FRAUDAR: Bounding Graph Fraud in the Face of Camouflage

Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, Christos Faloutsos
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213, USA
bhooi@andrew.cmu.edu, {hyunahs, abeutel, neilshah, kijungs, christos}@cs.cmu.edu

ABSTRACT

Given a bipartite graph of users and the products that they review, or followers and followees, how can we detect fake reviews or follows? Existing fraud detection methods (spectral, etc.) try to identify dense subgraphs of nodes that are sparsely connected to the remaining graph. Fraudsters can evade these methods using *camouflage*, by adding reviews or follows with honest targets so that they look "normal". Even worse, some fraudsters use *hijacked accounts* from honest users, and then the camouflage is indeed organic.

Our focus is to spot fraudsters in the presence of camouflage or hijacked accounts. We propose FRAUDAR, an algorithm that (a) is camouflage-resistant, (b) provides upper bounds on the effectiveness of fraudsters, and (c) is effective in real-world data. Experimental results under various attacks show that FRAUDAR outperforms the top competitor in accuracy of detecting both camouflaged and non-camouflaged fraud. Additionally, in real-world experiments with a Twitter follower-followee graph of 1.47 billion edges, FRAUDAR successfully detected a subgraph of more than 4000 detected accounts, of which a majority had tweets showing that they used follower-buying services.

1. INTRODUCTION

How can we detect if a politician has purchased fake followers on Twitter, or if a product's reviews on Amazon are genuine? More challengingly, how can we *provably* prevent fraudsters who sell fake followers and reviews for various web services from evading our detection systems? In this paper we focus on precisely this problem – specifically, how can we design a fraud detection system with strong, provable guarantees of robustness?

Given the rise in popularity of social networks and other web services in recent years, fraudsters have strong incentives to manipulate these services. On several shady websites, anyone can buy fake Facebook page-likes or Twitter followers by the thousands. Yelp, Amazon and TripAdvisor fake reviews are also available for sale, misleading consumers about restaurants, hotels, and other services and products. Detecting and neutralizing these actions is important for companies and consumers alike.

The tell-tale sign of such fraudulent actions is that fraudsters must add many edges, creating unusually *large* and *dense* regions in the adjacency matrix of the graph (see Figure 2). Smart fraudsters will also try to 'look normal', by adding links to popular items/idols (like famous singers/actors, or well-liked products) - this behavior is called "*camouflage*" in the recent literature. State-of-the-art algorithms, such as SPOKEN [23] and NETPROBE [21] exploit exactly the density signal, but do not account for "camouflage."

We propose FRAUDAR, a novel approach for successfully detecting fraudsters under camouflage, and we give *provable* limits on undetectable fraud. We provide data-dependent limits on the maximum number of edges a group of fraudulent adversaries can have without being detected, on a wide variety of real world graphs. As shown in Figure 1(a), FRAUDAR provides limits on undetectable fraud, and additionally provides novel optimizations that strengthen this bound.

Moreover, our method outperforms competitors and finds real world fraud on Twitter. In Figure 1(b) we find that FRAUDAR detects injected fraud with high accuracy, even in the case of camouflage, where prior methods struggle to detect fraudulent attacks. Additionally, when tested on a Twitter graph from 2009, FRAUDAR finds a 4031 by 4313 subgraph that is 68% dense. As shown in Figure 1(c-d), we find that *a majority* of the detected accounts had tweets showing that they used follower-buying services, and had gone undetected by Twitter for the 7 years since the data was collected. Finally, our method is scalable, with near linear runtime in the data size.

Thus, our main contributions are as follows:

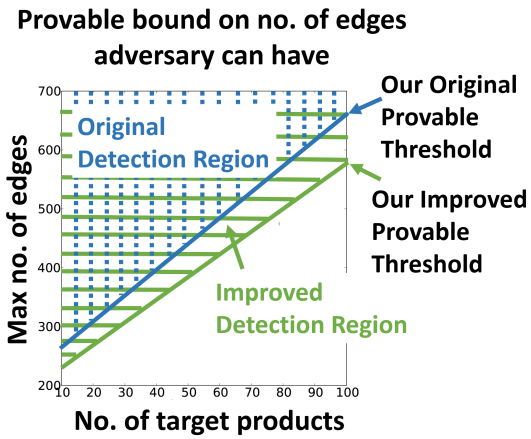
- **Metric**: we propose a novel family of metrics which satisfies intuitive "axioms" and has several advantages as a suspiciousness metric.
- **Theoretical Guarantees**: we provide a provable bound on how much fraud an adversary can have in the graph without being caught, even in the face of camouflage. Additionally, we improve the theoretical bound through novel optimizations that better distinguish fraud and normal behavior in real-world data.
- **Effectiveness**: FRAUDAR outperforms state-of-the-art methods in detecting various fraud attacks in real world graphs, and detects a large amount of previously undetected fraudulent behavior on Twitter.
- **Scalability**: FRAUDAR is scalable, with near-linear time complexity in the number of edges.

Furthermore, FRAUDAR offers natural extensibility and can easily incorporate more complex relations available in certain contexts such as review text, IP addresses, etc.

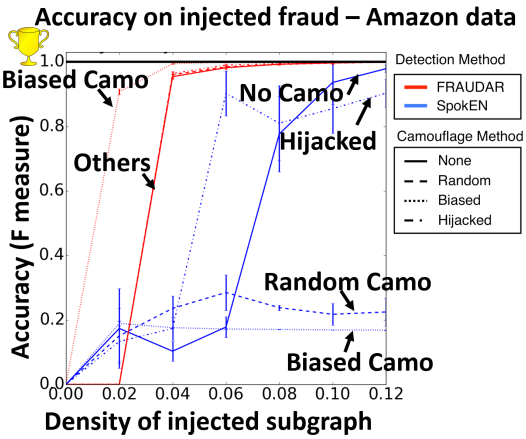
Reproducibility: Our code is open-sourced at **www.andrew.cmu.edu/user/bhooi/camo.zip**.

KDD '16, August 13-17, 2016, San Francisco, CA, USA

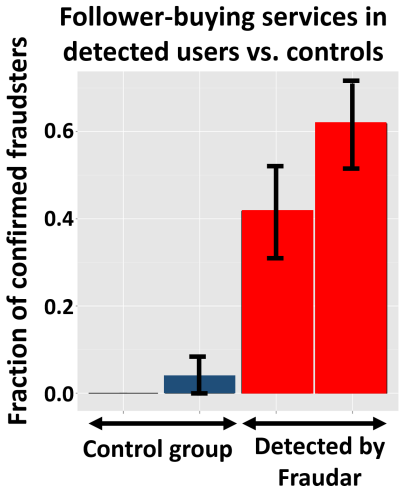
DOI: http://dx.doi.org/10.1145/2939672.2939747



(a) Our theoretical bound on detectable region



(b) FRAUDAR outperforms competitors



(c) FRAUDAR detects many confirmed fraudsters

(d) Sample fraudster detected

Figure 1: **(a) Our theoretical thresholds:** fraudsters in the detection region will be caught by our approach. Our novel optimizations improve the (data-dependent) bounds by lowering it to the green region. **(b) Fraudar outperforms competitors. (c) A large fraction of accounts flagged by our algorithm are confirmed fraudsters**: both among detected followees (left red bar) and followers (right red bar) compared to almost none among non-flagged accounts (2 control groups). Confirmation was done by inspecting Tweets that advertise *TweepMe* or *TweeterGetter*. **(d) Real-life results - a sample fraudster caught.**

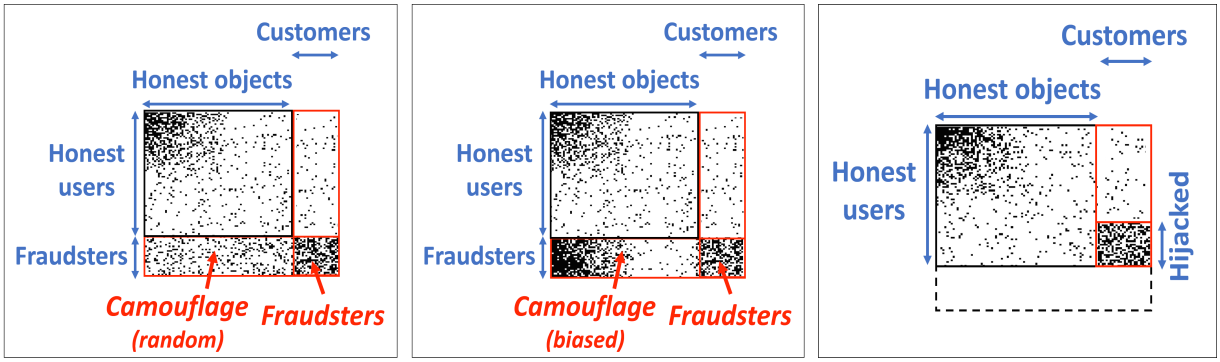
2. BACKGROUND AND RELATED WORK

Fraud detection has received significant focus in recent years. Many existing methods aim to detect fraud through review text [13, 20]. However, these approaches are typically not adversarially robust: spammers can carefully select their review texts to avoid detection. Even without knowledge of the detection system, they may mimic normal user reviews as closely as possible. Graph-based approaches detect groups of spammers, often by identifying unexpectedly dense regions of the graph of users and products. Such methods are potentially harder to evade, as creating fake reviews unavoidably generates edges in the graph. Graph-based methods may be classified into global and local methods.

Global methods: Building on singular value decomposition (SVD), latent factor models, and belief propagation (BP), these model the entire graph to find fraud. SPOKEN [23] considered the "eigenspokes" pattern produced by pairs of eigenvectors of graphs, and was later generalized for fraud detection [12]. FBOX [25] builds on SVD but focuses on detecting attacks missed by spectral techniques. Several methods have used HITS [15]-like ideas to detect fraud in graphs [3, 6, 9, 11, 30]. BP has been used for fraud classification on eBay [21], and fraud detection [1]. All of these methods have been successful in finding fraud but they offer no guarantees of robustness. [25] performs adversarial analysis for spectral algorithms, showing that attacks of small enough scale will necessarily evade detection methods which rely on the top k SVD components.

Local clustering methods: A different direction for fraud detection focuses on local subgraphs, by analyzing the properties of egonets to detect fraud [5,22]. COPYCATCH [2] and GETTHESCOOP [12] use local search heuristics to find relevant dense bipartite subgraphs. However, without guarantees on the search algorithm, the algorithms may not be robust to intelligent adversaries.

Dense subgraph mining: Finding dense subgraphs has been an important focus of graph theory communities and has been studied from a wide array of perspectives [7, 14]. Most closely related



(a) Attacks with random camouflage (b) Attacks with biased camouflage (c) Hijacked accounts

Figure 2: **Three examples of possible attacks**.

	CopyCatch	CatchSync	BP-based methods	SpokEn	fBox	GetTheScoop	FRAUDAR
Detects dense blocks	✓	✓	✓	✓	✓	✓	✓
Camouflage-resistant	✓	?			?		✓
Theoretical guarantees							✓

Table 1: Comparison between FRAUDAR and other fraud detection algorithms.

Symbol	Interpretation
$\mathcal{U} = \{u_1, \ldots, u_m\}$	Users
$\mathcal{W} = \{w_1, \ldots, w_n\}$	Objects
$\mathcal{V}$	Nodes of bipartite graph: $\mathcal{U} \cup \mathcal{W}$
G	Bipartite graph $G = (\mathcal{V}, \mathcal{E})$
$\mathcal{A}$	Subset of users
$\mathcal{B}$	Subset of objects
$\mathcal{S}$	Subset of nodes, $\mathcal{S} = \mathcal{A} \cup \mathcal{B}$
$g(\mathcal{S})$	Density metric
$f(\mathcal{S})$	'Total suspiciousness' metric (2)
$\mathcal{X}$	Current set of nodes in the greedy algorithm
Δ_i	$f(\mathcal{X} \setminus \{i\}) - f(\mathcal{X})$
$\hat{\mathcal{A}}, \hat{\mathcal{B}}$	Users (resp. objects) returned by FRAUDAR
m_0, n_0	No. of users (resp. objects) in fraud block
d_i	ith column sum of adjacency matrix
λ	Min. fraction of fraud edges per customer
$g_{\log}$	Logarithmic weighted metric

Table 2: Symbols and Definitions

to ours is Charikar's work on finding subgraphs with large average degree [4], which shows that subgraph average degree can be optimized with approximation guarantees. Variants have been proposed to efficiently find large, dense subgraphs [27], with approximation guarantees. To our knowledge, however, this is the first work which adapts this theoretical perspective to the challenges of fraud detection and camouflage resistance, and achieves meaningful bounds for our application. Moreover, our work differs from these in its setting of bipartite graphs, and in the use of edge reweighting to further increase accuracy.

Social network-based Sybil defense: Multiple identity or 'Sybil' attacks pose problems of malicious behavior in distributed systems. SybilGuard [32] and SybilLimit [31] use a decentralized random walk approach to limit the number of Sybil attackers. SumUp [26] and Iolaus [19] adapt this to content rating settings. However, these systems rely on a separate trust network between users; our setting is fundamentally different as our approach works directly with the user-product bipartite graph.

Handling camouflage: [8, 28] consider fraud detection methods that are robust to camouflage attacks. However, both methods focus on the time-series domain, observing changes in the behavior of fraudsters from system access logs rather than graph data.

A comparison between FRAUDAR and other fraud detection algorithms is summarized in Table 1. Our proposed method FRAUDAR is the only one that matches all specifications.

3. PROBLEM DEFINITION

Consider a set of m users $\mathcal{U} = \{u_1, \ldots, u_m\}$ and n objects $\mathcal{W} = \{w_1, \ldots, w_n\}$ connected according to a bipartite graph $G = (\mathcal{U} \cup \mathcal{W}, \mathcal{E})$. We can consider the objects to be followees on Twitter or products on Amazon. Table 2 gives a complete list of the symbols we use throughout the paper. We now describe our attack model and then our problem definition.

Attack model.

We assume that fraudsters are hired to use users they control to add edges pointing to a subset of nodes in $\mathcal{W}$. For example, a business may pay for followers on Twitter or positive reviews on Yelp. In general, fraudsters add a large number of edges, inducing a dense subgraph between the fraudster accounts and customers, as shown in the bottom right corner of each subplot of Figure 2. This general characteristic of fraud was found to be true in our experiments on real datasets, as well as in many other papers which use dense blocks to detect fraud [1, 2, 12, 21, 23].

To mask the fraud, fraudster accounts can add arbitrary "camouflage", i.e. edges pointing from their user accounts to any of the nodes in $\mathcal{W}$ that are not customers. We assume that fraudsters have *complete* knowledge of the graph and fraud detection mechanisms, enabling worst-case camouflage for any fraud detection system we create. Examples of the possible types of camouflage are given in Figure 2: (a) adding camouflage edges to random honest users, (b) camouflage biased toward high degree nodes, (c) using hijacked ac-

counts, whereby fraudster accounts have realistic patterns of camouflage essentially similar to that of honest users.

While it is trivial for fraud accounts to add edges to any other node, it is more difficult for customer accounts to get honest edges. In particular, we assume that a customer would try to increase their number of incoming edges by a significant portion, and as a result a fraction, $\lambda \in [0, 1]$, of their incoming edges will be from fraudsters. This assumption would manifest itself as customers wanting to boost their follower count to seem *noticeably* more popular or a restaurant wanting a *significant* number of positive ratings to shift its average "number of stars" on Yelp. We will demonstrate how using this real world pattern significantly improves fraud detection both theoretically and in practice.

Desired properties of detection approach.

Our goal is to detect dense subgraphs in G, typically indicative of fraudulent groups of users and objects, like in the bottom-right of Figure 2.

INFORMAL PROBLEM 1.
Given a bipartite graph, detect attacks so as to minimize the number of edges that fraudsters can add pointing to customers without being detected.

Given that we want our detection algorithm to be able to handle camouflage, we define the requirements for a *camouflage-resistant* algorithm:

DEFINITION 1. *Let $(\mathcal{A}, \mathcal{B})$ be a block consisting of fraudulent users and objects. A density metric g is* camouflage-resistant *if when any amount of camouflage is added by the adversary, $g(\mathcal{A} \cup \mathcal{B})$ does not decrease.*

That is, fraudsters cannot make themselves less suspicious by adding camouflage. Our goal is to find a fraud detection approach satisfying the following criteria:

PROBLEM DEFINITION 1 (DENSE SUBGRAPH DETECTION).
Design a class of density metrics for bipartite graphs, which can be optimized (1) in near-linear time, (2) within a constant factor of the optimum, and (3) is minimally affected by camouflage edges added by adversaries.

Obtaining theoretical guarantees on the near-optimality of the returned subgraph is important because, as we will later show, it allows us to offer guarantees against *worst-case* fraudsters.

4. PROPOSED METHOD

Given this problem definition and attack model, we now offer FRAUDAR and our theoretical analysis of FRAUDAR.

4.1 Metric

In this section, we propose a class of metrics g that have particularly desirable properties when used as suspiciousness metrics. Namely, we will show that if g takes the form in (1) and (2), then it can be optimized in a way that is (a) scalable; (b) offers theoretical guarantees, and (c) is robust to camouflage.

Let $\mathcal{A} \subseteq \mathcal{U}$ be a subset of users and $\mathcal{B} \subseteq \mathcal{W}$ be a subset of objects. Let $\mathcal{S} = \mathcal{A} \cup \mathcal{B}$, and $\mathcal{V} = \mathcal{U} \cup \mathcal{W}$. For the rest of this paper, we use g to denote the density metric that the algorithm will optimize, i.e. the algorithm will find $\mathcal{S}$ to (approximately) maximize $g(\mathcal{S})$. Note that g has a single argument, which is the union of the users and objects whose suspiciousness we are evaluating.

We propose using density metrics g of the following form:

$$g(\mathcal{S}) = \frac{f(\mathcal{S})}{|\mathcal{S}|} \tag{1}$$

where total suspiciousness f is:

$$\begin{aligned} f(\mathcal{S}) &= f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S}) \\ &= \sum_{i \in \mathcal{S}} a_i + \sum_{i,j \in \mathcal{S} \wedge (i,j) \in \mathcal{E}} c_{ij}, \end{aligned} \tag{2}$$

for some constants $a_i \geq 0$ and constants $c_{ij} > 0$.

Intuitively, the node suspiciousness $f_{\mathcal{V}}(\mathcal{S})$ is a sum of constants a_i corresponding to the users and objects in $\mathcal{S}$, which can be thought of as how individually suspicious that particular user or object is. The edge suspiciousness $f_{\mathcal{E}}(\mathcal{S})$ is a sum of constants c_{ij} corresponding to the edges in between $\mathcal{S}$, which can be thought of as how suspicious that particular edge is (e.g. the suspiciousness of the text of a review by user i for object j).

There are many advantages to metrics of this form. Firstly, metrics of this form can be optimized in a way that is (a) scalable; (b) offers theoretical guarantees, and (c) is robust to camouflage, as we demonstrate in the rest of this paper. All 3 of these properties hold due to the particular chosen form in (1) and (2).

Secondly, metrics of this form obey a number of basic properties (or "axioms") that we would intuitively expect a reasonable suspiciousness metric should meet, as we next show. These basic properties are adapted from the "axioms for suspiciousness metrics," proposed in [10], to our setting where node and edge weights exist.

AXIOM 1 (NODE SUSPICIOUSNESS). *A subset consisting of higher suspiciousness nodes is more suspicious than one consisting of lower suspiciousness nodes, if the other conditions are fixed. Formally,*

$$|\mathcal{S}| = |\mathcal{S}'| \wedge f_{\mathcal{E}}(\mathcal{S}) = f_{\mathcal{E}}(\mathcal{S}') \wedge f_{\mathcal{V}}(\mathcal{S}) > f_{\mathcal{V}}(\mathcal{S}') \Rightarrow g(\mathcal{S}) > g(\mathcal{S}')$$

AXIOM 2 (EDGE SUSPICIOUSNESS). *Adding edges within a subset increases the suspiciousness of the subset if the other conditions are fixed. Formally,*

$$e \notin \mathcal{E} \Rightarrow g(\mathcal{S}(\mathcal{V}, \mathcal{E} \cup \{e\})) > g(\mathcal{S}(\mathcal{V}, \mathcal{E}))$$

where $\mathcal{S}(\mathcal{V}, \mathcal{E})$ is the subgraph induced by $\mathcal{S}$ in the graph $(\mathcal{V}, \mathcal{E})$.

The **edge density** $\rho(\mathcal{S})$ of an induced subgraph is its number of edges divided by its maximum possible number of edges.

AXIOM 3 (SIZE). *Assuming node and edge weights are all equal, larger subsets are more suspicious than smaller subsets with the same edge density. Formally, given $a_i = a \; \forall \; i$, and $c_{ij} = b \; \forall \; (i, j) \in \mathcal{E}$:*

$$|\mathcal{S}| > |\mathcal{S}'| \;\; \wedge \;\; \mathcal{S} \supset \mathcal{S}' \;\; \wedge \;\; \rho(\mathcal{S}) = \rho(\mathcal{S}') \Rightarrow g(\mathcal{S}) > g(\mathcal{S}')$$

AXIOM 4 (CONCENTRATION). *A subset with smaller size is more suspicious than one with the same total suspiciousness but larger size. Formally,*

$$|\mathcal{S}| < |\mathcal{S}'| \;\; \wedge \;\; f(\mathcal{S}) = f(\mathcal{S}') \Rightarrow g(\mathcal{S}) > g(\mathcal{S}')$$

Density metrics g of the form defined in Equation (1) satisfy these properties:

THEOREM 1. *The density metric defined in* (1) *satisfies axioms 1 to 4.*

PROOF. See appendix A. ■

Note some simple metrics that violate these axioms: the edge density $\rho(\mathcal{S})$ itself, as a metric, does not increase with the size of $\mathcal{S}$ and hence violates axiom 3. On the opposite end, the total edge

weight function $\sum_{i,j\in\mathcal{S}\wedge(i,j)\in\mathcal{E}} c_{ij}$ as a metric violates axiom 4 as it does not consider how concentrated the edge weight is. In contrast, g scales in a reasonable manner as its size or concentration changes.

A simple example of a metric g as defined in (1) and (2) is the bipartite graph average degree:

EXAMPLE 1. *(Bipartite Graph Average Degree) Let $a_i = 0$, and let $c_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and 0 otherwise. In the expression (2) for $f(\mathcal{S})$, we add one term c_{ij} for each edge (i,j) for which i,j are both in the subset $\mathcal{S}$. Thus, $f(\mathcal{S})$ is equal to the number of edges in the subgraph spanned by $\mathcal{S}$, or half the total degree in the subgraph spanned by $\mathcal{S}$. As a result, $g(\mathcal{S}) = \frac{f(\mathcal{S})}{|\mathcal{S}|}$ is half the average degree of the subgraph spanned by $\mathcal{S}$.*

4.2 Algorithm

Let f and g be as given in (1) and (2). In this section, we give an algorithm for optimizing the density metric g in near-linear time.

Algorithm 0 describes our proposed FRAUDAR algorithm, a greedy approach inspired by that of [4] but which covers our broader objective class. We start with the entire set of nodes $\mathcal{U} \cup \mathcal{W}$, then repeatedly remove the node which results in the highest value of g evaluated on the remaining set of nodes. Formally, denote by $\mathcal{X}$ the current set we are optimizing over; initially we set $\mathcal{X} = \mathcal{U} \cup \mathcal{W}$. Let $\Delta_i = f(\mathcal{X}\setminus\{i\}) - f(\mathcal{X})$ be the change in f when we remove i from the current set. At each step, we will select i to maximize Δ_i, i.e. to leave behind the set with highest value of f. We then remove i from $\mathcal{X}$. We then repeat this process: we recompute the values of Δ_j, then choose the next node to delete, and so on. This leads to a shrinking series of sets $\mathcal{X}$ over time, denoted $\mathcal{X}_0, \ldots, \mathcal{X}_{m+n}$ of sizes $m+n, \ldots, 0$. At the end, we return the one of these that maximizes the density metric g.

The key fact that allows the algorithm to be efficient is the forms for f and g in (1) and (2). When i is removed, the only values of Δ_j which need to be updated are those where j is a neighbor of i. This is because for all other j, the expressions (1) and (2) ensure that Δ_j does not change. Hence, the updates are fast: for each $(i,j) \in \mathcal{E}$, over the lifetime of the algorithm we will perform at most one such update over this edge, for a total of $O(|\mathcal{E}|)$ updates. Using appropriate data structures, as we next describe, each update can be performed in $O(\log|\mathcal{V}|)$ time, totalling $O(|\mathcal{E}|\log|\mathcal{V}|)$ time.

Priority Tree.

Each element $i \in \mathcal{X}$ has a priority that will change as the algorithm progresses: the priority of element i at the tth iteration is $\Delta_i = f(\mathcal{X}_t \setminus \{i\}) - f(\mathcal{X}_t)$. This ensures that in Line 5, the element $i^* = \arg\max_{i\in\mathcal{X}_t} g(\mathcal{X}_t \setminus \{i\})$ we wish to find is exactly the element of highest priority, allowing us to retrieve it quickly (in $O(\log|\mathcal{V}|)$ time, as we explain below). Note that it does not matter if we use f or g in the $\arg\max$ since the denominator of g in (1), $|\mathcal{X}_t \setminus \{i\}|$, is the same for all possible deletions i.

These priorities are stored in the priority tree T constructed in line 2 of Algorithm 0. This data structure is a binary tree with all $|\mathcal{V}|$ elements as leaves, all at the bottom level of the tree. Each internal node keeps track of the maximum priority of its two children.

The priority tree supports fast retrieval of the maximum priority element (used in Line 5 of Algorithm 0); it does this by starting at the root and repeatedly moving to the child with higher priority. It also supports quickly updating priorities: since all the leaves can be stored in fixed locations, we can easily retrieve the leaf at any index to update its priority. Then, after updating that node's priority, we travel up the tree to update each parent up to the root (used in Line 6). Each of these operations on T takes $O(\log|\mathcal{V}|)$ time.

Scalability.

The bottleneck is the loop in Lines 5 to 7 which runs $m+n$ times. Lines 5 and 6 take $O(\log|\mathcal{V}|)$ as discussed, while Line 7 is constant time. Finally, we need $|\mathcal{E}|$ updates to node priorities, one for each edge. Thus the algorithm takes $O(|\mathcal{E}|\log|\mathcal{V}|)$ time.

4.3 Theoretical Bounds

So far, we have shown that g can be optimized in near-linear time. In this section, we will show that when f and g are of the form (1) and (2), FRAUDAR is guaranteed to return a solution of at least half of the optimum value.

THEOREM 2. *Let $\mathcal{A}, \mathcal{B}$ be the set of users and objects returned by* FRAUDAR. *Then:*

$$g(\mathcal{A}\cup\mathcal{B}) \geq \frac{1}{2} g_{OPT}$$

where g_{OPT} is the maximum value of g, i.e.

$$g_{OPT} = \max_{\mathcal{A}',\mathcal{B}'} g(\mathcal{A}'\cup\mathcal{B}')$$

PROOF. Let the elements of $\mathcal{V}$ be labeled $v_1, v_2, \ldots, v_{m+n}$. We define 'weight assigned to node v_i in $\mathcal{S}$' as

$$w_i(\mathcal{S}) = a_i + \sum_{(v_j\in\mathcal{S})\wedge((v_i,v_j)\in\mathcal{E})} c_{ij} + \sum_{(v_j\in\mathcal{S})\wedge((v_j,v_i)\in\mathcal{E})} c_{ji}$$

where $a_i (\geq 0)$ indicates the weight of node v_i and $c_{ij} (> 0)$ indicates that of edge (v_i, v_j) as in (2). Note that when node v_i is removed from the current set $\mathcal{S}$ at some point in the algorithm, $w_i(\mathcal{S})$ is the decrease in the value of f, since it is the sum of all terms excluded in (2) when node v_i is removed.

Now consider the optimal set $\mathcal{S}^*$. For each node $v_i \in \mathcal{S}^*$, we claim that $w_i(\mathcal{S}^*) \geq g(\mathcal{S}^*)$. Otherwise, removing a node with $w_i(\mathcal{S}^*) < g(\mathcal{S}^*)$ results in

$$\begin{aligned} g' &= \frac{f(\mathcal{S}^*) - w_i(\mathcal{S}^*)}{|S^*|-1} > \frac{f(\mathcal{S}^*) - g(\mathcal{S}^*)}{|S^*|-1} \\ &= \frac{f(\mathcal{S}^*) - f(\mathcal{S}^*)/|\mathcal{S}^*|}{|S^*|-1} = g(\mathcal{S}^*), \end{aligned}$$

which is a contradiction.

Let v_i be the node that FRAUDAR removes first among those in $\mathcal{S}^*$, and let $\mathcal{S}'$ be the set before FRAUDAR removes v_i. Then, since $\mathcal{S}' \supset \mathcal{S}^*$, $w_i(\mathcal{S}') \geq w_i(\mathcal{S}^*)$. Moreover, since FRAUDAR chooses to remove node v_i, for each of the other remaining nodes $v_j \in \mathcal{S}'$, $w_j(\mathcal{S}') \geq w_i(\mathcal{S}')$. Since each term in $f(\mathcal{S}')$ can be assigned to at most two nodes, summing over j gives $f(\mathcal{S}') \geq \frac{|\mathcal{S}'|w_i(\mathcal{S}')}{2}$. Also note that $g(\mathcal{A}\cup\mathcal{B}) \geq g(\mathcal{S}')$ since FRAUDAR returns the best solution that it encounters. We conclude that

$$g(\mathcal{A}\cup\mathcal{B}) \geq g(\mathcal{S}') = \frac{f(\mathcal{S}')}{|\mathcal{S}'|} \geq \frac{w_i(\mathcal{S}')}{2} \geq \frac{w_i(\mathcal{S}^*)}{2} \geq \frac{g(\mathcal{S}^*)}{2}.$$

■

4.4 Edge Weights and Camouflage Resistance

So far, we have seen that metrics of the form: $g(\mathcal{S}) = \frac{f(\mathcal{S})}{|\mathcal{S}|}$, where $f(\mathcal{S}) = \sum_{i\in\mathcal{S}} a_i + \sum_{i,j\in\mathcal{S}\wedge(i,j)\in\mathcal{E}} c_{ij}$ can be optimized efficiently and with approximation guarantees. In this section, we show how we can select metrics within this class that are resistant to camouflage, i.e. they do not allow fraudulent users to make themselves less suspicious by adding *camouflage edges*, i.e. edges toward honest objects.

Recall that a_i and c_{ij} are the weights of node i and edge ij, while $f(\mathcal{S})$ is the total node and edge weight in $\mathcal{S}$. A key idea of

Algorithm 1 FRAUDAR, which greedily removes nodes to maximize a metric g. Line 5 and 6 run in $O(\log |\mathcal{V}|)$ time, using a data structure described in Section 4.2.

Require: Bipartite $G = (\mathcal{U} \cup \mathcal{W}, \mathcal{E})$; density metric g of the form in (1)
1: **procedure** FRAUDAR (G, g)
2: Construct priority tree T from $\mathcal{U} \cup \mathcal{W}$ ▷ see Section 4.2
3: $\mathcal{X}_0 \leftarrow \mathcal{U} \cup \mathcal{W}$ ▷ suspicious set is initially the entire set of nodes $\mathcal{U} \cup \mathcal{W}$
4: **for** $t = 1, \ldots, (m + n)$ **do**
5: $i^* \leftarrow \arg\max_{i \in \mathcal{X}_i} g(\mathcal{X}_i \setminus \{i\})$ ▷ exonerate least suspicious node
6: Update priorities in T for all neighbors of i^*
7: $\mathcal{X}_t \leftarrow \mathcal{X}_{t-1} \setminus \{i^*\}$
8: **end for**
9: **return** $\arg\max_{\mathcal{X}_i \in \{\mathcal{X}_0, \ldots, \mathcal{X}_{m+n}\}} g(\mathcal{X}_i)$ ▷ return most suspicious set $\mathcal{X}_i$
10: **end procedure**

our approach is that instead of treating every edge equally, we assign a lower weight c_{ij} when the target object j has high degree. This is because objects of very high degree are not necessarily suspicious (since highly popular objects commonly exist). Thus, this weighting allows us to put greater emphasis on objects within unexpectedly dense subgraphs, rather than just high degree objects.

If we consider the adjacency matrix with rows representing users and columns representing objects, we would like to downweight columns with high column sum (column-weighting). A simple result we show in this section is that column-weightings are camouflage resistant. Recall that a density metric g is camouflage-resistant if $g(\mathcal{A} \cup \mathcal{B})$ does not decrease when any amount of camouflage is added by an adversary with fraudulent users $\mathcal{A}$ and customers $\mathcal{B}$. Let d_i be the the ith column sum, i.e. the degree of object i.

Formally, define a **column-weighting** as a choice of weighting in which each c_{ij} is a function of the respective column sum, i.e. $c_{ij} = h(d_j)$ for some function h.

THEOREM 3. *Let c_{ij} be a column-weighting. Then g (as defined in* (1) *and* (2)*) is camouflage resistant.*

PROOF. Adding camouflage only adds edges in the region between $\mathcal{A}$ (fraudulent users) and $\mathcal{B}^C$ (honest objects). It does not add or remove edges within the fraudulent block; moreover, the weights of these edges do not change either as their weights only depend on the column degrees of $\mathcal{B}$, which do not change when camouflage is added. Thus the value of g does not change. ■

A natural follow-up question is whether camouflage-resistance also holds for row-weightings (i.e. selecting c_{ij} to be a function of the corresponding row sum). It turns out that row-weightings are in general not camouflage resistant. This is because a fraudulent user account can add a large number of camouflage edges, thereby increasing their row sum, decreasing the weight of each of their edges. Thus $g(\mathcal{A} \cup \mathcal{B})$ decreases, meaning that g is not camouflage resistant.

Hence we may choose any column-weighting while ensuring camouflage resistance. The remaining question is what function to choose for the column-weighting, i.e. the function h where $c_{ij} = h(d_j)$. It should be decreasing (so as to downweight columns with high sum). It should shrink more slowly than $h(x) = 1/x$, since $h(x) = 1/x$ allows a single edge to contribute as much as the total contribution of a column with any number of edges, causing us to catch columns with single ones rather than dense blocks.

Within the remaining space of choices, we note that a very similar problem of downweighting based on column frequency appears in deciding the form of the 'inverse document frequency' term of the popular heuristic *tf-idf* weighting scheme [24], in which logarithmic weighting of frequency has been empirically found to perform well. We also show empirical results (in Section 5.1) that logarithmic weighting leads to strong theoretical bounds. For these reasons, we recommend using $h(x) = 1/\log(x + c)$, where c is a small constant (set to 5 in our experiments) to prevent the denominator from becoming zero, or excessive variability for small values of x. We use the resulting density metric (denoted $g_{\log}$) in our experiments.

4.5 Implications: Bounding Fraud

Figure 1(a) shows curves representing our theoretical bounds on the maximum amount of fraud that can be present for each possible size of the fraudulent block, based on Theorem 2. We now explain how such bounds can be computed from Theorem 2. Assume that the fraudulent block contains m_0 user accounts and n_0 customers.

In this section, we assume that no side information is present, so we set the a_i, the prior suspiciousness of each node, to 0. Thus here $g_{\log}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i,j \in \mathcal{S}} \frac{1}{\log(d_j + c)}$, where d_j is the degree of the jth object. Consider a fraudulent subgraph with m_0 user nodes and n_0 object nodes. Assume that each fraudulent customer has at least a certain fraction $0 < \lambda < 1$ of fraudulent edges: each fraudulent customer should be receiving at least a comparable fraction of fraudulent reviews to its actual honest reviews, otherwise it would not be profiting appreciably from the fraud.

THEOREM 4. *Let $(\hat{\mathcal{A}}, \hat{\mathcal{B}})$ be the block detected by* FRAUDAR. *Then the number of edges that a fraudulent block of size (m_0, n_0) can have without being detected is at most $2(m_0 + n_0) g_{\log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}}) \log(m_0/\lambda + c)$. In other words, our algorithm will detect a fraudulent block without fail if it contains more edges than this threshold.*

PROOF. By Theorem 2, $2g_{\log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}})$ is an upper bound on the value of $g_{\log}$ on any subgraph of users and objects. Since the fraudulent block has $m_0 + n_0$ nodes in total, thus $2(m_0 + n_0) g_{\log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}})$ is an upper bound on the value of total suspiciousness $f_{\log}$.

Moreover, each fraudulent customer has at most m_0 fraudulent edges joined to it, and since at least λ fraction of its edges must be fraudulent, it can have at most m_0/λ degree in total. Hence the weight of each fraudulent edge is at least $\frac{1}{\log(m_0/\lambda + c)}$. But since the total weighted degree is at most $2(m_0 + n_0) g_{\log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}})$, it follows that the number of fraudulent edges is at most $2(m_0 + n_0) g_{\log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}}) \log(m_0/\lambda + c)$. ■

We apply this bound to real data in Section 5.1.

	# of nodes	# of edges	Density	Content
Amazon [18]	28K (24K,4K)	28K	2.7e-4	Review
Trip Advisor [29]	84K (82K,2K)	90K	5.9e-4	Review
Epinion [17]	264K (132K,132K)	841K	4.8e-5	Who-trust-whom
Wiki-vote [17]	16K (8K,8K)	103K	1.5e-3	Vote

Table 3: Bipartite graph datasets used in our experiments.

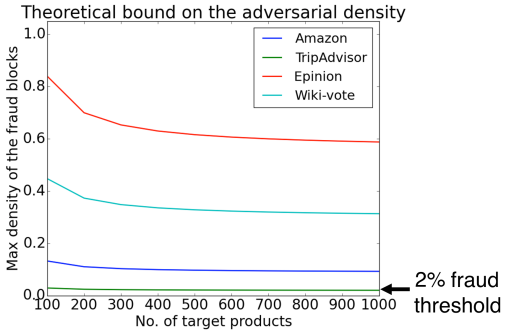


Figure 3: **FRAUDAR's bounds on fraud are stringent, on real graphs:** E.g., on TripAdvisor, the bound says that a fraudulent block containing 50 user accounts and anywhere between 100 and 1000 products must have density of $< 2\%$ to avoid detection.

5. EXPERIMENTS

We design experiments to answer the following questions:

Q1. Illustration of our theorem: How strong are the bounds that FRAUDAR provides in terms of bounding undetectable fraud in the graph? Does column weighting improve those bounds?

Q2. Evaluation on synthetic data: How accurately does FRAUDAR detect injected fraud under different types of camouflage attacks? Does FRAUDAR outperform state-of-the-art competitors?

Q3. Effectiveness in real-world data: Does FRAUDAR detect true fraud in real-world graphs? Have the fraudulent accounts already been detected by previous methods?

Q4. Scalability: Is FRAUDAR scalable with regard to the data size?

We implemented FRAUDAR in Python; all experiments were carried out on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM, running OS X 10.9.5. The code is available for download at **www.andrew.cmu.edu/user/bhooi/camo.zip**. We test FRAUDAR on a variety of real world datasets. Table 3 offers details on the datasets we used.

To test the accuracy of our method, we use synthetic attacks injected into our Amazon dataset. We structure our "attacks" as shown in Figure 2. We injected a fraudulent block of users and customers with varying densities. We assume $\lambda = 0.5$ for our theoretical bounds.

5.1 Q1. Illustration of our Theorem

In Figure 1 (a), we showed our theoretical bounds (Theorem 4) applied to compute the maximum number of edges an adversary with $m_0 = 50$ user nodes can have for various values of n_0. These are computed by running FRAUDAR under two weighting schemes. First, we use our $g_{\log}$ scheme exactly as in Theorem 4 to get an upper bound $2(m_0 + n_0)g_{log}(\hat{\mathcal{A}} \cup \hat{\mathcal{B}}) \log(m_0/\lambda + c)$ on the number of fraudulent edges; plotting this against n_0 gives the green region ('improved') in Figure 1 (a). The blue region ('original') comes from using the analogous procedure without the log-weighting, i.e. where $g(\mathcal{S})$ is half the average degree, as in Example 1.

In this case, we see that the log-weighted scheme provides stronger bounds, since the bound is lower, i.e. an adversary should have fewer edges in order not to be detected. Intuitively, this happens because down-weighting high degree columns decreases the weight of many of the honest high degree objects in the dataset, so groups of adversaries stand out more, resulting in stronger bounds on how many edges an adversary can have.

Next, we apply our FRAUDAR in the same way over various real-world graphs to analyze the theoretical upper bounds computed by FRAUDAR on the density that fraudulent blocks can have. We run FRAUDAR on four real-world graphs: *Amazon* [18], *Trip Advisor* [29], *Epinions* [17], and *Wiki-vote* [17]. The detailed description of each graph is in Table 3. For all datasets, Figure 3 shows the maximum number of fraudulent edges that an adversary can have without being detected, assuming 50 fraudulent users and varying the number of fraudulent customers. We see that we can detect fraud most easily in *Trip Advisor*, followed by *Epinion, Wiki-vote, Amazon*; even a fairly sparse block of density around 0.05 would stand out strongly in the *Trip Advisor* graph. While density is important in determining how easy it is to detect fraud in each graph (fraudulent blocks stand out more strongly in a sparse graph), it is not the only factor. Indeed, *Wiki-vote* is actually denser than *Amazon*. In fact, the difficulty of detecting fraud in each graph is mainly determined by its densest blocks, since an adversarial block that is significantly less dense than the densest normal blocks in the graph is unlikely to be detected.

5.2 Q2. Evaluation on Synthetic Data

In Figure 1 (b), we demonstrated that FRAUDAR can effectively detect fraud under four types of camouflage attacks: 1) Injection of fraud with no camouflage, 2) random camouflage, 3) biased camouflage and 4) hijacked accounts, more accurately than competitors.

We conduct experiments based on the settings at the beginning of this section, averaged over 5 trials. For the camouflage scenarios 2) and 3), the amount of camouflage added per fraudulent user account was (on average) equal to the amount of actual fraudulent edges for that user. For the 'Random Camo' case, for each fake user node, camouflage edges were chosen at random, with on average the same number of camouflage edges as fraudulent edges, as shown in Figure 2 (a). For the 'Biased Camo' case, for each fake user node, camouflage edges were directed toward each object with probability proportional to the degree of the object as shown in Figure 2 (b). For the 'Hijacked' case, we used a random subset of existing users to form the fraudulent block.

In each case, we injected 200 fraudulent users and 200 fraudulent products with various edge densities to the subsetted Amazon review graph of 2000 users and 2000 products, with a density of 0.0006. We compare FRAUDAR to SPOKEN in their F measure ($= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$) in detecting the fake users. In the first set of experiments, we assume that no honest user added an edge to the fraudulent target (i.e. object) nodes.

As seen in Figure 1(b), the results demonstrate that FRAUDAR works robustly and efficiently against all four attacks, achieving F-

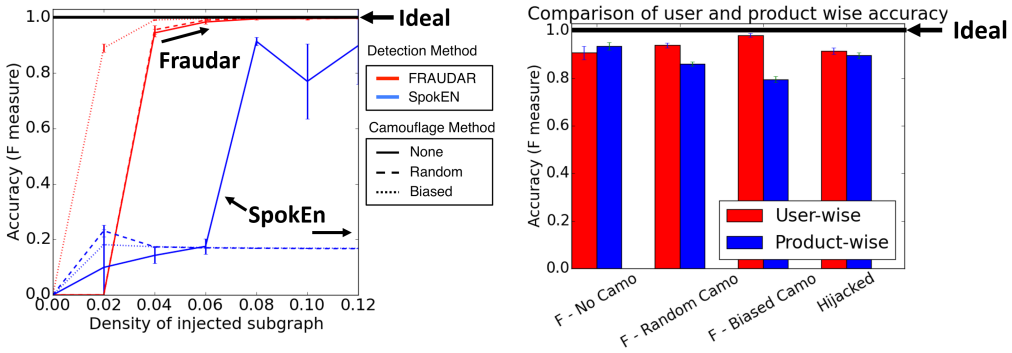


Figure 4: **(a) FRAUDAR outperforms competitors in multiple settings**. Accuracy of fraud detection on Amazon data in the experiment with "reverse camouflage" (edges from honest users to fraudulent products). **(b) FRAUDAR has similar and high accuracy both in detecting fraudulent users and fraudulent customers**. Comparison of accuracy on fake users and targets under four different camouflage attacks.

measures of over 0.95 on all four scenarios for densities of at least 0.04. On the other hand, SPOKEN was able to reach its maximum performance of 0.9 only when fraud blocks had densities of higher than 0.06 and under the 'no camouflage' scenario.

The experimental results in Figure 1(b) were based on the assumption that no honest user added an edge to the fraudulent target nodes. However, in a real-world environment, some honest users may add edges to the fraudulent target nodes (which we refer to as "reverse camouflage"). To incorporate this, we conducted another experiment using an attack model where we add edges between honest users and the fraudulent target nodes, but with sparser density compared to the fraud blocks. We added random edges to this region, with half the density of the fraud blocks. All other experimental settings were unchanged. The experimental results are shown in Figure 4 (a). For FRAUDAR, the results are generally similar. In contrast, SPOKEN shows slightly worse performance under this additional camouflage.

To show that FRAUDAR is effective both at catching fraudulent users accounts as well as fraudulent objects, we next separately evaluate the fraud detection of both fake users and fake targets using F measure. The basic experimental setup is same as before, with the density of the fraudulent blocks now fixed to 0.03. In Figure 4 (b), the bar plots are shown for the comparison. 'User-wise' (red) denotes the F measure of the detecting fake users, and 'target-wise' denotes the F measure of detecting fake target nodes. We see that in general, accuracy is high and fairly similar, but the performance in detecting fake users is slightly higher than that of detecting products.

5.3 Q3. Effectiveness on Real Data

In this section, we verify that FRAUDAR accurately detects a large block of fraudulent accounts in the Twitter follower-followee graph, as verified by hand labelling and by clear signs of fraud exhibited by a majority of the detected users. Indeed, a majority of the detected accounts had tweets advertising follower-buying services, and the tweets had not been removed or the accounts suspended for the 7 years since the data was collected. Figure 1(d) shows a sample fraudster caught by FRAUDAR.

The Twitter graph we use contains 41.7 million users and 1.47 billion follows; it was extracted in July 2009 and first used in [16]. On this graph, FRAUDAR detected a dense subgraph of size 4031 followers by 4313 followees. This subgraph is extremely dense, with 68% density, which is highly suspicious in itself.

To further investigate this block, we randomly sampled 125 followers and 125 followees in the block detected by FRAUDAR for hand labeling to determine how many of them appear fraudulent. To do this, we labeled which users were fraudulent based on the following characteristics of their profile data, chosen based on established criteria in the literature [25] summarized below.

- links on profile associated with malware or scams
- clear bot-like behavior (e.g. replying to large numbers of tweets with identical messages)
- account deleted
- account suspended

For comparison, we also construct two control groups of size 100 containing users that were not detected by the algorithm. The first control group contains randomly selected non-detected users. For the second (degree-matched) control group, we constructed it to match the follower count of users in the detected group; we do this by repeatedly selecting a random detected user, then finding another non-detected user who has at most 10% bigger or smaller follower count. During the labelling process, we shuffled the detected users with the control groups randomly and hid group memberships from labellers, labeling users in a "blind" manner.

Additionally, we also check and report how many of these users have Tweets containing the URLs of two known follower-buying services, *TweepMe* and *TweeterGetter*, showing that they had advertised these follower-buying services through tweets.

Note that this entire labelling process used only profile and tweet data and not follower-followee data, whereas our algorithm uses only follower-followee data, so the labelling is a fair estimate of the algorithm's accuracy. We present two pieces of evidence which strongly indicates fraud in the detected group. Firstly, the percentage of users with tweets advertising *TweepMe* or *Tweetergetter* is much higher among the detected users than among both control groups (Figure 5): 41% of the detected followers, and 26% for the detected followees. These rise to 62% and 42% respectively as shown in Figure 1(c) if we ignore deleted, protected and suspended accounts (for which profile information was unavailable). In the control groups, there were no mentions of *TweepMe* and very few mentions of *TweeterGetter*, as shown in Figure 5. Figure 5 shows the breakdown of our groups in terms of deleted and suspended users. Given the sparsity of *TweepMe* and *TweeterGetter* in the control groups, we see that the detected users are likely charac-

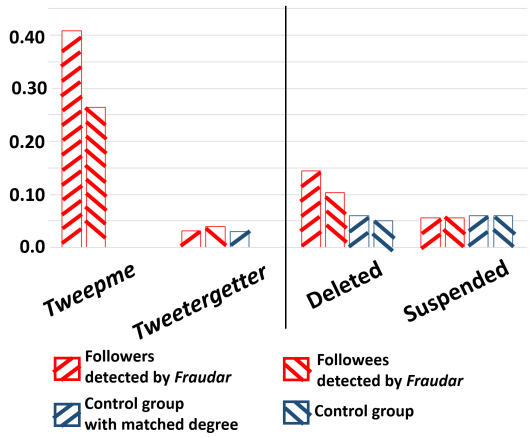


Figure 5: **FRAUDAR detects a large, clearly fraudulent block in Twitter.** A majority of the detected accounts were either deleted, suspended, or contained known follower-buying services, *TweepMe* and *TweeterGetter*. In comparison, the control groups had much less detected fraud.

Figure 6: **Follower-buying services:** a large fraction of detected accounts use *TweepMe* (bottom) or *TweeterGetter* (middle, top).

terized by a large block of users using these and possibly other follower-buying services, resulting in a dense block.

Secondly, we used our hand-labelling using the above criteria to determine how many of each group appear fraudulent. 57% of the detected followers and 40% of the followees were labelled as fraudulent, deleted or suspended accounts, but much fewer in the control groups, with 25% for the degree-matched control group, and 12% for control group with no condition. Thus both these results support the effectiveness of FRAUDAR in detecting fraudulent users in the real-world graphs.

5.4 Q4. Scalability

Figure 7 shows the near-linear scaling of FRAUDAR's running time in the number of edges. Here we used the Trip Advisor dataset, and subsampled user nodes in proportions of $0.7^0, \ldots, 0.7^{12}$. Slopes parallel to the main diagonal indicate linear growth.

6. CONCLUSION

In this paper, we propose FRAUDAR, a fraud detection algorithm which provably bounds the amount of fraud adversaries can have, even in face of camouflage. Our main contributions are as follows.

- **Metric**: we propose a novel family of metrics which satisfies intuitive "axioms" and has several advantages as a suspiciousness metric.
- **Theoretical Guarantees**: we provide theorems (See Theorem 2 in Section 4.3 and Theorem 4 in Section 4.5) on how FRAUDAR gives a provable upper bound on undetectable fraud. We also prove that our proposed metric is camouflage-resistant.
- **Effectiveness**: FRAUDAR was successfully applied on real-world graphs on fraud attacks with various types of camouflage, and outperformed the competitor. It also detected a large block of fraudulent activity in the Twitter follower-followee graph.
- **Scalability**: FRAUDAR runs near-linearly in the input size. (See Figure 7).

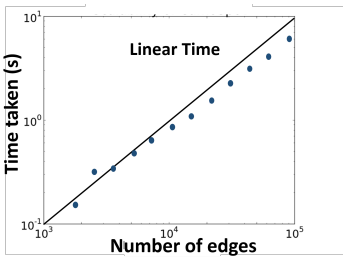


Figure 7: **FRAUDAR runs in near-linear time**: the curve *(blue)* shows the running time of FRAUDAR, compared to a linear function *(black)*.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1314632, DGE-1252522, and IIS-1408924.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

7. REFERENCES

[1] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In *ICWSM*, 2013.

[2] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *22nd WWW*, pages 119–130. International World Wide Web Conferences Steering Committee, 2013.

[3] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, 2012.

[4] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[5] C. Cortes, D. Pregibon, and C. Volinsky. *Communities of interest*. Springer, 2001.

[6] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi. Understanding and combating link farming in the twitter social network. In *21st WWW*, pages 61–70. ACM, 2012.

[7] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 87–93. IEEE, 2011.

[8] Z. Gu, K. Pei, Q. Wang, L. Si, X. Zhang, and D. Xu. Leaps: Detecting camouflaged attacks with statistical learning guided by program analysis.

[9] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB Endowment*, pages 576–587, 2004.

[10] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos. A general suspiciousness metric for dense blocks in multimodal data. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 781–786. IEEE, 2015.

[11] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Catchsync: catching synchronized behavior in large directed graphs. In *20th KDD*, pages 941–950. ACM, 2014.

[12] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Inferring strange behavior from connectivity pattern in social networks. In *Advances in Knowledge Discovery and Data Mining*, pages 126–138. Springer, 2014.

[13] N. Jindal and B. Liu. Opinion spam and analysis. In *ICDM 2008*, pages 219–230. ACM, 2008.

[14] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. *The University of Minnesota*, 2, 1995.

[15] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[16] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *19th WWW*, pages 591–600. ACM, 2010.

[17] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370. ACM, 2010.

[18] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[19] A. Molavi Kakhki, C. Kliman-Silver, and A. Mislove. Iolaus: Securing online content rating systems. In *22nd WWW*, pages 919–930. International World Wide Web Conferences Steering Committee, 2013.

[20] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[21] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *16th WWW*, pages 201–210. ACM, 2007.

[22] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller. Focused clustering and outlier detection in large attributed graphs. In *20th KDD*, pages 1346–1355. ACM, 2014.

[23] B. Prakash, M. Seshadri, A. Sridharan, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and community structure in large graphs. *PAKDD, 2010a*, 84, 2010.

[24] A. Rajaraman, J. D. Ullman, J. D. Ullman, and J. D. Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012.

[25] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. *arXiv preprint arXiv:1410.3915*, 2014.

[26] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *NSDI*, volume 9, pages 15–28, 2009.

[27] C. Tsourakakis. The k-clique densest subgraph problem. In *24th WWW*, pages 1122–1132. International World Wide Web Conferences Steering Committee, 2015.

[28] S. Virdhagriswaran and G. Dakin. Camouflaged fraud detection in domains with complex relationships. In *12th KDD*, pages 941–947. ACM, 2006.

[29] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis without aspect keyword supervision. In *17th KDD*, pages 618–626. ACM, 2011.

[30] B. Wu, V. Goel, and B. D. Davison. Propagating trust and distrust to demote web spam. *MTW*, 190, 2006.

[31] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17. IEEE, 2008.

[32] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review*, 36(4):267–278, 2006.

APPENDIX

A. PROOF OF THEOREM 1

PROOF. **Axiom 1 (Node Suspiciousness)**

$$g(\mathcal{S}) = \frac{f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S})}{|\mathcal{S}|} > \frac{f_{\mathcal{V}}(\mathcal{S}') + f_{\mathcal{E}}(\mathcal{S}')}{|\mathcal{S}|} = \frac{f_{\mathcal{V}}(\mathcal{S}') + f_{\mathcal{E}}(\mathcal{S}')}{|\mathcal{S}'|} = g(\mathcal{S}').$$

Axiom 2 (Edge Suspiciousness) Let $e = (u, v)$.

$$g(\mathcal{S}(\mathcal{V}, \mathcal{E} \cup \{e\})) = \frac{f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S}) + c_{uv}}{|\mathcal{S}|} > \frac{f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S})}{|\mathcal{S}|} = g(\mathcal{S}(\mathcal{V}, \mathcal{E})).$$

Axiom 3 (Size) Let $\mathcal{S} = \mathcal{A} \cup \mathcal{B}$, and ρ be the edge density.

$$g(\mathcal{S}) = \frac{f_{\mathcal{V}}(\mathcal{S}) + f_{\mathcal{E}}(\mathcal{S})}{|\mathcal{S}|} = a + b\left(\frac{\rho|\mathcal{A}||\mathcal{B}|}{|\mathcal{A}| + |\mathcal{B}|}\right) = a + b\rho\left(\frac{1}{|\mathcal{A}|} + \frac{1}{|\mathcal{B}|}\right)^{-1}$$

which is increasing in both $|\mathcal{A}|$ and $|\mathcal{B}|$.

Axiom 4 (Concentration)

$$g(S) = \frac{f(\mathcal{S})}{|\mathcal{S}|} > \frac{f(\mathcal{S})}{|\mathcal{S}'|} = \frac{f(\mathcal{S}')}{|\mathcal{S}'|} = g(\mathcal{S}').$$

■