Catching Synchronized Behaviors in Large Networks: A Graph Mining Approach

MENG JIANG and PENG CUI, Tsinghua University
ALEX BEUTEL and CHRISTOS FALOUTSOS, Carnegie Mellon University
SHIQIANG YANG, Tsinghua University

Given a directed graph of millions of nodes, how can we automatically spot anomalous, suspicious nodes judging only from their connectivity patterns? Suspicious graph patterns show up in many applications, from Twitter users who buy fake followers, manipulating the social network, to botnet members performing distributed denial of service attacks, disturbing the network traffic graph. We propose a fast and effective method, CatchSync, which exploits two of the tell-tale signs left in graphs by fraudsters: (a) *synchronized* behavior: suspicious nodes have extremely similar behavior patterns because they are often required to perform some task together (such as follow the same user); and (b) *rare* behavior: their connectivity patterns are very different from the majority. We introduce novel measures to quantify both concepts ("synchronicity" and "normality") and we propose a parameter-free algorithm that works on the resulting synchronicity-normality plots. Thanks to careful design, CatchSync has the following desirable properties: (a) it is *scalable* to large datasets, being linear in the graph size; (b) it is *parameter free*; and (c) it is *side-information-oblivious*: it can operate using only the topology, without needing labeled data, nor timing information, and the like., while still capable of using side information if available. We applied CatchSync on three large, real datasets, *1-billion-edge* Twitter social graph, *3-billion-edge,* and *12-billion-edge* Tencent Weibo social graphs, and several synthetic ones; CatchSync consistently outperforms existing competitors, both in detection accuracy by 36% on Twitter and 20% on Tencent Weibo, as well as in speed.

Categories and Subject Descriptors: H.3.5 [**Information Systems**]: Information Storage and Retrieval - On-line Information Services; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Anomaly detection, suspicious behavior, graph mining, connectivity pattern

This work is supported by National Natural Science Foundation of China, No. 61370022, No. 61210008, and No. 61303075; International Science and Technology Cooperation Program of China, No. 2013DFG12870; National Program on Key Basic Research Project, No. 2011CB302206; NExT Research Center funded by MDA, Singapore, WBS:R-252-300-001-490. Thanks for the support of National Science Foundation, Grant No. CNS-1314632 and IIS-1408924; Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053; U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA), No. W911NF-11-C-0088; a Facebook Fellowship; and the National Science Foundation Graduate Research Fellowship, Grant No. DGE-1252522.

Authors' addresses: M. Jiang, P. Cui, and S. Yang, Department of Computer Science and Technology, Tsinghua University, Beijing, China; emails: mjiang89@gmail.com, cuip@tsinghua.edu.cn, yangshq@tsinghua.edu.cn; A. Beutel and C. Faloutsos, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA; emails: abeutel@cs.cmu.edu, christos@cs.cmu.edu.

DOI: http://dx.doi.org/10.1145/2746403

ACM Reference Format:
Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Catching synchronized behaviors in large networks: A graph mining approach. ACM Trans. Knowl. Discov. Data 10, 4, Article 35 (June 2016), 27 pages.
DOI: http://dx.doi.org/10.1145/2746403

1. INTRODUCTION

Given a directed graph with many millions of nodes, can we tell which nodes are suspicious just based on the graph structure? For many real-world applications, fraudsters try to manipulate networks for personal gain. For example, in social networks, like Twitter's "who-follows-whom" graph, fraudsters are paid to make certain accounts seem more legitimate or popular through giving them many additional followers. The spammers deliver these purchases through either generating fake accounts or controlling real accounts through malware and using them to follow their "customers."[1] This phenomenon creates distorted images of popularity and trustworthiness, with unpleasant or even dangerous effects to honest users.

In this case, the attack is strictly manipulating the Twitter graph to give certain accounts undue credibility. Because the attack only requires adding edges to the graph, previous approaches for finding spam on Twitter that analyze users' tweets and profiles [Aggarwal 2011; Perez et al. 2011; Cao et al. 2012] will often miss this dubious behavior. Rather, we take a strictly graph mining approach, using exclusively the graph structure to find nodes that are suspicious because of their position in the graph.

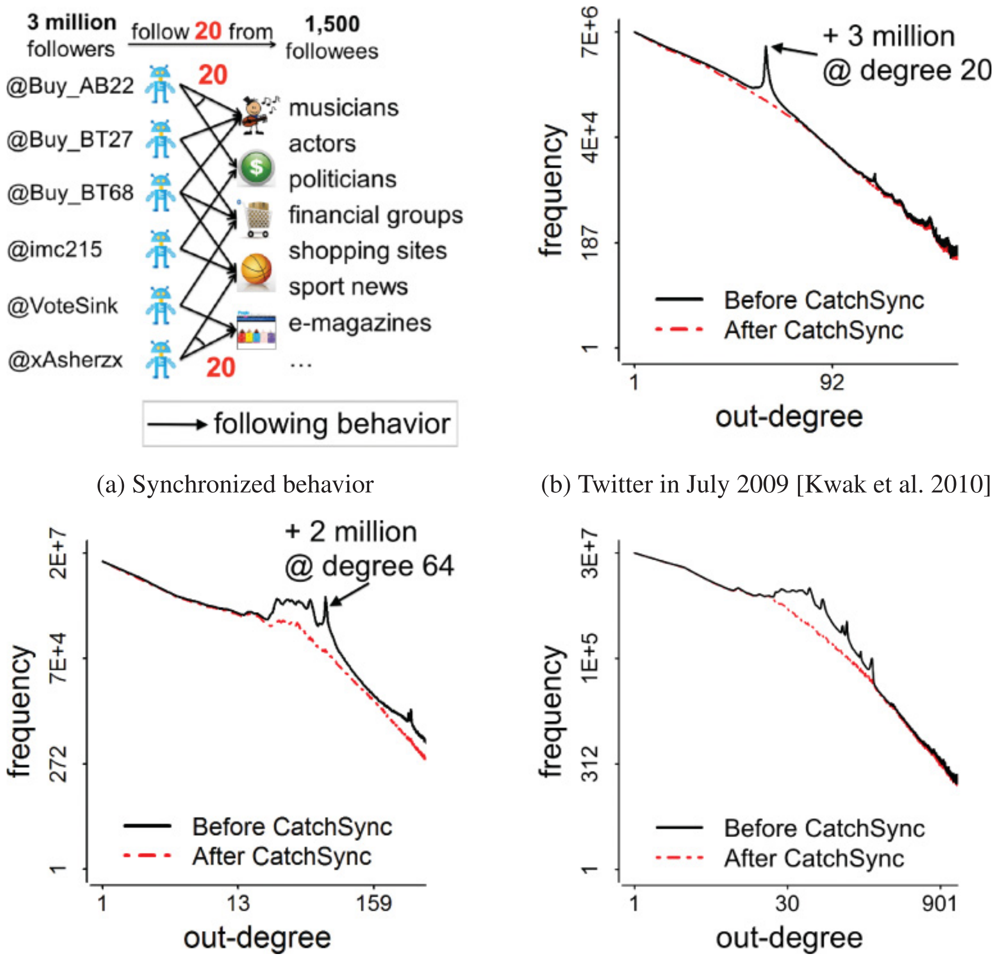
By abstracting the attack to a graph mining problem, we find that it covers a wide variety of suspicious behavior found in the real world. For example, botnets often control hundreds of thousands of machines and use them to perform distributed denial of service (DDOS) attacks on websites, creating a similar pattern in the "who-visits-whom" web traffic graph. Online, on sites like Amazon or Yelp, spammers will create accounts to skew ratings for certain products or places, manipulating edges in the "who-rates-what" graph. On Facebook, Page owners will pay spammers to "Like" their page, distorting the "who-Likes-what" graph.

In this paper, we focus on the Twitter attack, looking for groups of accounts used to unfairly bolster the popularity of their customers. Figure 1(a) illustrates the scenario: it shows a set of suspicious followers and their followees. The followers, all 3 million of them, follow exactly 20 users from the same group of followees, creating a strange, rare connectivity structure. The side information, like the similarity of the login-names (@Buy_AB22, @Buy_BT27, @Buy_BT68), is an extra reason to suspect that they were created by a script.

Our main viewpoint: In more detail, suspicious nodes including suspicious followers and botnets exhibit behavior that is (a) *synchronized* (cause to occur at the same rate): they often connect to the very same 20, 64, or 100 targets and (b) *abnormal / rare*: their behavior pattern is very different from the majority of nodes. In this paper, we propose a fast and effective method, CatchSync, to measure the two properties (*synchronicity* and the *normality*) of a group of nodes; we spot the suspicious nodes and efficiently catch them in the synchronicity-normality plot. We study three real social graphs from Twitter and Tencent Weibo (January 2011 and November 2011) (denoted by TwitterSG, WeiboJanSG, and WeiboNovSG for abbreviation) and use them for evaluations. Note that both have millions of nodes and billions of edges.

Figure 1 gives an elaborate illustration on the effectiveness of CatchSync. As we mentioned earlier, the distributions of the social network data have been seriously distorted by the volume of suspicious followers. Here, we plot the out-degree distribution

[1] Buy Twitter Followers. http://www.buy-followers.org; Buy Twitter Accounts. http://www.buytwitteraccounts.org.



(a) Synchronized behavior

(b) Twitter in July 2009 [Kwak et al. 2010]

(c) Tencent Weibo in January 2011

(d) Tencent Weibo in November 2011

Fig. 1. Suspicious followers and their footprints: (a) We spot synchronized behavior that millions of Twitter accounts follow the same group of followees. (b) Synchronized behavior causes spikes in the out-degree distribution; the distribution becomes smoother after the removal of our suspects. We have the same results on Tencent Weibo graphs in both (c) January 2011 and (d) November 2011.

of TwitterSG, WeiboJanSG, and WeiboNovSG in log–log scale (black lines), which should have smooth, power-law-like distributions [Faloutsos et al. 1999]. However, several spikes appear, which are presumably caused by suspicious followers [Broder et al. 2000]. For example, as shown in Figure 1(a), the 3 million followers on Twitter who connect to exactly 20 users create a spike at out-degree 20 on the distribution in Figure 1(b). After removing the nodes that CatchSync flags as suspicious, the distributions become much smoother and closer to a power law (red lines).

Main contributions: For catching the suspicious behavior in large graphs, our proposed method CatchSync has the following desirable properties:

—*Effectiveness*: it indeed spots groups of source-target groups, with suspicious behavior. We demonstrate the effectiveness with a variety of strong evidence, including

Table I. Compare CatchSync with Existing Approaches

		Catch synchronized?	Parameter free?	Side-info-oblivious?
Proposed CatchSync		√	√	√
Graph-based anomaly detection	AutoPart	×, across k groups	√, auto-decide k	√
	OutRank	×, high scoring nodes	×, threshold	√
	Oddball	×, near-cliques/stars	√	√
	CopyCatch	√, but temporally	×, seeds	×, timestamp
	NetProbe	×, fraudulent nodes	×, propagations	×, committed frauds
Subgraph mining	METIS	×, k equal parts	×, k	√
	SpokEn	×, well-connected	×, eigenvector	√
	DSE	×, d_G-dense	×, density d_G	√
Spammer detection	SPOT	×, spammers	√	×, text features
	SybilRank	×, sybils	×, seeds	×, early non-sybils

CatchSync can catch synchronized behaviors without the limitation of group number, subgraph density, and initial credibilities. It does not require any parameter or side information.

restoring degree distributions and feature space, classification accuracy based on hand-labeled data, and case studies.

—*Scalability:* it is linear in the number of edges, and thus applicable to internet-scale graphs.

—*Parameter free:* the operator does not need to specify any parameter such as the density, the number of groups, and the scale of each group.

—*Side information oblivious:* it needs no side information. It is solely based on topology, and it requires neither a training set, nor labeled nodes, nor node attributes, nor anything else, though it can incorporate the above for better performance.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 gives the problem definition and proposed method, and Section 4 introduces the details of the algorithm. Experiments are given in Section 5. Section 6 concludes the paper.

2. RELATED WORK

There is a significant body on research related to our proposed problem, which we categorize into three groups: graph-based anomaly detection, subgraph mining algorithms, social spammer detection. Table I discusses the majority of them from aspects of effectiveness, parameter setting, and side information, and it shows the advantages of our new approach CatchSync.

2.1. Graph-based Anomaly Detection

Many anomaly detection techniques have been developed based on graphs [Shekhar et al. 2001; Noble and Cook 2003; Chandola et al. 2009]. AutoPart [Chakrabarti 2004] groups similar nodes into clusters, and tags the edges deviating from the overall structure as outliers. However, we often have to face the lack of similarity between suspicious/normal behaviors on the graphs. Some recent works based on graphs propose to detect suspicious nodes and edges by discovering structural anomalies and propagating beliefs for fraudulent nodes [Sun et al. 2005; Chau et al. 2006; Eberle and Holder 2007; Chen et al. 2009]. OutRank [Moonesinghe and Tan 2008] is a random walk-based approach using a graph to detect outliers with similarity of objects. OddBall [Akoglu et al. 2010] gives rules in density, weights, ranks, and eigenvalues that are related to "neighborhood undirected sub-graphs," assuming near-cliques and stars are suspicious. NetProbe [Pandit et al. 2007] uses a list of committed frauds to blame likely fraudulent nodes on the graph with a belief propagation mechanism. CopyCatch [Beutel et al. 2013] detects temporally bipartite cores that are ill-gotten "Likes" of Facebook; however, they require side information such as edge creation time. A guide for lockstep behavior inference [Jiang et al. 2014c] introduces how to catch dense blocks and "staircases" that are formed by different types of lockstep behaviors. However, the fraudsters often perform with camouflage and thus the density is usually not high [Jiang et al. 2014a, 2014b]. Note that our work is orthogonal to the above. We look for synchronized behavior, which forms strange subgraph structure.

2.2. Subgraph Mining Algorithms

A number of subgraph mining algorithms have been investigated [Cook and Holder 2006]. METIS [Karypis and Kumar 1995] is a popular system that can partition the nodes of a graph in k roughly equal parts, such that the number of edges connecting nodes in different parts is minimized. CloseGraph [Yan and Han 2003] works on mining closed frequent graph patterns by pruning methods (reducing unnecessary subgraphs). Crochet [Pei et al. 2005] investigates the problem of mining cross-graph quasi-cliques, where each node in the subgraph is connected to at least a portion γ of the other nodes in the same subgraph. CloseMine [Liu et al. 2005] examines the relationship between the frequent graph-based and the closed frequent graph-based classification. MUSE [Zou et al. 2010] is proposed to mine uncertain subgraph patterns for computing expected supports of a pattern in an uncertain graph database. Recently, community detection has been a hot topic among the modern science of networks. Communities can be considered as fairly independent compartments of a graph [Fortunato 2010; Kumar et al. 2010]. EigenSpokes [Prakash et al. 2010] finds that the singular vectors of graphs exhibit a "spoke" pattern and thus the spoke detection algorithm can spot and extract tightly-knit communities. D-core framework [Giatsidis et al. 2011] for detecting communities characterized by dense connections among their nodes has extended the classic graph-theoretic notion of k-cores for undirected graphs to directed ones. A recent touch in dense subgraph problem takes into account the fact that not every participating node in the graph needs to belong to a community [Chen and Saad 2012]. DSE [Tsourakakis et al. 2013] shows that the well-known densest subgraphs are typically large graphs, with small edge density and large diameter, and presents an approximation algorithm based on a local-search heuristic. One weak point of subgraph mining methods is that they usually require typical parameters like density and number of clusters as input. The other point is that the suspicious nodes can easily evade high-density detection by reducing the number of targets and increasing the volume.

2.3. Social Spammer Detection

There are several recent works on detecting social spammers [Stringhini et al. 2010; Aggarwal 2011]. The deployment of social honeypots [Lee et al. 2010; De Cristofaro et al. 2014] can harvest deceptive spam profiles from social networking communities and for future statistical analysis of their properties for creating spam classifiers. SPOT [Perez et al. 2011] can catch suspicious Twitter profiles, by learning content-based features from text and malicious URLs in tweets and scoring their suspiciousness. SybilRank [Cao et al. 2012] relies on social graph properties to rank users according to their perceived likelihood of being sybils. SSDM [Hu et al. 2013] is a sociological framework that model social network information and content information for spammer detection. Purchased Twitter follower accounts have differences in their interaction and content sharing patterns in comparison to random Twitter users. However, our work is orthogonal to theirs when detecting the group attacks. We notice that the zombie followers on

Table II. Symbols and Definitions

Symbol	Definition and Description
$\mathcal{U}$	The set of nodes
N=$\|\mathcal{U}\|$	The number of nodes
$\mathcal{I}(u)$	The set of node u's sources
$\mathcal{O}(u)$	The set of node u's targets
$d_i(u)$=$\|\mathcal{I}(u)\|$	In-degree of node u (number of sources)
$d_o(u)$=$\|\mathcal{O}(u)\|$	Out-degree of node u (number of targets)
$hub(u)$,$aut(u)$	"Hubness" and "authoritativeness" of u
$sync(u)$	"Synchronicity" of node u's targets
$norm(u)$	"Normality" of node u's targets
$\mathbf{p}(u)$	k-dimensional feature vector of node u
$c(u,v)$	Closeness of u and v in feature space

Twitter or Twitter-like networks usually have very unique but synchronized behavior patterns.

In summary, our new method CATCHSYNC is parameter-free, and side-information oblivious, and effective in detecting suspicious nodes for their synchronized behaviors.

3. SYNCHRONIZED BEHAVIOR DETECTION

In this section, we first propose the problem of synchronized behavior detection and then present a fast and effective solution called CATCHSYNC. We provide not only the algorithm but also theoretical guarantees for the performance.

3.1. Problem Definition

Our goal is to find suspicious nodes on a directed graph and thus the problem is defined as

Given: a directed graph of N nodes in the node set $\mathcal{U}$.

Find: a set of suspicious source nodes (fake followers, botnets, etc.) $\mathcal{U}_{\text{sync}}$, or a set of suspicious target nodes (followees, target hosts, etc.) $\mathcal{V}_{\text{sync}}$, such that the nodes have *synchronized* and *abnormal* connectivity patterns.

"Synchronized" means that the nodes have very similar behavior patterns, and "abnormal" means that their behavior patterns are significantly different from the majority of nodes. Table II gives a list of the symbols we use throughout the paper.

3.2. Proposed Approach

In this section, we introduce our approach towards the above problem. First, we give a feature space for target nodes. Second, we show the definitions of synchronicity and normality that measure the nodes' behavior patterns. Then, we provide a general theorem of the normal shape of the synchronicity-normality plot. Next, we detect the outliers on the plot, which are suspicious nodes with synchronize behavior on the graph.

3.2.1. Feature Space. It has been established by past works that many data mining approaches on graphs benefit from exploiting the features from the nodes' behavior patterns, including (a) out-degree and in-degree, (b) HITS score (hubness and authoritativeness), (c) betweenness centrality, (d) node in-weight and out-weight, if the graph is weighted, (e) the score of the node in the ith left or right singular vector, and many more. We denote k-dimensional feature vector of node u by $\mathbf{p}(u) \in \mathbb{R}^k$. We extract the feature vector from graph structure that somewhat reflects the node's behavior pattern. The features could be *any* from the above and the vector could have *any* dimensionality.

Table III. Plots and Descriptions

Plot	Description
OutF-plot	A heat map of source nodes in feature space: typically, out-degree vs. hubness
InF-plot	A heat map of target nodes in feature space: typically, in-degree vs. authoritativeness
SN-plot	A heat map of source nodes in synchronicity vs. normality of their targets

In this paper, we choose the degree values and HITS score. Besides easy computation and plotting properties, for effectiveness of anomaly detection, we discuss the reasons of using them as follows.

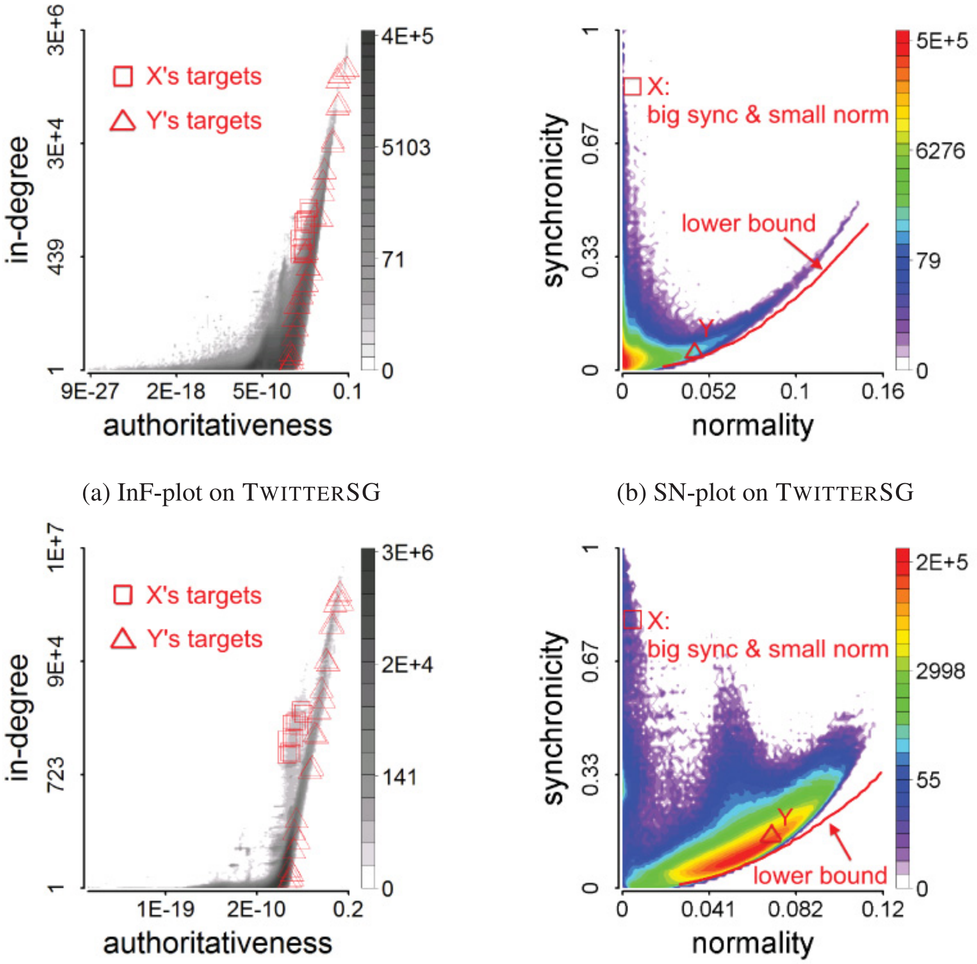
—*In-degree and out-degree (degree values):* We denote a set of u's source nodes by $\mathcal{I}(u)$ and a set of u's target nodes by $\mathcal{O}(u)$. The in-degree $d_i(u)$ of node u is the number of its sources, i.e., the size of $\mathcal{I}(u)$. The out-degree $d_o(u)$ of node u is the number of its targets, i.e., the size of $\mathcal{O}(u)$. In social networks, high in-degree indicates a large number of followers and the customers who benefit from the follower selling services often have similar high in-degree that are bigger than ordinary users, and smaller than real famous celebrities. The botnets, or zombie followers, often have similar out-degree, because it is easy for the services to set the botnets to follow the same number of customers. When the botnets' out-degree value is smaller, they present smarter but still synchronized behaviors. Furthermore, the degree values are related to the spike observation that we have introduced.

—*Hubness and authoritativeness (HITS score):* We denote by $hub(u)$ the hubness of node u and by $aut(u)$ the authoritativeness of u, according to Kleinberg's famous work [Kleinberg 1999]. If we formulate the follower–followee network as an adjacency matrix, the first left-singular vector includes the follower nodes' hubness, and the first right-singular vector includes the followee nodes' authoritativeness. A followee is very authoritative if it has many followers. Followers who connect to "authoritative" celebrities weight more than followers who connect to ordinary users. Zombie followers often have smaller hubness than the users who have the same out-degree, because their customers are not as famous as some top popular celebrities who are often connected to by ordinary users. Customers have smaller authoritativeness than the users who have the same in-degree, because most of their followers are botnets who are not important in the network.

These two clusters of features can indicate the users' behavioral patterns in social networks. As our experiments show, they work well in pin-pointing suspicious nodes. Note that if side information is available, it could be regarded as additional features that would be easily incorporated and, hopefully, improve performance.

Here, we present some plots of the feature spaces. For a source node u, we plot a heat map of the 2D feature space of out-degree $d_o(u)$ vs. hubness $hub(u)$ in log–log scale, called "OutF-plot." Similarly, for a target node u, the heat map of the feature space of in-degree $d_i(u)$ vs. authoritativeness $aut(u)$ in log–log scale is called "InF-plot." Table III summarizes the description of all the plots.

Specifically, Figures 2(a) and (c) are InF-plots of TwitterSG and WeiboJanSG. On TwitterSG, we denote by X one of the suspicious followers we mentioned in Figure 1(a) and by Y an ordinary user whose out-degree is the same as X's. We tag their targets (followees) in the Figure 2(a) and find out that X's targets are coherent in the InF-plot, while Y's targets are not. In other words, X's target nodes have similar in-degree and authoritativeness, but Y's targets are diverse in the feature space, ranging from top



(a) InF-plot on TwitterSG

(b) SN-plot on TwitterSG

(c) InF-plot on WeiboJanSG

(d) SN-plot on WeiboJanSG

Fig. 2. Synchronicity-normality plot: the source nodes X have synchronized and abnormal behavior because their targets are coherent in the InF-plots (a) and (c), while Y's targets are not. X has large synchronicity and small normality in the SN-plots (b) and (d), while Y is near the parabolic, theoretical lower limit.

popular to ordinary users just like Y. Similar thing happens on WeiboJanSG. Figure 2(c) shows that X's targets are located at a micro-cluster which is away from the majority of followee nodes. They have large in-degree values from 1,000 to 100,000 but they are not as authoritative as the ones who are followed by Y and of the same in-degree.

3.2.2. Synchronicity and Normality. We propose two novel concepts to investigate the behavior patterns of the source nodes: (a) "synchronicity" $sync(u)$ to qualify how synchronized the node u's targets are in the feature space (in-degree vs. authoritativeness); and (b) "normality" $norm(u)$ to qualify how normal u's targets are relative to the rest of the data. These two measures consider the relative position of u's target nodes in the feature space. We denote by $\mathbf{p}(v)$ the normalized feature vector of a target node v. We denote by $c(v,v')$ the closeness (similarity) between two target nodes v and v' in the feature space (InF-plot). Thus, we have

$$c(v, v') = \mathbf{p}(v) \cdot \mathbf{p}(v').$$

To quickly compute the closeness of each pair of nodes, we divide the feature space into G grid cells and map each node to a specific grid cell. If two nodes are in the same grid cell, they have similar feature vectors, and they are close in the feature space. Thus, we have

$$c(v, v') = \begin{cases} 1 & \text{if nodes } v \text{ and } v' \text{ are in the same grid cell} \\ 0 & \text{otherwise} \end{cases}.$$

Then, we have the definition of synchronicity and normality.

Definition 3.1 (*Synchronicity and Normality*). We define synchronicity of node u as the average closeness between each pair of u's targets (v,v'):

$$sync(u) = \frac{\sum_{(v,v') \in \mathcal{O}(u) \times \mathcal{O}(u)} c(v, v')}{d_o(u) \times d_o(u)}. \quad (1)$$

We define normality of node u as the average closeness between each pair of u's targets and other nodes (v,v'):

$$norm(u) = \frac{\sum_{(v,v') \in \mathcal{O}(u) \times \mathcal{U}} c(v, v')}{d_o(u) \times N}. \quad (2)$$

Both values of synchronicity and normality range from 0 to 1. We know that a suspicious source node u has uncommonly large $sync(u)$ and abnormally small $norm(u)$:

—*Uncommonly large sync(u):* A large value of $sync(u)$ indicates that in the feature space there has been a large group of nodes that have the same features as the node u. In our settings, if a follower node u has a big $sync(u)$ value, u performs similar "following" behaviors (similar out-degree and similar hubness) as many other nodes, say, connecting to the same group of followee nodes. A zombie follower would give an uncommonly large synchronicity value.

—*Abnormally small norm(u):* A small value of $norm(u)$ indicates that in the feature space the node u is an outlier from the majority of nodes on the graph. In our settings, if a follower node u has a small $norm(u)$ value, u performs very different "following" behaviors (different out-degree and different hubness) from the majority of followers in the social network, say, connecting to a very different group of followee nodes who are rarely connected by the majority of nodes. A zombie follower would give an abnormally small normality value.

For a source node u, we name u's target nodes in the InF-plot as *foreground* points and name all the nodes in the plot as *background* points. We provide a theorem of the normal shape of SN-plot, which could be the basis for catching suspicious nodes. Symbols and their descriptions have been presented in Table IV.

THEOREM 3.2. *For any foreground / background distribution, there is a parabolic lower limit in the synchronicity-normality plot.*

PROOF. In order to find the lower limit of synchronicity when given a normality, we define the problem as follows.

Given: G grid cells; $\vec{f} = <f_1, \ldots, f_G>$ and $\vec{b} = <b_1, \ldots, b_G>$ are probability vectors of counts of foreground and background points, where f_g and b_g are counts of points from the foreground and background cloud in grid cells g ($g = 1, \ldots, G$), and $f_g \leq b_g$; the normality n of $\vec{f}$.

Table IV. Symbols and Descriptions Used in Theorem 3.2

Symbol	Description
G	Number of grid cells in the feature space
$g = 1, \ldots, G$	Grid cell ID in the feature space
F	Total count of foreground points
B	Total count of background points
f_g ($\hat{f}_g$)	(Normalized) count of foreground points in the grid cell g
b_g ($\hat{b}_g$)	(Normalized) count of background points in the grid cell g
$\vec{f}$ ($\vec{\hat{f}}$)	G-length probability vector of (normalized) count of foreground points
$\vec{b}$ ($\vec{\hat{b}}$)	G-length probability vector of (normalized) count of background points
n	"Normality" of the foreground points in $\vec{f}$ compared to background points in $\vec{b}$
s	"Synchronicity" of the foreground points in $\vec{f}$

Find: values of $\vec{f}$ to *minimize* the synchronicity s.

Remind that (1) the synchronicity s is the synchronicity of foreground points, i.e., the dot product of (unit sum) foreground with the same foreground: $s = \sum_g \frac{f_g^2}{F^2}$; (2) the normality n is the dot product of (unit sum) foreground with (unit sum) background: $n = \sum_g \frac{f_g b_g}{FB}$.

Let $B(F)$ be the total counts: $\sum f_g = F$ and $\sum b_g = B$. Let $\hat{b}_g = b_g/B$ and similarly $\hat{f}_g = f_g/F$. Thus, the resulting vectors sum up to one (probability vectors). The problem definition is updated as follows.

Given: a (probability) vector $\vec{\hat{b}}$ with G entries.

Find: a (probability) vector $\vec{\hat{f}}$ with given normality $n = \vec{\hat{f}} \cdot \vec{\hat{b}} = \sum(\hat{f}_g * \hat{b}_g)$ and minimum synchronicity $s = \vec{\hat{f}} \cdot \vec{\hat{f}} = \sum \hat{f}_g^2$.

Report: both the optimal such vector $\vec{\hat{f}}_{\text{opt}}$, as well as the minimum synchronicity $s_{\min}$.

The method of Lagrange multipliers is a well-known strategy for finding the local minima (maxima) of a function subject to equality constraints. Here, the Lagrange function is

$$\mathcal{F}(\hat{f}_g, \lambda, \mu) = \left(\sum_g \hat{f}_g^2\right) + \lambda\left(\sum_g \hat{f}_g - 1\right) + \mu\left(\sum_g \left(\hat{f}_g * \hat{b}_g\right) - n\right). \tag{3}$$

The gradients of the function are

$$\frac{\partial \mathcal{F}}{\partial \hat{f}_g} = 2\hat{f}_g + \lambda + \mu \hat{b}_g = 0 \quad g = 1, \ldots, M, \tag{4}$$

and the two initial conditions are

$$\frac{\partial \mathcal{F}}{\partial \lambda} = \sum_g \hat{f}_g - 1 = 0, \tag{5}$$

$$\frac{\partial \mathcal{F}}{\partial \mu} = \sum_g \left(\hat{f}_g * \hat{b}_g\right) - n = 0. \tag{6}$$

From Equation (4) we have, after summing them all up

$$2 + G\lambda + \mu = 0. \tag{7}$$

From Equation (4) we have, after multiplying each with $\hat{b}_g$ and summing them all up

$$2 * n + \lambda + \mu s_b = 0, \tag{8}$$

where we call s_b the synchronicity of the background: $s_b = \sum_g \hat{b}_g^2 = \sum_g \frac{b_g^2}{B^2}$. Solving for μ, we get

$$\mu = -2 - G\lambda, \tag{9}$$

and substituting μ and for λ, we get

$$\lambda = 2(s_b - n)/(1 - G * s_b). \tag{10}$$

We can substitute the values of μ and λ into Equation (4) and solve for each $\hat{f}_g$, or, even faster, we multiply each of Equation (4) with the corresponding $\hat{f}_g$ and we add, obtaining:

$$2 * s + \lambda + \mu n = 0 \tag{11}$$

which gives that the (optimal) s_{opt} satisfies

$$s_{\text{opt}} = 1/2(-\lambda - \mu n). \tag{12}$$

If the Hessian matrix is positive definite at a point, then the function is a *convex* function and it attains a local *minima* at the point. Here, the Hessian is a diagonal matrix with "2" in the first M positions, and zeros everywhere else. So eventually the minimum synchronicity is

$$\boxed{s_{\min}(n) = (-Gn^2 + 2n - s_b)/(1 - Gs_b)} \tag{13}$$

That is, the minimum synchronicity $s_{\min}$ for a given normality n, is a quadratic function of n. □

Here, we discuss the minimum synchronicity when the normality becomes small and approaches zero. If $n = 0$, we know $s_{\min}(0) = \frac{-s_b}{1-Gs_b}$. Since the synchronicity value of background points, $s_b = \sum_g \hat{b}_g^2 = \sum_g \frac{b_g^2}{B^2} \in [0, 1]$, we have $s_{\min}(0) \in [0, \frac{1}{G}]$. If G is a big number, 100 or 1,000, we know that $s_{\min}(0) \to 0$. The parabolic function is close to a "$y = ax^2$" form.

Figures 2(b) and (d) show the SN-plots of source nodes in TwitterSG and Weibo-JanSG. Note that the source node X has synchronized and abnormal behavior and Y does not, as shown in Figures 2(a) and (c). X has much bigger synchronicity and smaller normality than Y. The red parabola is the theoretical lower limit of synchronicity with a given normality, which has been given in the proof. Y is close to the parabola, while X is far away from the lower bound. The next step to find the suspicious nodes like X is to detect the outliers in the SN-plots.

3.2.3. Outliers in SN-Plot. Using Theorem 3.2, we can now define how to catch the outliers in the SN-plot. Informally, a node is an outlier if it is too far away from the lower limit. Formally, we define the *residual score*, $r_{source}(u)$, as

$$r_{\text{source}}(u) = sync(u) - s_{\min}(norm(u)). \tag{14}$$

The residual score captures how far above the theoretical minimum a node's synchronicity is, and thus indicates how suspicious it is. In detail, given a node u, we compute its synchronicity value $sync(u)$ and normality value $norm(u)$. Given the normality value $norm(u)$, with Equation (13), we know the lower bound of synchronicity value $s_{\min}(norm(u))$. Often an ordinary node has a small synchronicity value, i.e., $sync(u) - s_{\min}(norm(u)) = \epsilon$ (ϵ is a tiny positive number). If the node u has a uncommonly high synchronicity value, say, if $sync(u) \gg s_{\min}(norm(u))$, i.e., $r_{\text{source}}(u) \gg 0$, u is a very suspicious node with synchronized behavioral pattern.

The set of suspicious source nodes $\mathcal{U}_{\text{sync}}$ includes the nodes whose suspiciousness is $\alpha = 3.0$ standard deviations away from the mean:

$$\mathcal{U}_{\text{sync}} \leftarrow \{u : r_{\text{source}}(u) > \mu[r_{\text{source}}] + \alpha \times \sigma[r_{\text{source}}]\}. \tag{15}$$

Note, here $\mu[r_{\text{source}}]$ is the mean residual score for all source nodes, and $\sigma[r_{\text{source}}]$ is the standard deviation of residual scores.

Similarly, we denote by $r_{\text{target}}(v)$ the residual score of a target node v, which captures how suspicious its followers are. Then, we could have the set of suspicious targets $\mathcal{V}_{\text{sync}}$:

$$\mathcal{V}_{\text{sync}} \leftarrow \{v : r_{\text{target}}(v) > \mu[r_{\text{target}}] + \alpha \times \sigma[r_{\text{target}}]\}. \tag{16}$$

The default value of α is chosen according to Tax's classical outlier detection work in Tax and Duin [1998]. In the experimental section, we will validate that the performance of our method is not too sensitive to the choice of α.

4. CATCHSYNC ALGORITHM

In this section, we present the implementation of CATCHSYNC and analyze the complexity.

4.1. Implementation

The approach is outlined below in Algorithm 1. We first derive a feature space for target nodes. We then compute synchronicity and normality of the source nodes' behaviors, according to the relative positions of their target nodes in the feature space. Finally, we use a distance-based outlier detection method to detect the outliers in the synchronicity-normality plot.

ALGORITHM 1: CATCHSYNC: Catch Suspicious Nodes with Synchronized Behaviors in a Large Graph.

```
Input: A directed graph of $N$ nodes in the set $\mathcal{U}$.
Output: A set of source nodes $\mathcal{U}_{\text{sync}}$ who have synchronized and abnormal behaviors and a set
        of targets in $\mathcal{V}_{\text{sync}}$.
Step 1: plot a (2D) feature space of target nodes.
foreach node $v$ as a target do
    Compute in-degree $d_i(v)$ and authoritativeness $aut(v)$.
Give InF-plot $d_i(v)$ vs. $aut(v)$ (See Figures 2(a) and (c)).
Step 2: plot synchronicity-normality of source nodes.
Divide the InF-plot into grids.
foreach node $u$ as a source do
    Compute synchronicity $sync(u)$ and normality $norm(u)$ with Equations (1) and (2).
Give SN-plot $sync(u)$ vs. $norm(u)$ (See Figures 2(b) and (d)).
Step 3:
Adapt a distance-based method to report suspicious sources $\mathcal{U}_{\text{sync}}$ and targets $\mathcal{V}_{\text{sync}}$.
```

To be precise, we choose 2D feature spaces and specifically out-degree vs. hubness, for each source node, and in-degree vs. authoritativeness, for each target node. The hubness (authoritativeness) is the first left- (right-) singular vector of the graph's adjacency matrix. The algorithm to compute these values is omitted for saving space. When we divide the InF-plot into a grid in Step 2, we do so in log-space with base b, such that we specify a side length L and a grid cell goes from b^{aL} to $b^{(a+1)L}$ for integer

a. Setting $L = 1$ and $b = 2$, our grid is partitioned on powers of 2, with hubness and authoritativeness, partitioned by 2^0, 2^{-1}, 2^{-2}, $\ldots$, and with the out-degree and in-degree partitioned by 2^0, 2^1, 2^2, $\ldots$.

4.2. Complexity Analysis

Computing the degree and HITS score of each node is linear in the number of edges E. Second, the process of computing synchronicity and normality is linear in the number of nodes N. We denote by G the number of grid cells that we divided the InF-plot into, dependent on b and L. Therefore, the total time complexity is $O(E + NG)$, and CatchSync is a scalable algorithm, able to process huge, directed graphs.

4.3. Accuracy Guarantee for Speed-Up

We denote by $p(v)$ a *normalized* feature value in the feature vector $\mathbf{p}(v)$, while the distribution of the feature is logarithmic. If the length of a grid cell on this feature axis is from powers of b, 2 as default, and there are two integers a and a' to estimate $p(v)$ (the feature of node v) and $p(v')$ (the feature of node v'), respectively, then

$$b^{aL} \leq p(v) < b^{(a+1)L} \text{ and } b^{a'L} \leq p(v') < b^{(a'+1)L} \text{ with } (a \leq a' < 0).$$

Then, the approximated closeness (similarity) is

$$c(v, v') = \begin{cases} 1 & \text{if } a = a' \\ 0 & \text{otherwise} \end{cases}.$$

Suppose the minimum integer among the nodes to estimate the feature is M, the range of the standard similarity is

$$c_0(v, v') \in [b^{(a'-M)L}/b^{(a+1-M)L}, b^{(a'+1-M)L}/b^{(a-M)L}) = [b^{(a'-a-1)L}, b^{(a'-a+1)L}).$$

So the approximation error is

$$||c(v, v') - c_0(v, v')|| \leq \begin{cases} 0 & \text{if } a = a' \\ b^{(M-|a-a'|)L} & \text{otherwise} \end{cases}.$$

According to the distribution of nodes in the feature space where the pair of nodes usually comes from the same grid, we know that the approximation error is very small.

5. EXPERIMENTS

In this section we present an empirical evaluation of CatchSync, demonstrating its effectiveness in spotting suspicious behavior. Although much of the research on anomaly detection frames the problem as a labeling task, in the real-world anomaly detection is a combination of machine learning, manual verification, and discovering new types of attacks as they arise. Here, we provide evidence that CatchSync is effective at both the classic problem of labeling suspicious behavior, as well as surfacing new patterns of unusual group behavior:

—*Detection effectiveness*: We demonstrate CatchSync's ability to accurately label suspicious behavior and remove anomalies through three techniques.
(a) Injected attacks: We begin by testing the accuracy, precision, and recall on synthetic graphs with injected group attacks. We compare our algorithm against state-of-the-art methods and show that CatchSync performs the best.
(b) Labeling task: We also test our accuracy, precision, and recall on two real datasets, where we use the hand-labeled nodes from a random sample of accounts as ground truth.
(c) Restore normal patterns: For all of these cases, we show that removing the suspicious nodes restores the power-law properties of the graph's degree distribution,

Table V. Synthetic Data

	# of Nodes	# of Injected source nodes	# of Injected target nodes	Camouflage type	Camouflage ratio
Synth-1M	1,034,100	31,000	1,000	–	–
Synth-2M	2,034,100	31,000	1,000	–	–
Synth-3M	3,034,100	31,000	1,000	–	–
Synth-Rand1	3,034,100	31,000	1,000	Random	+10%
Synth-Rand5	3,034,100	31,000	1,000	Random	+50%
Synth-Pop1	3,034,100	31,000	1,000	Popular	+10%
Synth-Pop5	3,034,100	31,000	1,000	Popular	+50%

We inject five different sizes of group attacks with or without camouflage on random power-law graphs of 1 million to 3 million nodes.

Table VI. Real Data from Twitter and Tencent Weibo

	Nodes	Edges	Suspicious/Labeled source nodes
TwitterSG	41,652,230	1,468,365,182	173 / 1,000
WeiboJanSG	117,288,075	3,134,074,580	237 / 1,000
WeiboNovSG	353,509,867	12,168,482,951	

The real-world graphs we study are complete social graphs with multimillion nodes and billion edges. We have used human labor to label a small piece of both of them for ground truth.

which when distorted is a common sign of spam, and remove anomalous patterns in the feature spaces (OutF-plots and InF-plots).

—CatchSync *properties*: We test a number of properties of CatchSync, including the robustness with respect to α, the speed and the scalability.

—*Discovery*: We demonstrate the effectiveness of CatchSync as a discovery tool. We discuss a number of the unusual accounts caught and patterns detected in the Twitter and Weibo datasets.

5.1. Evaluation: Data and Ground Truth

We carry out experiments on synthetic and real datasets to evaluate the performance of CatchSync. The synthetic datasets are described in Table V, while the real datasets are in Table VI.

5.1.1. Synthetic Data.

Description. We generate random power-law graphs, following the Chung–Lu model[2] [Chung and Lu 2002], and with a power-law exponent -1.5 since most real-world networks have been shown to have this value [Faloutsos et al. 1999]. Next, we inject groups of source and target nodes.

To demonstrate the effectiveness, we vary the following properties of the synthetic graphs:

—*Size of graph*: The random power-law graphs we generate contain approximately 1M, 2M, or 3M nodes, named as Synth-1M, Synth-2M, and Synth-3M.

—*Size of injection*: We inject five source and target nodes groups of different sizes. The smallest group has 1,000 new sources, connecting to *20* of 100 new targets, because of the real case that the smallest number of fake followers a user can buy is often 1,000. The size of the injected group doubles one by one (2,000, 4,000, 8,000) and

[2] Following the model, we assign out-degrees $d_o(u)$ and in-degrees $d_i(v)$ to each node u and v respectively; we then create edge (u, v) with probability proportional to $d_o(v)d_i(u)$.

thus the largest group has 16,000 sources and 1,600 targets. The total number of injected sources is 31,000.

—*Camouflage*: The injected source nodes may try to use "camouflage" to evade the detection, for example, the fake accounts can follow *popular* celebrities like President Obama, or some **random** users, though they connect to tens or hundreds of customers. Inspired by this, we try two different techniques on SYNTH-3M: for each injected node, let it connect to (a) some "random," ordinary targets; (b) some from the top 100 "popular" targets. We vary the weights of camouflage d_{camou}: (a) $d_{\text{camou}} = 10\%$, 18 injected targets and 2 for camouflage; (b) $d_{\text{camou}} = 50\%$, 10 injected ones and 10 for camouflage. Specifically, we denote by SYNTH-RAND1 (SYNTH-RAND5) the injected graph with 10% (50%) "random" camouflage, and by SYNTH-POP1 (SYNTH-POP5) the graph with 10% (50%) "popular" camouflage.

With the different settings above, we have the five synthetic datasets.

Evaluation. If we denote the injected nodes by positive samples, and the others by negative samples, we can record the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) rates, which we use the standard definition [Fawcett 2006] to calculate the three popular metrics: accuracy, precision, and recall. High accuracy, precision, and recall will be a better method.

5.1.2. Real Data.

Description. We use our three real world datasets, TWITTERSG, WEIBOJANSG, and WEIBONOVSG, all of which are complete graphs of popular online social networks with billions of edges.

Thanks to the public download links,[3] CATCHSYNC is reproducible on the TWITTERSG data. As the webpage says, due to Twitter's new Terms of Services, academic researchers cannot access the side information like the tweet data. Fortunately, we can usually get the who-follows-whom data, or directed graphs from different applications. Then, we can operate our side-information oblivious method CATCHSYNC.

WEIBOJANSG was crawled in January 2011 from Tencent Weibo, one of the biggest micro-blogging services in China, while WEIBONOVSG was crawled in November 2011 from the same platform. For each dataset CATCHSYNC only uses the graph structure, but we have the user ID and name associated with the nodes, so that we can provide real links to check the users' profile information.

Evaluation. For TWITTERSG, WEIBOJANSG, and WEIBONOVSG, we sample 1,000 nodes and conduct a user study to label them as suspicious or normal accounts. Half of the nodes are randomly selected from the set U_{sync} and half are not. Although the average suspiciousness of samples is higher than that of the entire dataset, it is fair for all the algorithms in our experiments. The 5 volunteers are all 20 to 25-year-old college students who have been social network users for at least 3 years. They are provided URL links directed to the 1,000 users' Twitter or Tencent Weibo pages, and read their tweets and profile information. A user is labeled as a suspicious one if the volunteer finds he or she matches too many of the following clues:

—*Disabled account*: It has been disabled by the services. For example, Weibo user @marra_xiao_bai had 9 followers and 36 followees in 2011. Twitter user @wYWvk0310 had 666 followers and 926 followees in 2010. But both of them have been disabled now.

—*Suspicious user name*: They have strange self-declared information that follows a narrow pattern such as Twitter names in the form of @"Buy_XX##" (@Buy_AB22, @Buy_BT47), Weibo names in the form of "a#####" (@a58444, @a70054).

[3] http://an.kaist.ac.kr/traces/WWW2010.html.

—*Many followees butfew or zero tweets*: It has hundreds of followees but it never posts a single tweet. Twitter user @P8igBg801 had 923 followees in 2010 and @AjaurNYj2 had 869 followees, but both of them post nothing.

—*Malicious tweet content*: The account posts duplicated tweets or malicious links for monetary purposes. For example, Twitter user @Buy_BT66 posts only three messages but all of them are about "bed flat for sale." Weibo account @aa52011 posts hundreds of similar messages about online games.

Finally, we give a user a "suspicious" label if three (more than a half) of the volunteers think it is suspicious. Our task here is to detect the users with the "suspicious" labels. Similarly with the evaluation method on synthetic data, we use accuracy, precision, and recall to evaluate the effectiveness. A good detection algorithm will have high values of accuracy, precision, and recall.

5.2. Competing Algorithms

We carefully implement the following state-of-the-art methods as competing algorithms: (a) OddBall [Akoglu et al. 2010], looking for near-cliques and stars that are suspected as strange nodes in the graph; (b) OutRank [Moonesinghe and Tan 2008], using random walk model across the similarity measure to give the outlierness of each node; (c) SpokEn [Prakash et al. 2010], using pairs of eigenvectors to find well-connected communities. When operating on the labeled real data, we implement a content-based spammer detection method SPOT [Perez et al. 2011], which learns the words and the number of malicious links in the accounts' tweets.

As mentioned before, our CatchSync is orthogonal to the text-based methods like SPOT. Thus, we develop a hybrid method, CatchSync+SPOT, that suspects the nodes detected by *either* CatchSync *or* SPOT. It learns from both the graph structure and text-based features from tweets.

All the algorithms are implemented with Java, and all experiments are performed on a single machine with Intel Xeon CPU at 2.40GHz and 32GB RAM.

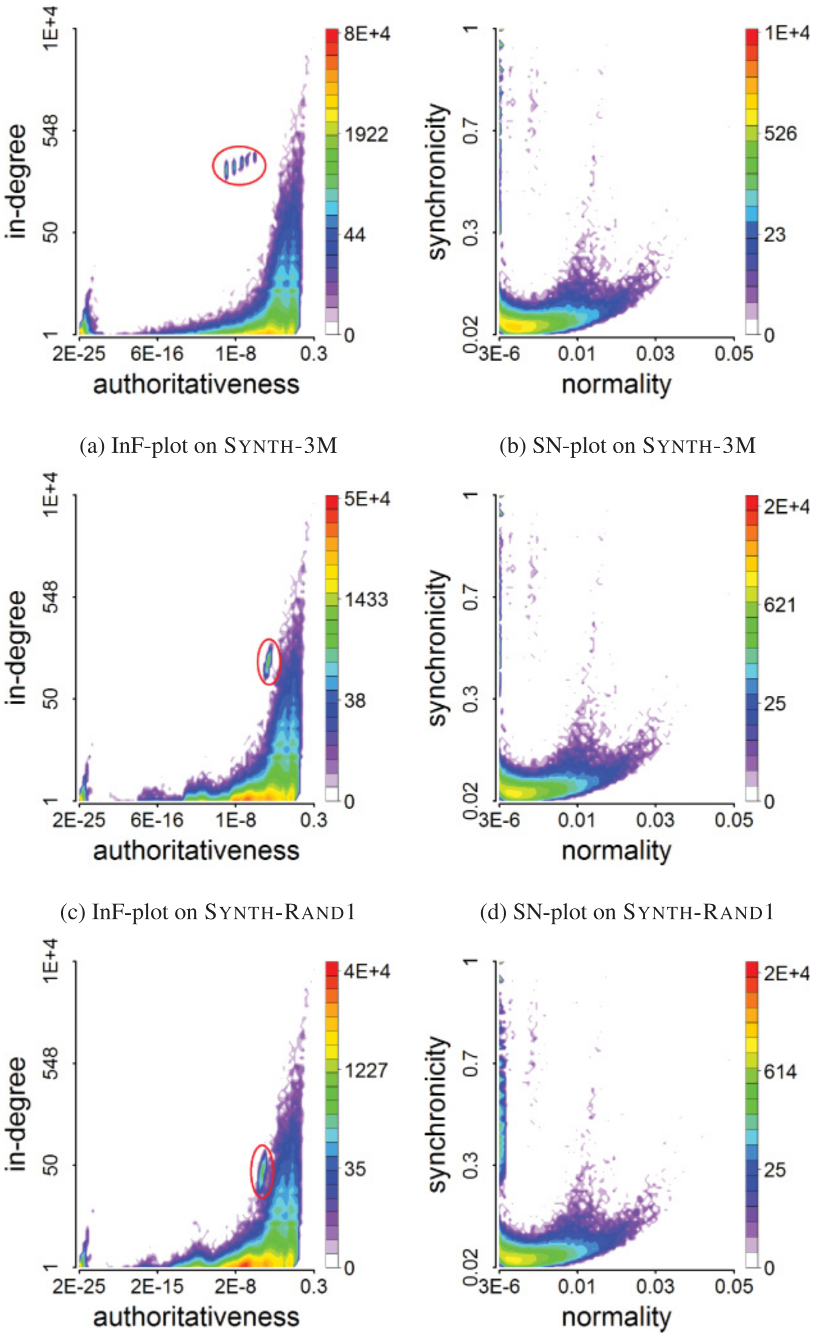
5.3. Detection Effectiveness on Synthetic Data

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method on synthetic data. We first show how we capture injected group attacks in feature space and synchronicity-normality plot. Second, we give accuracy, precision, and recall on detecting the injected nodes. Finally, we show the observation of restoring the power law by removing suspicious nodes.

5.3.1. Injected Group Attacks Shown in Feature Space and SN-Plot. In Figure 3(a), we plot the feature space of synthetic graph Synth-3M. The injected node groups are outliers from the majority. The SN-plot in 3(b) in our method CatchSync can easily catch them since they fall along the synchronicity axis (with almost zero normality and very high synchronicity values). Next in Figures 3 and 4, we show how well the SN-plots work in the cases of "random camouflage" and "popular camouflage."

For random camouflage, when the weight of camouflage becomes larger, from d_{camou}=10% in Synth-Rand1 to d_{camou}=50% in Synth-Rand5, the injected target nodes becomes closer to the dominating part in the InF-plot (See Figures 3(c) and (e)). The SN-plots in Figures 3(d) and (f) show that the injected source nodes still have very large synchronicity and extremely small normality values.

For popular camouflage, when the weight of camouflage becomes larger, from d_{camou}=10% in Synth-Pop1 to d_{camou}=50% in Synth-Pop5, the injected target nodes becomes not only closer to the dominating part in the InF-plot, but merged into the nodes, in other words, invisible for traditional anomaly detection methods (See



(a) InF-plot on SYNTH-3M

(b) SN-plot on SYNTH-3M

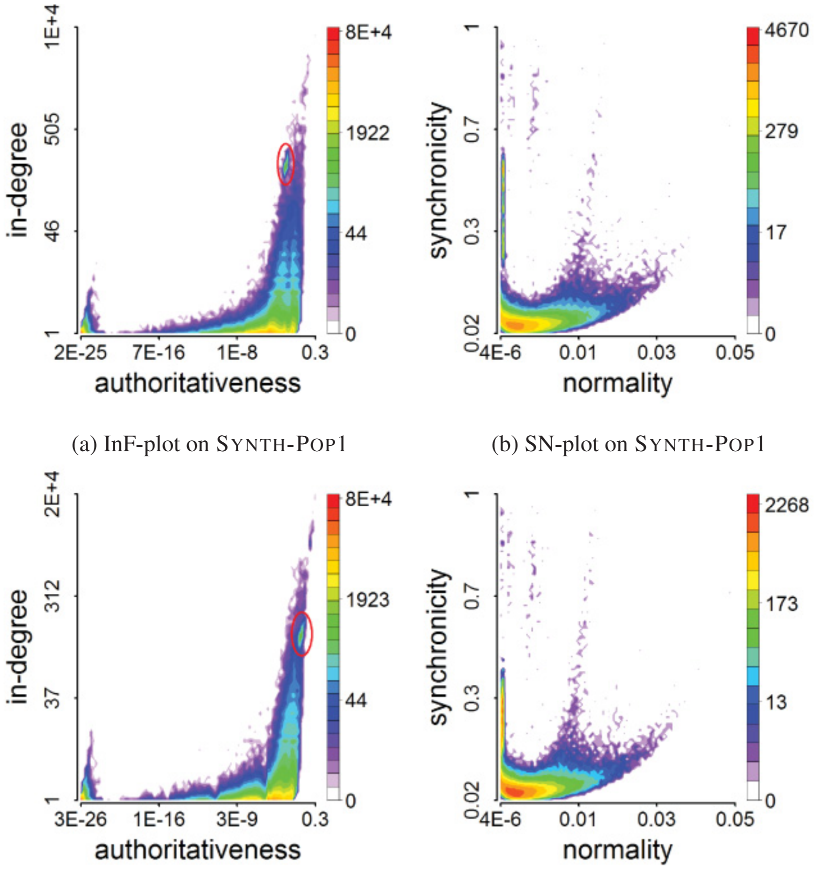
(c) InF-plot on SYNTH-RAND1

(d) SN-plot on SYNTH-RAND1

(e) InF-plot on SYNTH-RAND5

(f) SN-plot on SYNTH-RAND5

Fig. 3. CATCHSYNC can catch injections, despite of "random camouflage": The data set in (a) has no camouflage and the SN-plot in (b) can easily catch the injections. Camouflage can hide the injected nodes in or put them close to the dominating parts in (c) and (e), but the SN-plots still catch them in (d) and (f).



(a) InF-plot on SYNTH-POP1

(b) SN-plot on SYNTH-POP1

(c) InF-plot on SYNTH-POP5

(d) SN-plot on SYNTH-POP5

Fig. 4. CATCHSYNC can catch injections, despite of "popular camouflage": Different from random camouflage, popular camouflage can make the injected nodes too similar with the normal ones to be distinguished with the feature space plot. The injected nodes have been merged into dominating parts in (a) and (c), but still well performing, SN-plots can catch them in (b) and (d).

Figures 4(a) and (c)). The SN-plots in Figures 4(b) and (d) show that the injected source nodes still have very large synchronicity and extremely small normality values, though they have been closer to the majority.

In summary, our method CATCHSYNC can successfully catch the injected group attacks with SN-plots that have smart strategies to hide themselves in InF-plots.

5.3.2. Accuracy, Precision, and Recall on Injected Node Detection. Table VII shows the accuracy on detecting the injected nodes from the three synthetic graphs with 1, 2, and 3 million nodes. When there is no "camouflage," CATCHSYNC reaches greater than 95% accuracy. Table VIII shows the results on synthetic graphs with different weights of "random" or "popular" camouflage. Our method CATCHSYNC can outperform the best of the other methods by 29.6% accuracy on SYNTH-RAND1 and 27.5% accuracy on SYNTH-POP5. Figure 5 plots the precision-recall curves to test the performance of ranking the suspiciousness of nodes. Our method CATCHSYNC (the red-filled triangle) can achieve both higher precision and recall values.

Table VII. CatchSync Consistently Beats Competitors

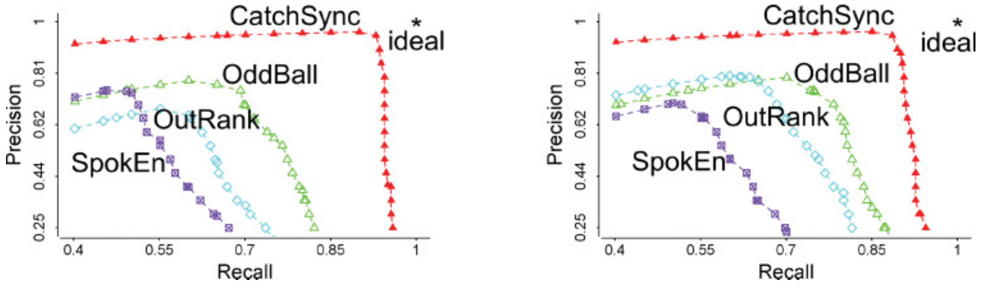
Synthetic graph	Synth-1M	Synth-2M	Synth-3M
CatchSync	**0.998**	**0.987**	**0.956**
OddBall	0.827	0.796	0.755
OutRank	0.805	0.777	0.725
SpokEn	0.695	0.682	0.677

It reaches higher accuracy, nearly 100%, on detecting injected nodes on different sizes of synthetic graphs (1, 2, and 3 million nodes).

Table VIII. CatchSync Consistently Wins, Despite "Camouflage"

Synthetic graph	Synth-Rand1	Synth-Rand5	Synth-Pop1	Synth-Pop5
Camouflage (d_{camou})	10%	50%	10%	50%
CatchSync	**0.910**	**0.764**	**0.885**	**0.792**
OddBall	0.702	0.525	0.657	0.433
OutRank	0.678	0.516	0.694	0.392
SpokEn	0.586	0.470	0.553	0.351

Though the synthetic graphs have been injected with 10% or 50% links as "Random Camouflage" or "Popular Camouflage." Our method CatchSync reaches higher accuracy on detecting injected nodes.



(a) Synth-Rand1

(b) Synth-Pop5

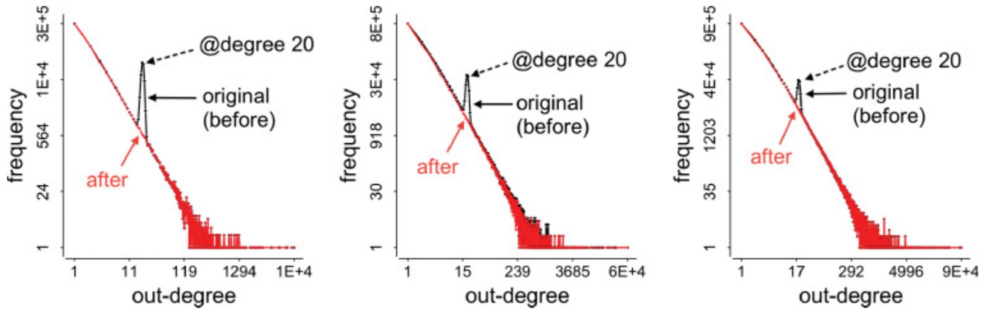
Fig. 5. CatchSync achieves higher precision and recall when detecting injected nodes on synthetic graphs with (a) 10% random camouflage and (b) 50% popular camouflage.

5.3.3. Restoring the Power Law. The injected source nodes connect to 20 from the same set of targets. Due to this anomalous behavior pattern, the out-degree distribution has a spike at degree 20. Figure 6 plots the out-degree distributions before and after we operate CatchSync on the synthetic graphs of different sizes. The spike on the distribution shrinks with the size of the graph increasing. No matter how big the spike is, our method can detect the injected nodes and we see if we remove them and their out-going edges, then the power-law degree distribution is restored.

5.4. Detection Effectiveness on Real Data

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method on real data. We first give accuracy, precision, and recall on detecting frauds in real data. Second, we show how we restore the power law by removing suspicious nodes. Finally, we show our observations on changes in the feature space.

5.4.1. Accuracy, Precision, and Recall on Real Data. Table IX shows the accuracy on detecting the labeled suspicious nodes from the three real social graphs. Also in Figure 7, we plot the precision-recall curves of CatchSync, OutRank, SPOT, and the hybrid



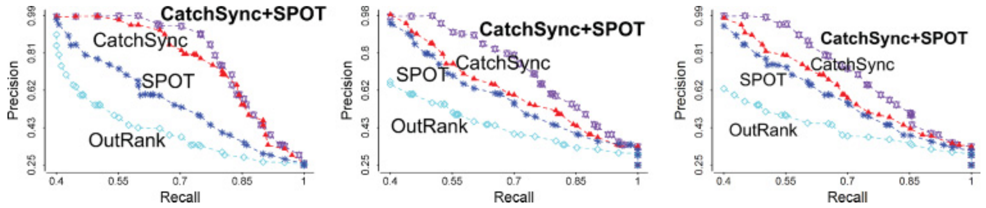
(a) SYNTH-1M (b) SYNTH-2M (c) SYNTH-3M

Fig. 6. CatchSync restores the power law: the degree distribution is recovered after the removal of suspicious nodes.

Table IX. CatchSync+SPOT Outperforms Every Single Other Method

	TwitterSG	WeiboJanSG	WeiboNovSG
CatchSync	0.751	0.694	0.654
OutRank	0.412	0.377	0.336
SPOT	0.597	0.653	0.611
CatchSync+SPOT	**0.813**	**0.785**	**0.709**

CatchSync is better than OutRank at learning the structure, while SPOT learns the text; the combination wins the last.



(a) TWITTERSG (b) WEIBOJANSG (c) WEIBONOVSG

Fig. 7. CatchSync+SPOT is the best at ranking the suspiciousness: it reaches the highest precision and recall when we are detecting suspicious nodes. (Better to be viewed in color.)

algorithm CatchSync+SPOT. We examine the results and give the following observations and explanations.

—CatchSync *out performs* OutRank. CatchSync learns graph-based features of synchronized behavior that OutRank cannot capture using a random walk model with a data dedicated threshold.

—CatchSync *outperforms* SPOT. CatchSync learns graph-based features from the structural information, and SPOT learns text-based features from users' tweets. Since the main characteristics of the suspicious users are group attacks, CatchSync has high accuracy, precision, and recall than SPOT.

Actually, CatchSync is *complementary* to SPOT: combining the flagged nodes, we get even better performance (purple line on Figure 7). The hybrid algorithm uses both of them to catch the different types of attackers. CatchSync+SPOT consistently outperforms the competitor in detection accuracy by 36% on TwitterSG, 20% on WeiboJanSG, and 16% on WeiboNovSG. We analyze CatchSync's False Positives and False Negatives as a clue to further improve the performance of catching suspicious behaviors for social networking services.

—CatchSync*'s FNs can often be caught by* SPOT. CatchSync's false negatives post a large number of messages with links directed to online game or shopping websites. However, they have few followers and followees, and thus, CatchSync, with poor structural information, cannot score their suspiciousness, but SPOT can.

—CatchSync*'s FPs are group attacks that are difficult to label*. The supposedly false positives have common user names and profile information, so the volunteers consider them as normal accounts. However, CatchSync detects their synchronized behaviors and reports that they operate in group, consistently following the same group of users. This demonstrates that group attacks are difficult for humans to label.

We suggest the social network administrators use our CatchSync on their who-follows-whom graphs, while also using methods like SPOT that learn text-based features from tweets and profiles.

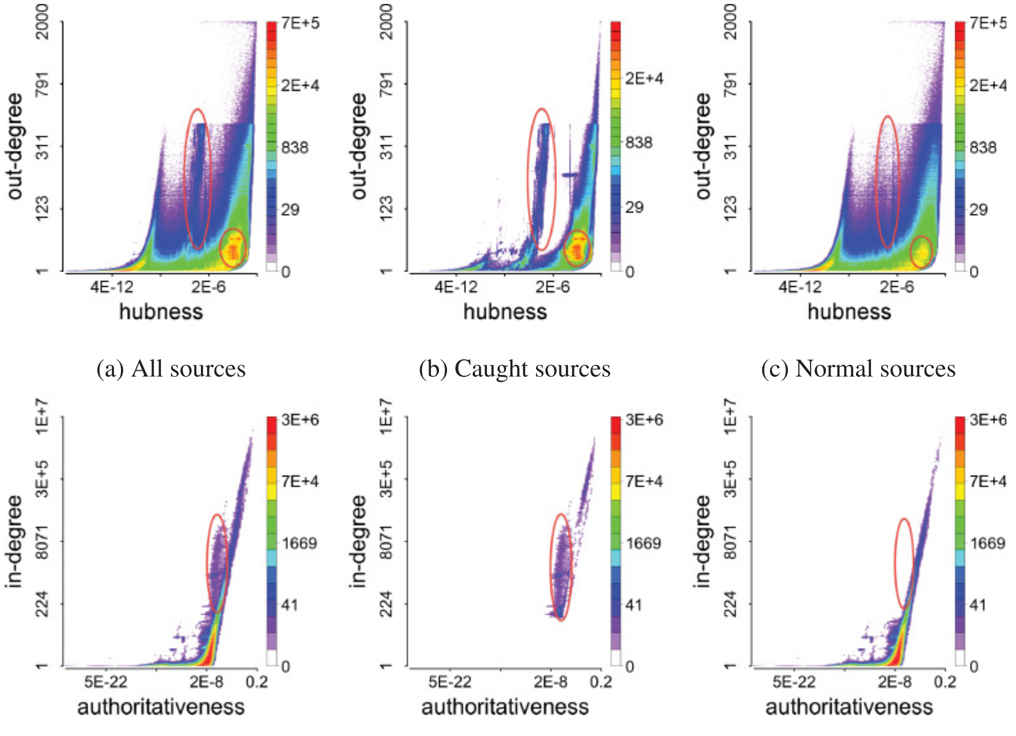
5.4.2. Restoring the Power Law. While in the synthetic datasets the recovery of the power law followed directly from our high recall, this is not necessarily the case on real-world datasets because we can only measure our accuracy on the subset of nodes we label. Looking at the out-degree distribution of TwitterSG in Figure 1(b), WeiboJanSG in Figure 1(c), and WeiboNovSG in Figure 1(c), we see that removing the millions of caught suspicious nodes from the graph does leave only a smooth power-law distribution on the remaining part of the graph. Because a power-law distribution has been found to be typical of social networks and because the original distribution is not directly used in CatchSync, this is strong evidence that our recall on the full datasets is high and that CatchSync is effective.

5.4.3. Observations in the Feature Space. We provide interesting observations from the change of feature space before and after we operate CatchSync on WeiboJanSG. Figure 8(a), (b), and (c) are OutF-plots arranged as an equation: all nodes *minus* suspicious nodes with synchronized behaviors *equals* normal nodes. Figure 8(b) shows the suspicious source nodes look synchronized and abnormal in the OutF-plot: they are coherent in red clusters or on blue stripes that deviate from the majority. The red clusters and blue stripes disappear in Figure 8(c) after we remove them from the graph. Figure 8(d), (e), and (f) show a similar equation of InF-plots. Figure 8(e) shows that the suspicious targets are in a purple cluster in Figure 8(f) the cluster disappears after we remove them. The above observations provide evidence of the suspiciousness of the nodes who have synchronized behaviors. Our method CatchSync can remove the strange patterns in the feature space.

5.5. CatchSync Properties

In this section, we demonstrate properties of our proposed method. First, we show its robustness with respect to parameter α. Next, we show its robustness with respect to the number of grid cells G. Finally, we demonstrate that the method has high speed and scalable to the data.

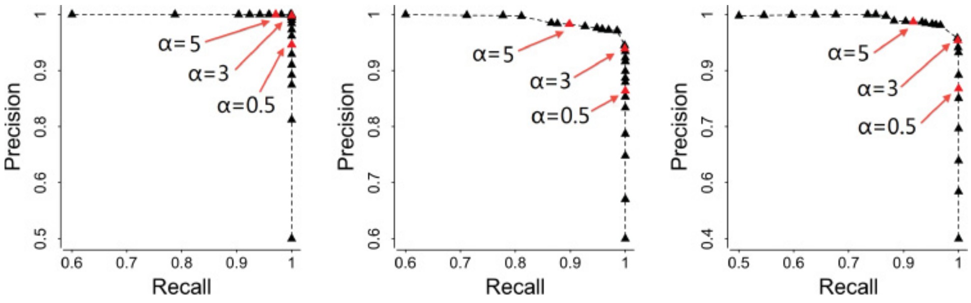
5.5.1. Robustness with Respect to α. Within the synthetic data, we conduct experiments on the robustness to changes in α, the number of standard deviations from the mean for a node to be labeled as suspicious. In short, α=3.0 gives either the best result, or very close to it, and so do nearby values of α. In more detail, we test the sensitivity of precision and recall with respect to α, on the synthetic graphs of three different sizes. Figure 9 plots precision-recall curves: the ideal point is, of course, (1.0, 1.0); although α changes from 0.5 to 5.0, both precision and recall are still over 0.8. The performance



(a) All sources (b) Caught sources (c) Normal sources

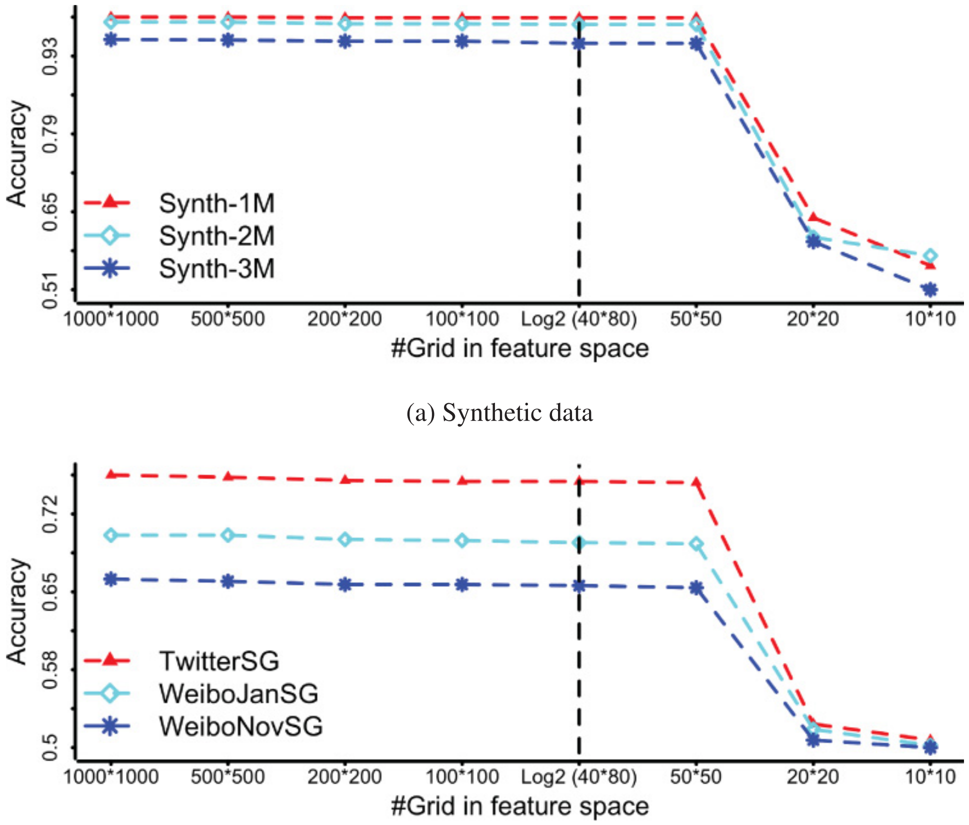
(d) All targets (e) Caught targets (f) Normal targets

Fig. 8. Sources and targets caught by CatchSync are outliers: (a,b,c) and (d,e,f) form two equations of OutF-/InF-plots. (a) minus (b) equals (c); (d) minus (e) equals (f), where (a,d) show all nodes, (b,e) show suspicious nodes and (c,f) show normal ones.



(a) Synth-1M (b) Synth-2M (c) Synth-3M

Fig. 9. Robustness is near ideal: Although the performance of CatchSync is insensitive, we suggest $\alpha = 3.0$ as the default.



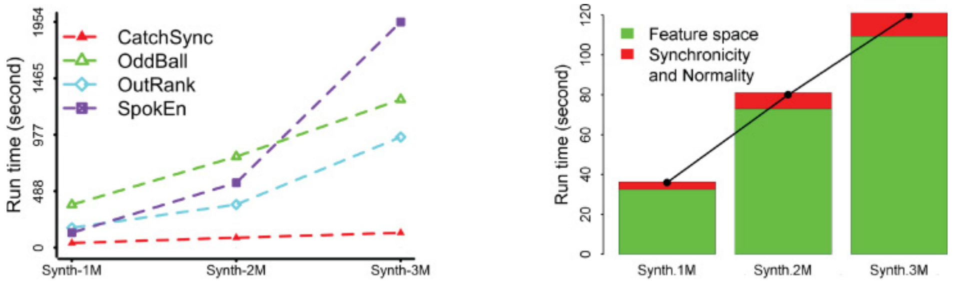
(a) Synthetic data

(b) Real data

Fig. 10. Robustness to grid size: Note that with the number of grid cells G decreasing from $1{,}000^2$ to 500^2, 200^2, ..., the detection performance remains high in the accuracy evaluation. $b = 2$ is a good setting for generally applying CatchSync. However, when G has been as small as 20×20, 10×10, CatchSync suddenly fails to detect the injected nodes and suspicious accounts.

of our algorithm is rather robust on α. We set $\alpha = 3.0$ as the default value for all of our other experiments. Note that approximately 99.7% of the observations fall within three standard deviations of the mean in the normal distribution. The suspicious nodes take the small percentage (0.3%) but still a big number since the graphs often contain millions of nodes, which makes this detection problem rather challenging.

5.5.2. Robustness with Respect to the Number of Grid Cells G. For ensuring the scalability of computing synchronicity and normality values, we make an approximation algorithm inside our method CatchSync. With the number of grid cells in the feature space G decreasing from $1{,}000^2$ to 500^2, 200^2 ..., we test the performance (the accuracy of detecting suspicious nodes). We plot the number of grid cells vs. accuracy in Figure 10(a) for synthetic data including Synth-1M, Synth-2M, and Synth-3M, and in Figure 10(b) for real data including TwitterSG, WeiboJanSG, and WeiboNovSG. We can see the robustness with respect to G when G is larger than 50^2, though the accuracy will decrease a lot when G is smaller than 20^2. Note that if we choose 2 as the default value for b, while in our cases, the default number of grid cells is 40×80, the performance is as good as the complex settings like $G = 1{,}000^2$.



(a) Scalability

(b) Run time in two steps

Fig. 11. CatchSync is fast and scalable: (a) Run time to detect injected nodes as the graph grows. (b) Run time for feature space computation and synchronicity-normality plot.

5.5.3. Speed and Scalability. We measure the run time on synthetic graphs with 1–3 million nodes. Figure 11(a) plots processor time vs. graph size, showing that CatchSync (the red-filled triangles) scales linearly with the graph size and runs faster than alternatives. Figure 11(b) shows the run time in each step. The measures, synchronicity and normality, could be computed very fast, taking only 10% time of the features (degree and HITS score), while the features can be previously chosen and calculated. Therefore, CatchSync performs fast online for large graphs.

5.6. Discovery: A Case Study

As was mentioned earlier, detecting suspicious behavior is not merely a labeling problem. In the real world there are always new types of attacks that arise and distort the service being provided. While we have demonstrated that CatchSync is successful at detecting classic spammy behavior, it also discovers more subtle types of suspicious behavior that a simpler labeling analysis would miss.

Looking online, it is easy to see that fraud on Twitter is much more complex than individual users posting tweets for money. In general, users can get paid to tweet and the amount is based on how many followers they have.[4,5] As a result, this has created marketplaces for buying Twitter followers, which besides providing politicians the appearance of popularity,[6] also raises the value of the "Tweeter." Additionally there are marketplaces[7] to buy and sell Twitter accounts, again with the number of followers being the primary value. Because the market is complex, labeling accounts can be difficult with only subtle red flags raising eyebrows. Here, we examine closer some of the Twitter accounts we caught and we use side information to explain the range of suspicious behavior detected by CatchSync.

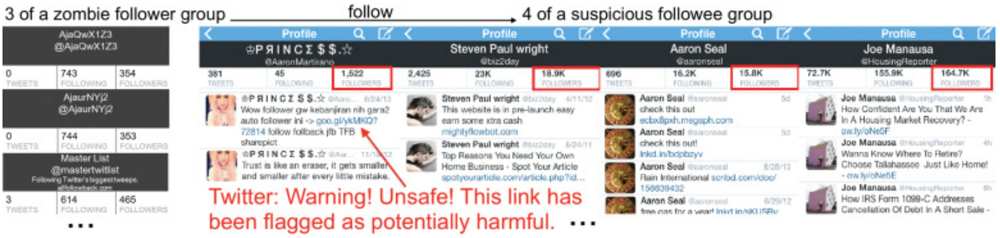
Figure 12 shows a tiny subset (three followers and four followees), from a large, suspicious group of 91K followers and about 700 followees, that was caught by CatchSync. We see three accounts on the left that follow the four accounts on the right (and many others). Overall, each account, on its own, raises a few small suspicions, but collectively these accounts raise many more suspicions. Below we break down the types of accounts we find:

[4] PaidPerTweet - Get Paid For Tweets. http://paidpertweet.com.

[5] Twitter Advertising: Sponsored Tweets. http://sponsoredtweets.com.

[6] Newt Gingrich's Twitter follower count under scrutiny - Faster Forward - The Washington Post. http://www.washingtonpost.com/blogs/faster-forward/post/newt-gingrichs-twitter-follower-count-under-scrutiny/2011/08/02/gIQAVQ33pI_blog.html.

[7] Buy, Sell, and Trade Twitter accounts. http://socialsellouts.com.


3 of a zombie follower group
follow
4 of a suspicious followee group
Profile
Twitter: Warning! Unsafe! This link has been flagged as potentially harmful.
...

AjaQwX1Z3
@AjaQwX1Z3
0 TWEETS | 743 FOLLOWING | 354 FOLLOWERS

AjaurNYj2
@AjaurNYj2
0 TWEETS | 744 FOLLOWING | 353 FOLLOWERS

Master List
@mastertwitlist
Following Twitter's biggest tweeps.
allfollowback.com
3 TWEETS | 614 FOLLOWING | 465 FOLLOWERS

♔PЯINCΣ$$.☆
@AaronMartirano
381 TWEETS | 45 FOLLOWING | 1,522 FOLLOWERS

♔PЯINCΣ$$.☆ @Aaro... 6/24/13
Wow follower gw kebanjiran nih gara2 auto follower ini -> goo.gl/ykMKQ?72814 follow follback jfb TFB sharepict

♔PЯINCΣ$$.☆ @Aar... 11/18/12
Trust is like an eraser, it gets smaller and smaller after every little mistake.

Steven Paul wright
@biz2day
2,425 TWEETS | 23K FOLLOWING | 18.9K FOLLOWERS

Steven Paul wright @biz2day 4/11/12
This website is in pre-launch easy earn some xtra cash mightyflowbot.com

Steven Paul wright @biz2day 8/24/11
Top Reasons You Need Your Own Home Business - Spot Your Article spotyourarticle.com/article.php?id...

Aaron Seal
@aaronseal
696 TWEETS | 16.2K FOLLOWING | 15.8K FOLLOWERS

Aaron Seal @aaronseal 5d
check this out
ecbx8pxh.megaph.com

Aaron Seal @aaronseal 5d
check this out!
lnkd.in/bdpbzyv

Aaron Seal @aaronseal 8/28/13
Rain International scribd.com/doc/156839432

Aaron Seal @aaronseal 6/29/12
free gas for a year! lnkd.in/sKUSRv

Joe Manausa
@HousingReporter
72.7K TWEETS | 155.9K FOLLOWING | 164.7K FOLLOWERS

Joe Manausa @HousingReporter 1h
How Confident Are You That We Are In A Housing Market Recovery? - ow.ly/oNe5F

Joe Manausa @HousingReporter 4h
Wanna Know Where To Retire? Choose Tallahassee Just Like Home! - ow.ly/oNe5E

Joe Manausa @HousingReporter 8h
How IRS Form 1099-C Addresses Cancellation Of Debt In A Short Sale -

Fig. 12. CatchSync at work: using *only* structure information, we examine the biggest group that CatchSync flagged (91,035 followers, 667 followees); we show three of the former and four of the latter. Side information *corroborates* our findings, raising several red flags: (a) the three shown followers have ∼0 tweets, and near-identical counts of followers and followees, (b) the four shown followees mainly tweet urls, one of which is flagged by Twitter as unsafe.

5.6.1. Dedicated Followers. Looking in Figure 12, we see on the left three followers: @AjaQwX1Z3, @AjaurNYj2, and @mastertwitlist. All three accounts have a slightly unusual name, few or no tweets, follow approximately 700 other accounts, and are surprisingly followed by approximately 400 accounts. Alone, each account may look slightly unusual but none of this evidence looks truly incriminating. As a group, however, the accounts are clearly suspicious because, along with them all having the same red flags, they all follow the same group of slightly unusual people.

5.6.2. Surprising Followees. On the right side of Figure 12, we see four of the accounts being followed. Within the group of followees, we find a few common patterns of obviously spam accounts, "SEO experts" tweeting suspicious content, and small business owners with an unusual number of followers. In the first case, @AaronMartirano is slightly suspicious with a most recent gibberish tweet (with words "auto follower," "follback") linking to an empty Blogger that had been labeled "*Unsafe*" by Twitter. Slightly different to the right we see @aaronseal, whose profile offers the GPS coordinates of a Bell Credit Union in Kansas and tweets to free Wordpress themes or asks users to "Like our Facebook Page" for a restaurant. Similarly we observe @biz2day, a self-described "webmaster in the advertising and SEO business," who does not tweet "Unsafe" content, but links to other suspicious content like get-rich-quick schemes. For both @aaronseal and @biz2day, the consistent odd linking raises a red flag, and paired with the synchronized followers suggests that these are possibly purchased tweets and followers were bought to inflate the price. Last we see @HousingReporter a real estate agent with 164,700 followers—more than Massachusetts Senator and U.S. Presidential Hopeful Elizabeth Warren. A small red flag, but of course it is possible for a small business owner to want to appear more popular and thus buy followers.

In all of these cases it is of course impossible to know for sure how their followers were obtained or why they tweet the way they do. However, given that CatchSync found these very different accounts based *only* on the graph structure, all of these other contextual red flags provide strong additional evidence that the followees caught are in fact very suspicious and that CatchSync is effective at catching even subtle or hidden suspicious behavior.

6. CONCLUSION

We propose a novel method called CatchSync that exploits two signs of artificial and non-organic behavior, synchronicity and normality, to automatically report and catch suspicious nodes on large directed graphs. CatchSync has desirable properties

—*Effectiveness*: it spots synchronized behavior and indeed catches suspicious source-target groups.
—*Scalability:* its complexity is linear in the number of edges.

- *Parameter free:* the operator can easily implement the algorithm without specifying any parameters such as the density, the number, and scale of groups.
- *Side information oblivious:* it needs no side information. It is solely based on topology, and it requires neither labeled nodes nor node attributes, though it can incorporate them for better performance.

Experimental results using both real and synthetic datasets demonstrated that CatchSync can catch the suspicious behavior patterns that previous approaches cannot capture.

ACKNOWLEDGMENTS

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, DARPA, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

Charu C. Aggarwal. 2011. *An Introduction to Social Network Data Analytics*. Springer.

Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*. Springer, 410–421.

Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copy-Catch: Stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, IW3C2, 119–130

Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. *Comput. Netw.* 33, 1 (2000), 309–320.

Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *NSDI*. USENIX, 197–210.

Deepayan Chakrabarti. 2004. Autopart: Parameter-free graph partitioning and outlier detection. In *Knowledge Discovery in Databases: PKDD 2004*. Springer, 112–124.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)* 41, 3 (2009), 15.

Duen Horng Chau, Shashank Pandit, and Christos Faloutsos. 2006. Detecting fraudulent personalities in networks of online auctioneers. In *Knowledge Discovery in Databases: PKDD 2006*. Springer, 103–114.

Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han, and S. Yu Philip. 2009. Graph OLAP: A multi-dimensional framework for graph data analysis. *Knowl. Inf. Syst.* 21, 1 (2009), 41–63.

Jie Chen and Yousef Saad. 2012. Dense subgraph extraction with application to community detection. *IEEE Trans. Knowl. Data Eng.* 24, 7 (2012), 1216–1230.

Fan Chung and Linyuan Lu. 2002. The average distances in random graphs with given expected degrees. *Proc. National Acad. Sci.* 99, 25 (2002), 15879–15882.

Diane J. Cook and Lawrence B. Holder. 2006. *Mining Graph Data*. John Wiley & Sons.

Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. 2014. Paying for likes? Understanding facebook like fraud using honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 129–136.

William Eberle and Lawrence Holder. 2007. Discovering structural anomalies in graph-based data. In *7th IEEE International Conference on Data Mining Workshops, 2007 (ICDM Workshops 2007)*. IEEE, 393–398.

Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. 1999. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, Vol. 29. ACM, 251–262.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognit. Lett.* 27, 8 (2006), 861–874.

Santo Fortunato. 2010. Community detection in graphs. *Phys. Rep.* 486, 3 (2010), 75–174.

Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. 2011. D-cores: Measuring collaboration of directed graphs based on degeneracy. In *IEEE 11th International Conference on Data Mining (ICDM 2011)*. IEEE, 201–210.

Xia Hu, Jiliang Tang, Yanchao Zhang, and Huan Liu. 2013. Social spammer detection in microblogging. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. AAAI Press, 2633–2639.

Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014a. CatchSync: Catching synchronized behavior in large directed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 941–950.

Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014b. Detecting suspicious following behavior in multimillion-node social networks. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 305–306.

Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014c. Inferring strange behavior from connectivity pattern in social networks. In *Advances in Knowledge Discovery and Data Mining*. Springer, 126–138.

George Karypis and Vipin Kumar. 1995. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. (1995).

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM (JACM)* 46, 5 (1999), 604–632.

Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2010. Structure and evolution of online social networks. In *Link Mining: Models, Algorithms, and Applications*. Springer, 337–357.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 591–600.

Kyumin Lee, James Caverlee, and Steve Webb. 2010. Uncovering social spammers: Social honeypots+ machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 435–442.

Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and S. Yu Philip. 2005. Mining behavior graphs for" backtrace" of noncrashing bugs. In *SDM*. SIAM, 286–297.

H. D. K. Moonesinghe and Pang-Ning Tan. 2008. Outrank: A graph-based outlier detection framework using random walk. *Int. J. Artif. Intell. Tools* 17, 01 (2008), 19–36.

Caleb C. Noble and Diane J. Cook. 2003. Graph-based anomaly detection. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 631–636.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. 2007. Netprobe: A fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, 201–210.

Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 228–238.

Charles Perez, Marc Lemercier, Babiga Birregah, and Alain Corpel. 2011. Spot 1.0: Scoring suspicious profiles on twitter. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*. IEEE, 377–381.

B. Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Advances in Knowledge Discovery and Data Mining*. Springer, 435–448.

Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. 2001. Detecting graph-based spatial outliers: Algorithms and applications (a summary of results). In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 371–376.

Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2010. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 1–9.

Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. 2005. Neighborhood formation and anomaly detection in bipartite graphs. In *5th IEEE International Conference on Data Mining*. IEEE, 418–425.

David M. J. Tax and Robert P. W. Duin. 1998. Outlier detection using classifier instability. In *Advances in Pattern Recognition*. Springer, 593–601.

Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 104–112.

Xifeng Yan and Jiawei Han. 2003. CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 286–295.

Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010. Mining frequent subgraph patterns from uncertain graph data. *IEEE Trans. Knowl. Data Eng.* 22, 9 (2010), 1203–1218.

Received September 2014; revised March 2015; accepted March 2015