets (as used in existing work) to show BadLink’s superior performance and applicability to different types of data.

In summary, our main contributions include:
1) [Metrics] As the core of BadLink, we propose two novel metrics, complicity score and fraud density score, capturing both graph and information-theoretical properties of fraud user groups. Both metrics distinguish fraud groups from normal users significantly better than existing metrics, especially when fraud groups exhibit synchronized behavior on different subsets of modes.
2) [Automatic mode weighting] Our information-theoretical metrics give different modes different weights based on their distributional properties. They capture how likely a group of users synchronize on a particular value. This provides an automatic way to identity those key modes there fraudster multiplex or share resources.
3) [Scalability] Our algorithm runs in near-linear time as number of edges on typical real-world data. We design and implement BadLink as a general and scalable system for fraud detection. All of our algorithms are embarrassingly parallel and run on scalable Spark framework.
4) [Real-world applications] Using four datasets with different scenarios, from social network to user registration, we demonstrate that BadLink is generally applicable, and achieves state-of-the-art performance.

2 RELATED WORK
In general, there are two kinds of approaches for fraud detection. The first kind is based on mining for individual behavior or profile patterns for fraudulent users. [8, 21] build classifiers to identify fraudulent users with known malicious features, such as time and text, but the supervised approach prevents them from capturing new patterns. [12, 18, 24] find users that have different feature distributions from the majority, and thus are unsupervised. To build such patterns, the methods need large amount of information about a single user, which is not always available, especially in scenarios like our new user registrations.

Other approaches work on detecting similar or clustering patterns over different users. People have adopted two methods to find fraud clusters: feature-vector-based and graph-based. For feature-vector based approaches, singular-value decomposition (SVD) and clustering are the most popular algorithms. [15, 23] use the top eigenvectors from SVD to find the abnormal users. These methods often leverage extensive feature engineering. [28] uses similarity between users’ clicks as features, and use a non-parametric clustering algorithm to find similar users.

Many projects focus on detecting unexpectedly dense blocks on social graphs for follower-followee or customer-product connections, using methods from the network science and graph algorithms such as [2, 19]. Average degree [6] and K-cores [25] are popular methods to find such blocks. [31] pioneered the method of dense subgraph based detection for fake accounts, but it only focuses on a single mode: IP addresses. [13] extends the average degree metric of [6] and tries to handle fraud users’ camouflage by edge weighting. [29] adds the time dimension to consider temporal clustering behavior.

[14, 26] also focus on detecting fraud across multimodes. However, they directly detect dense blocks on the tensor formed by raw logs, ignoring varying distributions and different contributions of modes for fraud detection. In comparison, BadLink captures information across multi-modes, as well as the graph features, which not only significantly improves the performance but also simplifies the algorithm design.

3 BACKGROUND
We introduce the black market and analyze the economics behind frauds. Furthermore, we infer the reason of clustering behavior of fraudsters.

3.1 Frauds and the Black Market
As most frauds are designed for financial gains, it is essential to understand the economics behind frauds. Only when the fraudsters’ benefit is bigger than their cost do they want to perform a scam. To maximize profits, fraudsters have to share/multiplex at these different resources (e.g., phone numbers, accounts, and followers) over multiple frauds.

Shared resources and common tasks are more fundamental and much harder for fraudsters to avoid than other behavior features, and thus become the key to fraud detection. Considering the unavoidable resource sharing, fraudulent users exhibit synchronized behavior compared to the legit users. For example, [27] finds many phone number reuse cases, and [3] observes that the IP addresses of many spam proxies and scam hosts fall into a few uniform ranges.

There is another reason for behavior synchronization. Fraudsters use a large number of accounts to complete a few tasks during a short time period, be it following a paying user, or setting up many new accounts. This is because the financial gain for a single task is too small, and only many of them can cover fraudsters’ fixed initial cost (e.g., to develop the software to conduct fraud).

3.2 Key Observations
Fraudsters exhibit evident clustering behavior because of resource multiplexing and the common tasks. We summarize three critical observations in these clusters that directly lead to BadLink design.

1) Similarity does not make a user suspicious, but unusual similarity does. Two users can be similar in many modes, but the
similarities on some modes, even certain values, are more suspicious than others. Intuitively, it might be suspicious if two users share an IP address or following the same random “nobody”. However, it is not so suspicious if they have a common gender, city, or following the same celebrity. In other words, two users are likely to be fraud complicities not because they are similar, but because the probability that they are identical in a mode, or at a certain value, is quite low. We call the users with such similar behavior “complicities.”

2) A fraud group contains an unusually large number of complicities. Fraudsters have to conduct the same behavior many times to achieve their economy of scale. Thus, we expect to find many pair-wise complicities among fraudsters. Much literature has pointed out that the large cluster size is a crucial indicator of frauds [5, 29]. Intuitively, while it is natural for a few family members to share a phone number, it is highly suspicious when dozens of users share one. In our dataset, we can see the fraud group sizes range from 70 to 667 while it is rare to see a normal group with 20 or more users.

3) A group is suspicious not because it has many edges, but because it contains many similar unusual edges. Fraudsters usually dedicate a number of fraudulent accounts for the same job, and thus it is likely that users within the same fraud group are similar in an unusual but consistent way to each other. In contrast, even if a real legit user is similar to some fraudsters (e.g., on the same IP subnet with a fraud group), he or she is unlikely to be similar on many modes to fraud users. To avoid detection, fraudsters often generate camouflage activities, such as letting each fake user follow some random people to decrease the consistency [13]. A good fraud detection algorithm needs to be camouflage resistant.

4 METHODS
Like many fraud detection systems, BadLink takes as input a log containing a number of user profiles and their activities. Our goal to differentiate fraud groups from the legit users in an unsupervised way. BadLink focuses only on detecting group frauds.

4.1 Problem Formulation
BadLink is a graph-based algorithm. The entire dataset with K modes forms a weighted undirected graph \( G = (V, E) \). We model each user (a tensor with K modes) as a node, and we connect two users with an edge if there is some similarity between them, e.g., sharing the same value on a mode. Formally, \( u_i \in V \) denotes a user and \( u^k_i \) denotes the k-th mode of \( u_i \). Note that \( u^k_i \) may contain a collection of values, and we use \( u^k_i[m] \) to represent the m-th value in the collection. The collection of a user is useful to capture multiple actions, e.g. if a user purchases multiple items, we can use each \( u^k_i[m] \) to represent each single item. Edge \( e_{ij} \) between \( u_i \) and \( u_j \) has an edge weight, \( S_{ij} \), aka. the complicity score we will introduce in Section 4.3. BadLink focuses on finding subgraphs that represent fraud clusters. We denote a single cluster as \( G = (V,E) \), where \( G \subseteq G \).

Given the subgraph \( G \), we design its fraud density score \( \mathcal{F}_G \) so that members in groups with higher \( \mathcal{F}_G \) are more likely to be fraudulent. Thus, the goal of BadLink becomes computing a list of fraud clusters ranked by \( \mathcal{F}_G \), given the graph G.

Note that although the formulation is quite general, the real challenge is to design the two scores \( S_{ij} \) and \( \mathcal{F}_G \), which will elaborate in the rest of this section.

4.2 Workflow Overview
BadLink is a filtering-based workflow. It works in multiple stages and in each stage we filter out some less suspicious nodes and edges. Fig. 1 provides an overview of the flow. We implement each stage as a separate Spark [30] job. In summary, the stages include:

**Initialization and removing edges with low complicity scores.**
We scan the logs and construct the initial graph. During the construction, we minimize the number of edges we need to consider further. We then compute the edge weights (i.e., complicity scores \( S_{ij} \)) for each edge, and dynamically determine a threshold to filter out less suspicious edges. In other words, we only consider the very unusual similarities.

**Computing and filtering on subgraph fraud scores.** We find all connected subgraphs and compute a fraud density score for each. Then we rank the subgraphs based on the score and delete the ones with low scores.

**Refining for subgraph quality.** After the two steps above, the remaining subgraphs exhibits evidently synchronized behavior and

![Diagram](image_url)
thus are highly suspicious. However, there are still some individual legit users included (e.g., accidentally sharing an IP subnet). We refine the clusters by finding the subset of nodes within each subgraph that maximizes the fraud density score, details in Sec 4.4.

4.3 Computing Pairwise Complicity Score

Complicity score. First, we define a metric to capture how unusually similar a user pair is. Consider the probability that mode $k$ takes value $x$ is $p^k(x)$. We define a metric $I^k_{i,j}(x)$, to capture the unusualness of $u_i$ and $u_j$ having the same value $x$ on mode $k$, or in other words, the information of the event that $u_i$ and $u_j$ both take value $x$, we define

$$I^k_{i,j}(x) = \begin{cases} \log \frac{1}{p^k(x)} & \text{if } \exists a, b \text{ s.t. } u^k_i[a] = u^k_j[b] = x \\ 0 & \text{otherwise} \end{cases},$$

(1)

where $\equiv$ is the customizable equals-to operator that defaults to the natural equals-to function for the data type. We will discuss more about the operator below.

Intuitively, if they do not take the same value $x$, which we expect normal, we get zero information. Otherwise, we get some information. Also, the information we get for two users to share the same value $x$ is related to the overall probability of that value. For example, if two users both follow Donald Trump on Twitter, there is not much surprise, but if they both follow a nobody, they are more suspicious.

Thus, we define the complicity score between $u_i$ and $u_j$ on mode $k$ by summing up the information across all possible $x$.

$$I_{i,j}^k = \sum_X I^k_{i,j}(x)$$

(2)

To compute the complicity score across all $K$ modes, following typical information property, we can define

$$S_{i,j} = \sum_{k=1}^K I_{i,j}^k$$

(3)

Intuitively, the higher $S_{i,j}$ is, the more similar $u_i$ is to $u_j$. In practice, the $S$-score shows large variance. For example, user pairs sharing IP subnet and device ID get high $S$-score. In contrast, a normal user is likely to share these values with no one, and thus $S_{i,j}$ is close to zero.

Extending the complicity score. We can extend the complicity score to accommodate different data types and distributions.

First, note that we use $\equiv$ to denote a customizable “equals-to” definition for each mode. For example, people usually define two users sharing the same IP subnet if the first 24 bits of their IP addresses are the same. As another example, for timestamps, people usually treat two timestamps within the $\Delta$ range to be the same.

Second, it is tricky to determine $p^k(x)$, as we do not always know the distribution of mode $k$. In this case, for modes with discrete values (e.g., categorical), we assume a uniform distribution and simply set $p^k(x) = 1/q^k$ for all $x$, where $q^k$ is the number of unique values of mode $k$. This approximation works well for many fraud-related properties such as IP subnets and phone numbers, which usually has a Poisson distribution [14].

However, the uniform assumption works poorly for low-entropy distributions, such as the long-tail distribution, common in modes like items purchased or users followed. Low entropy means many users behave similarly anyways, independent of frauds. Intuitively for such distributions, there is no surprise to follow a celebrity (head of the distribution), but much information if they both follow someone at the tail. For example, 20% of users get more than 80% of the follows in the online social network. The dense subgraphs between the celebrities and their fans are very unlikely fraudulent. If mode $k$ has a long tail distribution, its entropy is very low. For example, the entropy of uniform distribution over 50 values is 3.91, but the entropy of a long-tail distribution with 90% probabilities centered in one value is only 0.71. We compute the empirical distribution (i.e., histogram), and use it to calculate $p^k(x)$, when we find the values in a mode has low entropy. We also provide an interface so that users can define their own $p^k(x)$ function.

Optimization on computing the complicity score. In theory, in a graph with $|V|$ nodes, there are $O(|V|^2)$ edges, and thus naïvely it takes $O(K|V|^2)$ time for graph initialization and traversal. However, as not many users are similar to each other, the graph is quite sparse. We use the key-value approach. The key corresponds to a value $x$ on the $k$-th mode and the value denotes the set of users having $x$ as $N(x)$. Since any user shares $x$ with other users in $N(x)$, we increase the value of $I^k_{i,j}(x)$ where $\forall (u_i, u_j) \in N(x)$. Thus we parallel to compute all key-value pairs across $K$ modes by traversing the overall event records in the dataset. It takes $O(C + |E|)$ to build the graph $G$. We apply $S$-score on several real-world datasets in Table 3 and the edge densities of $G$ are all lower than 0.0028, which indicates the $G$ is quite sparse.

Furthermore, we can delete the values with high in-degree and all events caused by them in some cases before the graph initialization, since they are useless to fraud detection. For example, in the bipartite graph of followers and follows, a node of high in-degree may be a famous celebrity or politician, so we delete the node and all its edges. The same applies to popular products on Amazon. This filtering significantly reduce the computation cost of building $G$.

Determining the threshold to remove normal edges. We iterate through all edges in the graph, and compute their $S$-score. We remove the edge if $S_{i,j} < \theta$. We determine $\theta$ with

$$\theta = \frac{C}{|V|} \sum_{k=1}^K \frac{I(k)}{K},$$

(4)

and

$$I(k) = \frac{\sum_{u_i, u_j \in E} I^k_{i,j}}{w}$$

where $w$ is the total number of pair of $u_i$ and $u_j$ in which $I^k_{i,j} > 0$.

In event records, fraud groups we assumed have synchronized behavior on one mode at least, which indicates they obtain information on the mode, whereas normal users do not. Thus, we determine $\theta$ as the product of two terms: the average number of events per user, $C/|V|$, and the average information per user on a mode, $\sum_{k=1}^K I(k)$. If the degree of a node goes to zero, we remove the node too. Note that tuning the value of $\theta$ may lead to further improvement of performance in different types of dataset.

Finding candidate fraud clusters. As we have filtered out many edges, it is likely that graph $G$ is partitioned into multiple connected components. Instead of using clustering algorithms like [28], we
use these connected subgraphs as candidates for fraud groups. It is a reasonable choice not only because the connected property indicates similarity, but also because it is efficient to compute using algorithms like [17].

We only keep large-enough connected components and treat all small ones as normal, according to observation (2) discussed above. We determine a magnitude threshold

$$\psi = \sum_{k}^{K} \frac{|V|}{q^k}$$

where $$\psi$$ is the sum of the average number of users having similar value over all $$K$$ modes. Thus we understand $$\psi$$ as the average size of components. We select all the components with size bigger than $$\psi$$ as candidate fraud groups.

**Computing the fraud density scores.** We compute the fraud density score of a candidate subgraph $$G$$ by dividing the sum of its edge weights by its number of nodes:

$$\mathcal{F}_G = \frac{\sum_{u_i, u_j \in V} S_{i,j}}{|V|}.$$  \hspace{1cm} (6)

The $$\mathcal{F}_G$$ score captures three metrics of the cluster $$G$$: edge weight $$S$$, cluster size $$|V|$$ and edge density $$\rho = \frac{|E|}{|V|^2}$$. The first encodes the information-theoretical similarities and the latter two encode graph-based clustering features.

Specifically, the fraud density score satisfies the following three conditions (corresponding to the “Axioms” in [14]), if for each case, we keep all other parameters the same: 1) Complicity: edges with higher weights result in a higher $$\mathcal{F}_G$$; 2) Size: a larger cluster has a higher $$\mathcal{F}_G$$; and 3) Consistency: the denser the cluster is (dense as in having higher total $$S$$-score), the higher $$\mathcal{F}_G$$ is. Appendix A shows the formal analysis.

In contrast, naive metrics with either one side of graph-features or information do not meet all three conditions above. For example, the edge density $$\rho(G)$$ is not a good metric because it does not satisfy condition 1). The numerator of Eq. 6, $$\sum_{u_i, u_j \in V} S_{i,j}$$, that emphasis on the information only, does not satisfy condition 3).

### 4.4 Improving Group Quality

Depending on the data distribution and the definition of mode equality operator $$=$$, it is possible that some legit users have some edges with non-negligible $$S$$ connecting to one or more fraud users incidentally, as we have discussed in Section 3.2

Based on observation (3), we can eliminate false positives by increasing the consistency across all members of the group. As fraud users share common resources and tasks, they are more similar to each other than an incidentally included legit user.

For group $$G$$, increasing $$\mathcal{F}_G$$ implies increasing group size and consistency across all nodes. Thus, our problem reduces to finding the subgraph $$G_0$$ of $$G$$ that maximizes the $$\mathcal{F}_{G_0}$$.

It is a typical densest subgraph problem that can be solved using flow network[10]. However, it is difficult to scale to a dataset with millions of nodes. Therefore, we propose a greedy algorithm that runs in near-linear time, similar to the ideas in [6, 13]. Algorithm 1 shows the pseudo code of the algorithm.

Intuitively, removing a node $$u_i$$ (and all associated edges) decreases the numerator of Eq. 6 (that we denote as $$t$$) by $$t_u =$$

**Algorithm 1** Algorithm to find a subgraph that maximizes $$\mathcal{F}_G$$

**Require:** subgraph $$G = (V, E)$$; metric $$\mathcal{F}$$

1. Set $$X_0 = \emptyset$$, $$n = |V|$$

2. for $$t = 1 \ldots n$$ do

   $$u^* = \max_{u \in V \setminus X_{t-1}} \mathcal{F}_{G \setminus X_{t-1} \setminus u}$$

   Update all $$t_u$$ of neighbors of $$u^*$$

   $$X_t \leftarrow X_{t-1} + u^*$$

3. return $$\max_{X_i \in X_0 \ldots X_n} \mathcal{F}_{G \setminus X_t}$$

### 4.5 Analysis

**Complexity.** In graph initialization, it takes $$O(C + |E|)$$ to build $$G$$ based on the optimization in Sec 4.3. The cost for finding connected components in $$G$$ is $$O(|E|)$$, and for each group, it takes $$O(|E|)$$ to compute the fraud density score. To improve the quality of the fraud group using Algorithm 1, it takes $$O(|E| \log |V|)$$ time per fraud group, as it is likely that $$|E| << |E|$$, $$|V|$$, and thus the cost is small.

In summary, the overall complexity of BadLink is near-linear to the number of edges in the graph. Even more importantly, we only use algorithms that are “embarrassingly parallel”. Thus we can easily implement them on frameworks such as Apache Spark [30].

**The information between user pairs.** We use total information across all modes and all values to serve as the complicity score for a user pair. The information is a good metric on capturing unusual similarities. We have the following observations. First, we randomly sample 50 normal users and 50 fraud users from a single group, using our real datasets with ground truth labels, and plot their pair-wise complicity score as a heatmap in Fig. 2. Users 1-50 in the

![Figure 2: a subgraph in G formed by sampled 50 normal users (1-50) and fraud users (51-100) from a single group in our real-world dataset: (1) S-score captures the usually similarity of fraud users well; (2) The fraud group forms a region that is quite dense and dark in the Heatmap.](image)
We use four datasets to evaluate BadLink performance. The first two are real-world datasets. As the gold labels are costly to obtain, even in our production environment, we follow the evaluation methodology of [13, 14] and introduce two synthetic datasets. 

[Registration] is from our production system. It contains over 26,075 log records of a single user activity type: new account registration. It contains many features such as IP address, phone number, timestamp and the time spent to complete the registration form. For evaluation purposes only, we obtain “gold labels” by observing these users for a few months, and label those who have conducted frauds during this period as positive, while all others as negative. The dataset contains 16,154 good users and 9,921 bad users. Note that we can only use these labels for evaluation but not training, as our goal is to make the detection when the registration event happens, rather than a few months later.

[Twitter] is an open dataset that dates back to 2010[11]. We sample about 40 million events from the dataset. It is not designed for fraud detection and there are no gold labels about frauds.

[Reviews] is a real dataset containing 1835 users, 2,928 products and nearly 20,000 product reviews from Amazon [16]. All the users and reviews are legit. We then inject fraud users and reviews into the dataset by the same method as [13], and we use the modified datasets for evaluation. We inject 200 fraud users, each making fake reviews on 5, 10, 15 or 20 products (obviously, the fewer reviews, the more difficult to detect). Also, we reproduce the four types of camouflage: 1) no camouflage; 2) random camouflage by reviewing random products; 3) biased camouflage by reviewing random products with probability proportional to original review numbers, and 4) hijacked camouflage by hijacking normal users’ account into a fraud account (i.e. contains all normal user activities but adding the fraudulent reviews.). Considering the combination of the number of fake reviews per fraud user, and the camouflage patterns, we have 16 versions of the datasets for evaluation.

[CrossSpot] is a dataset we synthesis using the same method as [14]. First, we generate six 6-mode datasets by Erdös–Rényi-Poisson model [14] with 1,000 legit users and 10,000 events each. Values of all modes are uniformly randomly distributed over 500 unique values. We generate five fraud groups of 50 users and 200 fraud events (with values concentrated on three values each). We inject these fraud groups in different ways to the six datasets so that dataset 1 contains fraud groups with synchronized behavior on only one mode (a different one for each set), and dataset 2 contains fraud groups with synchronized behavior on two of the modes, and so on. Obviously, the fewer similar modes there are, the harder the detection is.

Evaluation metrics. For datasets with gold labels, including the synthetic datasets, we evaluate them with standard metrics such as precision, recall and the F1 score (the harmonic mean of precision and recall). For the [Twitter] dataset, we only report qualitative findings due to the lack of ground truth labels.

Existing algorithms in comparison. We compare with the following three state-of-the-art fraud detection algorithms.

CrossSpot [14] proposes a suspiciousness metric that its algorithm maximizes to find a dense block in multi-mode tensor.

Fraudar [13] proposes a reweighting method of the 2-mode graph to handle camouflage and also maximizes a weighted average degree.
To compare with these approaches, we do our best to reproduce their results, including the dataset in their evaluation. Note that as the dataset used in Fraudar is no longer available, we use the [Reviews] with same fraud injection instead.

All four algorithms have configuration parameters to tune, and we try different values. For CrossSpot, we try random initialization seeds of 400, 500, and 600. We find little difference with these configurations though. Fraudar only reports the single most suspicious group on each run. M-zoom gives the ranking of suspiciousness of fraud group. To make it work on multiple groups, we iteratively delete edges in the largest group and run Fraudar and M-zoom again. We run it for 8, 10, 12 times, and report the best results.

Not all three algorithms support all the datasets. For example, Fraudar only support 2-mode datasets. Our approach, in contrast, works on all these datasets. Thus we only provide comparison where applicable.

5.2 Real-world dataset performance

Performance on [Registration] dataset. Using the gold labels in the [Registration] dataset, we compare BadLink with CrossSpot and M-zoom. For examining influence of noisy modes to these methods, We design two experiments. In the first experiment, we use two key modes from the dataset, IP subnet and 7-digit prefix of phone. In the second experiment, we add three noisy modes sequentially with no evident synchronized behavior, timestamp, city of phone, and city of IP.

Fig. 3 (a) summarizes the precision and recall we get in each method in the first experiment with 2 modes. Our key observations include: (1) BadLink achieves the overall best accuracy and precision. In fact, BadLink achieves the best-F1 of 0.9823 while M-zoom’s is about 0.8586. (2) CrossSpot performs poorly on the dataset, mainly because it classifies good users into bad groups when bad groups form sparse blocks compared to normal groups in the tensor formed by raw logs.

To understand the reason for the differences, we manually inspect the detection results. For example, BadLink detects a large cluster containing 667 users whose phone prefixes are the same. As another example, we also find a 51-member group sharing a single IP subnet and 16 unique phones. In addition to the gold labels, we are convinced that they are real frauds because we notice that it takes longer than 30 minutes to complete the registration form, much longer than the average time of 9 minutes for normal users. Consulting the operators, we understand that it is because they are waiting for the fake cell phone short message (SMS) verification from a black market service provider.

We inspect M-zoom’s results and find a group with 75 good users and 125 bad users. This is because many good users are from the same city as fraud users. These false positives indicate the drawback of graph-feature-only approach: it does not consider the probability differences for sharing a city vs IP address. The situation of CrossSpot is very similar with M-zoom.

Fig. 3 (b) shows varying of F1-score of these approaches as the number of modes increase. (1) BadLink achieves the similar perfect accuracy and precision regardless of modes, because it minimizes the effect of noise modes leveraging the information and filtering. (2) M-zoom’s F1-score swing evidently, since treating each mode equally leads to negative impact of noisy modes. CrossSpot still has poor performance despite the increasing of modes.

Performance on [Twitter] dataset. BadLink reports 157 fraud groups with 213, 627 frauds in the dataset. Lacking scalable implementations, the other three techniques fail to produce any results on this dataset with 40 million records after running for one day.

Without gold labels, we manually inspect the detection results. One group contains 3, 246 users following a common set of 5, 596 users. None of these followers has over 1, 500 followers except for the 3, 246 followers from the group, and many of them only have a couple of hundred other followers. This is a very suspicious behavior of paid “zombie fans”, common in the black market.

5.3 Synthetic data comparison

Comparing to CrossSpot and Fraudar on [Reviews]. Using the four different numbers of reviews per fraud user and the four camouflage patterns, we compare BadLink performance with Fraudar and CrossSpot. Fig. 4 shows the F1-score comparison.

First, it is challenging for all methods when each fraud user only makes five reviews. BadLink does significantly better than the other two. Taking biased camouflage case as an example, BadLink achieves a recall of 0.93 but with a significant number of false positives (with precision at only 0.74). However, it is still much
better than the other two whose precisions are less than 0.10. This is because the S-score helps tell apart the injected fake reviews (as several fraud users reviewed a single one), while neither CrossSpot nor Fraudar captures this similarity.

Second, with 15 reviews per fraud user, BadLink achieves the best F1-score under all camouflage cases. For the random camouflage case, BadLink can catch all frauds with a single false positive. In comparison, Fraudar achieves a precision of 0.82. This is because the graph-only feature, edge density ρ, is very sensitive to the randomly inserted camouflage edges, while our S-score considers the probability of each product actually getting a review, in addition to the edge density and thus is much more reliable.

Third, BadLink is resilient to different camouflage patterns. In contrast, Fraudar works on the edge density of nodes. When the number of camouflage makes the density of fraud users similar to that of the legit users, Fraudar fails to work. Our S-score is different from the edge density by considering the information (i.e. surprises) on the edges.

Fourth, CrossSpot performs consistently poorly because it tries to find a relatively dense block (subgraph) on a single mode, but camouflage destroys the dense block pattern.

Comparing to CrossSpot on [CrossSpot] dataset. The six versions of the [CrossSpot] datasets differ by how many modes fraud users have synchronized behavior on. Fig. 5 shows the comparison between BadLink and CrossSpot in precision and recall.

CrossSpot works poorly if there are less than four modes with significant similarities, while BadLink works well on all numbers of modes, which demonstrates the robustness of BadLink for mode selection. This is because when CrossSpot evaluates the edge density by treating each mode equally. In contrast, we combine the information across all modes into a coherent S-score.

5.4 Impact of each optimization.

Allowing custom distribution function $p^k(x)$. An important feature of BadLink is to automatically detect modes with low-entropy distributions, and allow using custom $p^k(x)$. The products in the [Reviews] dataset have such a long-tail distribution. In fact,
parameter settings vs. the final precision and recall. Threshold $\theta$, controlling the minimal $S$-score we consider suspicious, provides a trade-off between recall and precision. A larger $\theta$ allows more precision but less recall. We automatically set these parameters and the choice is marked in Fig.7. The parameter choice balances precision and recall, and in fact we can achieve an F1-score of 0.86.

**Scalability.** We implement BadLink on several real-world datasets (details in Table 3). The edge densities of $G$ are quite low across all datasets, which indicates that both time and space complexities are near-linear to the number of nodes in the graph. Fig.8 confirms the near-linear scaling of running time of BadLink using Amazon-Food dataset. Varying the number of edges is achieved by subsampling rating actions in the dataset.

6 CONCLUSION AND FUTURE WORK

Group-based fraud detection is a promising methodology to catch frauds on the Internet because 1) it does not require a long activity history for a single user; and 2) it is difficult for fraudsters to avoid due to their economic constraints. Unfortunately, existing work does not cover the entire picture of a fraud group: they either focus on the grouping feature based on graph features like edge density, or probability-based features, but not both. To our knowledge, we are the first to combine these features into a single set of metrics: the complicity score and fraud density score. Both scores allow customization to accommodate different data types and data distributions. Even better, algorithms built around these metrics only use localized graph features, and thus scale easily on modern big data frameworks. We have applied BadLink to a real production dataset and achieve state-of-the-art results comparing to other existing approaches.

As future work, in addition to applying BadLink to more production scenarios, we will explore ways to incorporate rules from domain experts into the system, further improving accuracy. We have applied BadLink to a production dataset. Varying the number of edges is achieved by subsampling near-linear to the number of nodes in the graph. Fig 8 confirms the scaling.

REFERENCES

[1] Melih Abdullahyoglu, Melih Abdullahyoglu, Melih Abdullahyoglu, and Melih Abdullahyoglu. 2017. HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network. In ACM SIGKDD. 1507–1515.

[2] Reid Andersen. 2008. A local algorithm for finding dense subgraphs. 1083–1089.

[3] David S Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M Voelker. 2007. Spamscatter: characterizing internet spam hosting infrastructure. In USENIX Security Symposium on USENIX Security Symposium. 1132–1141.

[4] Richard J Bolton and David J. Hand. 2002. Statistical Fraud Detection: A Review. Statist. Sci. 17, 3 (2002), 235–249.

[5] Qiang Cao, Christopher Palow, Christopher Palow, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In ACM CCS 477–488.

[6] Moses Charikar. 2000. Greedy Approximation Algorithms for Finding Dense Components in a Graph. Springer Berlin Heidelberg. 84–95 pages.

[7] Naveen Du, Chunqiu Lin, Wenjun Zhou, Zhenshuai Heu, and Hui Xiong. 2016. Catch Me If You Can: Detecting Pickpocket Suspects from Large-Scale Transit Records. In ACM SIGKDD. 87–96.

[8] Manuel Egele, Gianluca Stranghini, Christopher Kruegel, and Giovanni Vigna. 2017. Towards Detecting Compromised Accounts on Social Networks. IEEE TDFSC 14, 4 (2017), 447–460.

[9] Linton C Freeman. 2010. The Development of Social Network Analysis - with an Emphasis on Recent Events. (2010).

[10] Giorgio Gallo. 1989. A Fast Parametric Maximum Flow Algorithm and Applications. Siam Journal on Computing 18, 1 (1989), 30–55.

[11] Kwak Haewoon, Lee Changhyun, Park Hosung, and Moon Sue. 2010. What is Twitter, a Social Network or a News Media? http://an.kaist.ac.kr/traces/WWW2010.html.

[12] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Gajjinejmann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. [n. d.]. BIRDNEST: Bayesian Inference for Ratings-Fraud Detection. 495–503.

[13] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. FRAUDAR: Bounding Graph Fraud in the Face of Camouflage. In ACM SIGKDD. 895–904.

[14] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2016. Spotting Suspicious Behaviors in Multimodal Data: A General Metric and Algorithms. IEEE TKDE 28, 8 (2016), 2187–2200.

[15] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring looketp behavior from connectivity pattern in large graphs. Knowledge & Information Systems 48, 2 (2016), 399–428.

[16] McAuley Julian. [n. d.]. Amazon product data. http://jmcauley.ucsd.edu/data/amazon/.

[17] U Kang, Charalampos E Tzourakakis, and Christos Faloutsos. 2011. PEGASUS: mining peta-scale graphs. Knowledge & Information Systems 27, 2 (2011), 303–325.

[18] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2017. FairJudge: Trustworthy User Prediction in Rating Platforms. CoRR (2017).

[19] Victor E. Lee, Ning Ruan, Ruoming Jin, and Chiu Aggarwala. 2010. A Survey of Algorithms for Dense Subgraph Discovery. Springer US, Boston, MA, 303–336.

[20] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. InSIGchi Conference on Human Factors in Computing Systems. 1361–1370.

[21] Karan Mangla, Karan Mangla, and Karan Mangla. 2011. Facebook immune system. In The Workshop on Social Network Systems. 8.

[22] U.S. Department of Justice Federal Bureau of Investigation. 2015. 2015 Internet Crime Report. https://pdf.ic3.gov/2015/IC3Report.pdf.

[23] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. 2014. Spotting Suspicious Link Behavior with $fBox$: An Adversarial Perspective. In IEEE ICDM. 959–964.

[24] Neil Shah, Alex Beutel, Bryan Hooi, Leman Akoglu, Stephan Gunnemann, Disha Makhija, Mohit Kumar, and Christos Faloutsos. 2017. EdgeCentric: Anomaly Detection in Edge-Attributed Networks. In IEEE ICDM. 327–334.

[25] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2017. Patterns and anomalies in $k$-cores of real-world graphs with applications. Knowledge & Information Systems 3 (2017), 1–34.

[26] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-Zoom: Fast Dense-Block Detection in Tensors with Quality Guarantees. In ECML PKDD. 264–290.

[27] Kurt Thomas, Dmitry Latskov, Elke Burzstein, Tadek Pietraszok, Chris Grier, and Damon McCoy. 2014. Dialing back abuse on phone verified accounts: In Proceedings of the 2014 ACM SIGKDD. ACM, 465–476.

[28] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and Tong Zhang. 2015. Crowd Fraud Detection in Internet Advertising. In Proceedings of the 24th International Conference on WWW. 1100–1110.

[29] WanHong Xu, WanHong Xu, Christopher Palow, Christopher Palow, and Christos Faloutsos. 2013. CopyCatch: stopping group attacks by spotting looketp behavior in social networks. In International Conference on WWW. 119–130.

[30] Matei Zaharia, Mosharaf Chowdhury, Thathagata Das, Ankur Dave, Justin Ma, Murphy McCue, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In Usenix NSDI. 3–14.

[31] Yao Zhao, Yingjian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. 2009. BrotGraph: Large Scale Spanning Botnet Detection. In USENIX NSDI 2009. April 22-24, 2009. Boston, MA, USA. 321–334.