CatchSync : Catching Synchronized Behavior in Large Directed Graphs

Meng Jiang[123], Peng Cui[123], Alex Beutel[4], Christos Faloutsos[4], Shiqiang Yang[123]

[1]Tsinghua National Laboratory for Information Science and Technology
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]Beijing Key Laboratory of Networked Multimedia, Tsinghua University, China
[4]Computer Science Department, Carnegie Mellon University, PA, USA
jm06@mails.tsinghua.edu.cn, cuip@tsinghua.edu.cn, abeutel@cs.cmu.edu
christos@cs.cmu.edu, yangshq@tsinghua.edu.cn

ABSTRACT

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 pattern, 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 on 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, etc., while still capable of using side information, if available. We applied CATCHSYNC on two large, real datasets *1-billion-edge* Twitter social graph and *3-billion-edge* Tencent Weibo social graph, 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

Keywords

Zombie Follower; Anomalous Behavior; Outlier Detection

KDD'14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623632.

1. INTRODUCTION

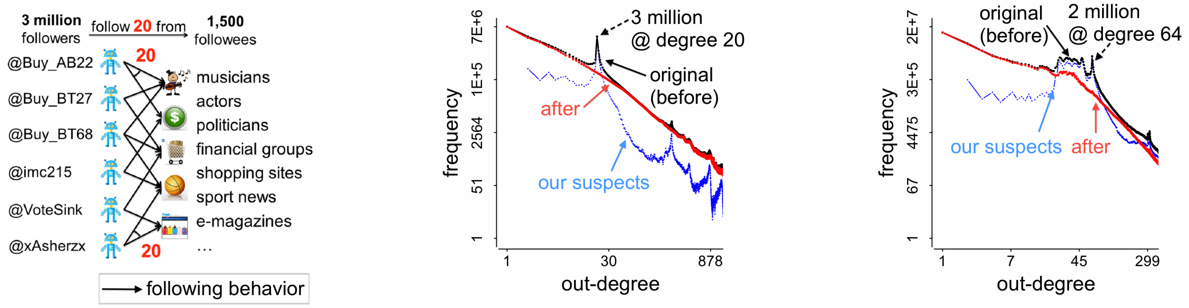
Given a directed graph within millions of nodes, can we tell which nodes are suspicious just based on the graph structure? For many applications, fraudsters try to manipulate networks for personal gain. For example, in social networks, like Twitter's "who-follows-whom" graph, fraudsters are paid to make certain accounts seem more legitimate or famous 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, 2].

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 [4, 32, 9] 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 1a 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 10, 100 or 500 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



(a) Synchronized behavior (b) TWITTERSG (c) WEIBOSG

Figure 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 at out-degree distribution and the distribution becomes smoother after the removal of our suspects. (c) We have the same result on Tencent Weibo.

		Synchronized behaviors?	Parameter free?	No side information?
Proposed CATCHSYNC		√	√	√
Graph-based Anomaly Detection	AUTOPART [10]	×, edges across 2 of k node groups	√, but search for the best k	√
	OUTRANK [28]	×, high scoring nodes	√, but data dedicated threshold	√
	ODDBALL [6]	×, near-cliques or near-stars	√	√
	COPYCATCH [7]	√, temporally synchronized	×, seeds for propagation	×, timestamp
	NETPROBE [30]	×, fraudulent nodes	×, propagation matrix	×, committed frauds
Subgraph Mining	SPOKEN [33]	×, well-connected communities	×, eigenvector computation	√
	DSE [13]	×, d_G-dense subgraphs	×, density d_G	√
Spammer Detection	SPOT [32]	×, twitter spammers	√	×, text and URLs in tweets
	SYBILRANK [9]	×, social sybils	√, seeds for propagation	×, early non-Sybils

Table 1: Compare CATCHSYNC with existing approaches. It does not require any parameter or side information.

them in the synchronicity-normality plot. We study two real social graphs from Twitter and Tencent Weibo (denoted by TWITTERSG and WEIBOSG 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 of TWITTERSG and WEIBOSG in log-log scale, which should have smooth, power-law-like distributions. However, several spikes appear, which are presumably caused by suspicious followers [8]. For example, as shown in Figure 1a, the 3 million followers on Twitter who connect to exactly 20 users create a spike at out-degree 20 on the distribution in Figure 1b. After removing the nodes that CATCHSYNC flags as suspicious (blue points), the distributions become much smoother and closer to a power law (red points).

Main contributions: In short, the proposed method CATCHSYNC has the following desirable properties:

- *Effectiveness*: it indeed spots groups of source-target groups, with suspicious behavior (see Section 5).
- *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 parameters 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.

Organization: We have the usual organization: Survey, problem definition, proposed method, experiments and conclusions.

2. RELATED WORK

There is a significant body on research related to our problem, which we categorize into four groups: graph-based anomaly detection, subgraph mining algorithms, social spammer detection. The majority of them are discussed in Table 1.

Graph-based Anomaly Detection: Many anomaly detection techniques have been developed [5, 34, 11, 23, 22] including discovering structural anomalies [29, 16],propagating beliefs for fake or fraudulent nodes [35, 12, 28]. AUTOPART [10] finds outlier edges across node groups; however, we need to detect suspicious nodes. ODDBALL [6] assumes near-cliques and stars are suspicious; however, these anomalies do not show up in directed graphs. NETPROBE [30] uses a list of committed frauds to blame all the fraudulent nodes on the graph; COPYCATCH [7] detects temporally bipartite cores that are ill-gotten “Likes” of Facebook; however, they require side information such as edge creation time. Note that our work is orthogonal to the above. We look for synchronized behavior, which forms strange subgraph structure.

Subgraph Mining Algorithms: A number of subgraph mining algorithms have been investigated [15], such as mining frequent subgraph patterns [39, 27, 40], mining dense subgraphs [25, 38, 13, 20] and finding quasi-cliques [31, 37]. In addition, some variants of community detection algorithms have been proposed to discover dense clusters [19] and well-connected communities [33]. These require typical parameters like density and number of clusters as input. Suspicious nodes can easily evade high density detection by reducing the number of targets and increasing the volume.

Social spammer detection: There are several recent works on detecting social spammers, for example, fake accounts on microblogging and social networks, mainly using the content-based features [4, 9, 21]. Perez *et al.* [32] proposed SPOT to catch suspicious Twitter profiles, by learning text and malicious URLs in tweets and scoring their suspiciousness. Our work is orthogonal to theirs when detecting the group attacks.

3. SYNCHRONIZED BEHAVIOR DETECTION

In this section, we first propose the problem of synchronized behavior detection and then present a fast and effective solution.

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.) $\mathcal{U}_{sync}$, and a set of suspicious target nodes (followees, target hosts, etc.) $\mathcal{V}_{sync}$ that the source nodes have *synchronized* and *abnormal* behaviors connecting to the target nodes. The word "synchronized" means that the source nodes have very similar behavior pattern, and "abnormal" means that their behavior pattern is very different from the majority of nodes. Table 2 gives a list of the symbols we use throughout the paper.

Symbol	Definition and Description
$\mathcal{U}$	The set of nodes
N=$\|\mathcal{U}\|$	The number of node
$\mathcal{I}(u)$	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

Table 2: Symbols and Definitions

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, (e) node in-weight and out-weight, if the graph is weighted, (f) the score of the node in the i-th 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. In this paper, we choose the degree values and HITS score. 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 $\mathcal{O}(u)$. Also we denote by $hub(u)$ the hubness of node u and by $aut(u)$ the authoritativeness of u, according to Kleinberg's famous work [24]. We choose these features for two reasons: they are fast to compute, as well as easy to plot. As our experiments show, they work well in pin-pointing suspicious nodes. Note that if the side information is available, it could be regarded as additional features that would be easily incorporated, and hopefully, the performance could be better.

Here we present some plots of the feature spaces. For a source node u, we plot a heat map of the 2-D 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 3 summarizes the description of all the plots.

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

Table 3: Plots and Descriptions

Specifically, Figure 2a and 2c are InF-plots of TWITTERSG and WEIBOSG. On TWITTERSG, we denote by *X* one of the suspicious followers we mentioned in Figure 1a and by *Y* an ordinary user whose out-degree is the same as *X*'s. We tag their targets (followees) in the Figure 2a 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 popular to ordinary users just like *Y*. Similar thing happens on WEIBOSG. Figure 2c 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 $c(v,v')$ the closeness (similarity) between two target nodes v and v' in the feature space (InF-plot). For fast computing 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

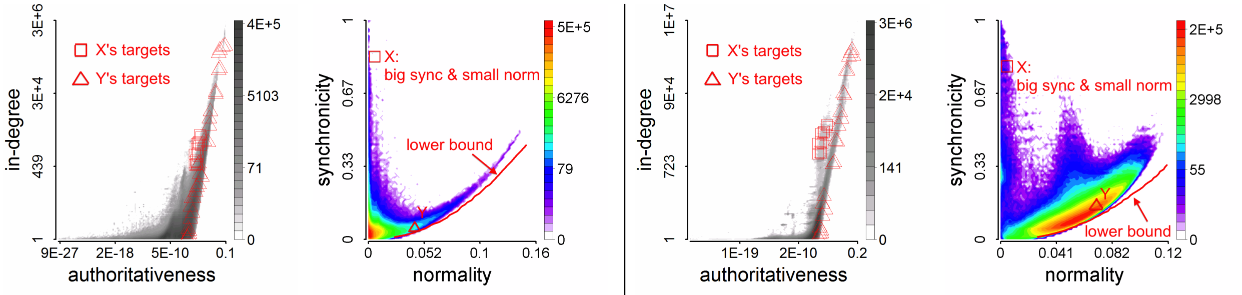
$$c(v, v^{'}) = \begin{cases} 1 & \text{if nodes } v \text{ and } v^{'} \text{ are in the same grid cell} \\ 0 & \text{otherwise} \end{cases}$$

Then we have the definition of synchronicity and normality.

Definition 1 Synchronicity and Normality
We define synchronicity of node u as the average closeness between each pair of u's targets $(v,v^{'})$*:*

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

We define normality of node u as the average closeness between



(a) InF-plot on TwitterSG **(b) SN-plot on TwitterSG** **(c) InF-plot on WeiboSG** **(d) SN-plot on WeiboSG**

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

each pair of u's targets and other nodes (v,v'):

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

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 $norm(u)$. 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

Theorem 1 *For any foreground/background distribution, there is a parabolic lower limit in the synchronicity-normality plot.*

Proof. See appendix. It is based on Lagrange multipliers. The parabolic low limit is

$$s_{min} = (-Mn^2 + 2n - s_b)/(1 - Ms_b)$$

where M is the total number of grid cells, s_{min} is the minimum value of synchronicitiy of foreground points, s_b is the synchronicity of background points, and n is a given normality value. □

Figure 2b and 2d show the SN-plots of source nodes in TwitterSG and WeiboSG. Note that the source node *X* has synchronized and abnormal behavior and *Y* does not, as shown in Figure 2a and 2c. *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

Here we introduce how to catch the outliers in the SN-plot based on Theorem 1. Informally, we want the nodes that are too far away from the lower limit. Formally, we denote by $r_{source}(u)$ the *residual score* of a source node u's synchronicity which indicates how suspicious it is. The set of suspicious source nodes $\mathcal{U}_{sync}$ includes the nodes whose suspiciousness is $\alpha = 3.0$ standard deviations away from the mean:

$$\mathcal{U}_{sync} \leftarrow \{u : r_{source}(u) > \mu[r_{source}] + \alpha \times \sigma[r_{source}]\} \tag{3}$$

Similarly, we denote by $r_{target}(v)$ the suspiciousness of a target node v, which is the proportion of v's sources that are reported in $\mathcal{U}_{sync}$. Then we could have the set of suspicious targets $\mathcal{V}_{sync}$:

$$\mathcal{V}_{sync} \leftarrow \{v : r_{target}(v) > \mu[r_{target}] + \alpha \times \sigma[r_{target}]\} \tag{4}$$

The default value of α is chosen according to Tax's classical outlier detection work in [36]. In the experimental section, we will validate that the performance of our method does not depend much on α.

4. CATCHSYNC ALGORITHM

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

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, directed graph.
Input: A directed graph of $N$ nodes in the set $\mathcal{U}$.
Output: A set of source nodes $\mathcal{U}_{sync}$ who have synchronized
        and abnormal behaviors and a set of targets in $\mathcal{V}_{sync}$ .
Step 1: plot a (2-D) 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 Figure 2a and 2c).
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 Eq. (1) and (2).
Give SN-plot $sync(u)$ vs $norm(u)$ (see Figure 2b and 2d).
Step 3:
Adapt a distance-based method to report suspicious sources
$\mathcal{U}_{sync}$ and targets $\mathcal{V}_{sync}$.
```

In detail, we choose 2-dimensional feature spaces and specifically out-degree vs hubness, for each source node, and in-degree vs authoritativeness, for each target node. The out-degree (in-degree) is the size of set of a source's targets (a target's sources). 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 grids in Step 2, the length of each side of a grid is $log2$, that is, the grid lines at degree and HITS score are on powers of 2.

Complexity analysis. We now examine CatchSync's complexity. We first compute degree and HITS score of each node. This process is linear in the number of edges E. Second, the process of computing synchronicity and normality is linear in the

	Nodes	Edges	Description	Suspicious/Labeled source nodes
TwitterSG	41,652,230	1,468,365,182	Twitter social graph in July 2009 [26]	173 / 1,000
WeiboSG	117,288,075	3,134,074,580	Tencent Weibo social graph in January 2011	237 / 1,000

Table 4: Real data: 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.

number of nodes N. We denote by G the number of grids that we divided the InF-plot into. Thus, the total time complexity is $\mathcal{O}(E+NG)$. CatchSync is a scalable algorithm and able to process huge, directed graphs.

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 labelling task, in 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 labelling 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) Labelling task: We also test our accuracy, precision, and recall on two real datasets, where we use the labeled data from random sampling of TwitterSG and WeiboSG as ground truth of suspicious and normal nodes.
  (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 edge degree, 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 TwitterSG and WeiboSG 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 5, while the real datasets are in Table 4.

	# of Nodes	# of Injected sources	Camouflage
Synth-1M	1,034,100	31,000	-
Synth-2M	2,034,100	=16K	-
Synth-3M	3,034,100	+8K+4K	-
Synth-3M-Rand	3,034,100	+2K+1K	+10% Random
Synth-3M-Pop	3,034,100		+50% Popular

Table 5: Synthetic data: we inject 5 different sizes of group attacks on random power law graphs of 1-3M nodes.

5.1.1 Synthetic data

Description. We generate random power-law graphs, following the Chung-Lu model [1] [14], and with a power-law exponent -1.5 since most real-world networks have been shown to have this value [17]. 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 5 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 [1]. The size of the injected group doubles one by one and 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 Barack Obama, Taylor Swift, 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 also vary the weights of camouflage d_{camou}: (a) d_{camou} = 10%, 18 injected targets and 2 for camouflage; (b) d_{camou} = 50%, 10 injected ones and 10 for camouflage. Specifically, we denote by Synth-3M-Rand the injected graph with 10% random camouflage, and by Synth-3M-Pop the graph with 50% popular camouflage.

With different settings of the above, we have the 5 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 [18] 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 also use our two real world datasets, TwitterSG and WeiboSG, both of which are complete graphs of popular online social networks with billions of edges. Thanks to the public download links [2], CatchSync is reproducible. As the webpage says, due to Twitter's new Terms of Services, we 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.

WeiboSG was crawled in January 2011 from Tencent Weibo, one of the biggest microblogging services in China. For each dataset

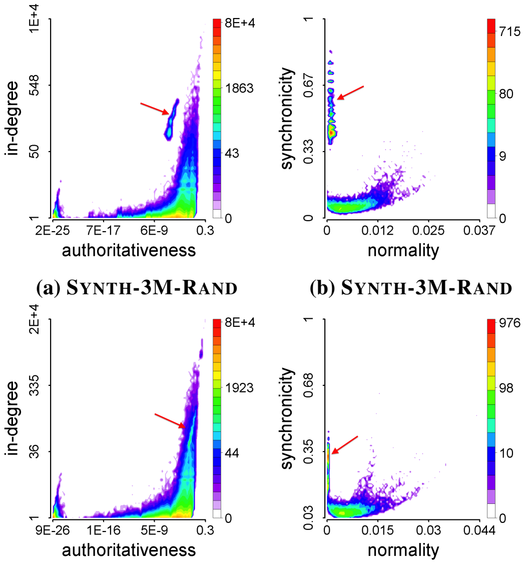
[1] Following the model, we assign out-degrees $d_o(u)$ and in-degrees $d_i(v)$ to each node u and v respectively; we then create edge (u, v) with probability proportional to $d_o(v)d_i(u)$.

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

Synthetic graph	Synth-1M	Synth-2M	Synth-3M	Synth-3M-Rand		Synth-3M-Pop	
Camouflage (d_{camou})	None (0)	None (0)	None (0)	10%	50%	10%	50%
CatchSync	**0.998**	**0.987**	**0.956**	**0.910**	**0.764**	**0.885**	**0.792**
OddBall	0.827	0.796	0.755	0.702	0.525	0.657	0.433
OutRank	0.805	0.777	0.725	0.678	0.516	0.694	0.392
SpokEn	0.695	0.682	0.677	0.586	0.470	0.553	0.351

Table 6: CatchSync consistently wins, despite of "camouflage": it reaches higher accuracy on detecting injected nodes.

CatchSync only uses the graph structure, but we also have user id and name associated with the nodes, so that we can provide real links to check the users' profile information.



(a) Synth-3M-Rand **(b) Synth-3M-Rand**

(c) Synth-3M-Pop **(d) Synth-3M-Pop**

Figure 3: Our CatchSync catches injections, despite of "camouflage": camouflage can hide the injected nodes in or put them close to dominating parts in (a) and (c), but SN-plots can catch them with big synchronicity and small normality in (b) and (d).

Evaluation. For WeiboSG and TwitterSG, we sample 1,000 nodes and conduct 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 1000 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).
- **Many followees but few 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 3 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 3 (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 also use accuracy, precision and recall to evaluate the effectiveness. A good detection algorithm will have high values of accuracy, precision and recall.

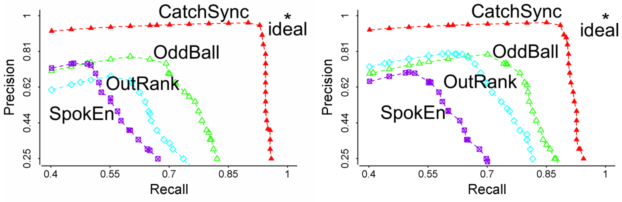
5.2 Competing Algorithms

We carefully implement the following state-of-the-art methods as competing algorithms: (a) OddBall [6], looking for near-cliques and stars that are suspected as strange nodes in the graph; (b) OutRank [28], using random walk model across the similarity measure to give the outlierness of each node; (c) SpokEn [33], 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 [32], 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



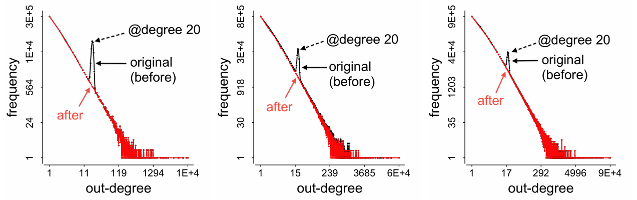
(a) Synth-3M-Rand **(b) Synth-3M-Pop**

Figure 4: CatchSync achieves higher precision and recall.

Injected group attacks shown in feature space and SN-plot. We plot the feature space (InF-plots) and SN-plots of two synthetic graphs with camouflage Synth-3M-Rand and Synth-3M-Pop in Figure 3. When the weight of camouflage is small (d_{camou}=10% in Synth-3M-Rand), the InF-plot in Figure 3a shows the injected node groups as outliers from the majority. With the SN-plot in Figure 3c, CatchSync can easily catch them since they fall along the synchronicity axis. When d_{camou} is as big as 50%, Figure 3c shows that the camouflage can hide the injected nodes in the dominating part. Our SN-plot in Figure 3c can catch them for their big synchronicity and small normality values.

Accuracy, precision and recall on injected node detection. Table 6 shows the accuracy on detecting the injected nodes from all the 5 synthetic datasets. When there is no "camouflage", CatchSync reaches greater than 95% accuracy. When the nodes have camouflage, it can still outperforms the best of the other methods by 29.6% accuracy on Synth-3M-Rand and 27.5% accuracy on Synth-3M-Pop. Figure 4 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 higher recall.

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 5 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.



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

Figure 5: CatchSync restores the power law: the degree distribution is recovered after the removal of suspicious nodes.

5.4 Detection Effectiveness on Real Data

Accuracy, precision and recall on real data. Table 7 shows the accuracy on detecting the labeled suspicious nodes from the two real social graphs. Also in Figure 6, we plot the precision-recall curves of CatchSync, OutRank, SPOT and the hybrid algorithm CatchSync+SPOT. We examine the results and give the following observations and explanations.

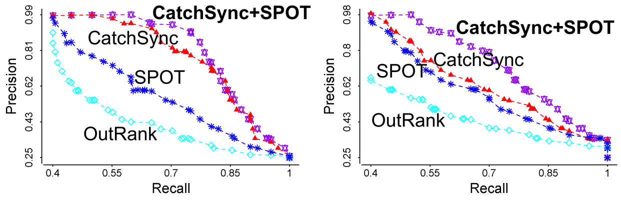
- **CatchSync outperforms OutRank.** CatchSync learns graph-based features of synchronized behavior that OutRank cannot capture using a random walk model with a data dedicated threshold.
- **CatchSync outperforms SPOT.** CatchSync learns graph-based features from the structural information, and SPOT 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 6). The hybrid algorithm uses both of them to catch the different types of attackers. CatchSync+SPOT consistently outperforms the competitors in detection accuracy by 36% on TwitterSG and 20% on WeiboSG. We suggest the social network applications to operate our CatchSync on their who-follows-whom graphs, while they have used methods like SPOT that learns text-based features from their tweets and profiles.

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 data sets because we can only measure our accuracy on the subset of nodes we label. Looking at the out-degree distribution of TwitterSG in Figure 1b and WeiboSG in Figure 1c, 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.

	TwitterSG	WeiboSG
CatchSync	0.751	0.694
OutRank	0.412	0.377
SPOT	0.597	0.653
CatchSync+SPOT	**0.813**	**0.785**

Table 7: CatchSync+SPOT outperforms each part: CatchSync is better than OutRank at learning the structure, while SPOT learns the text; the combination wins the last.



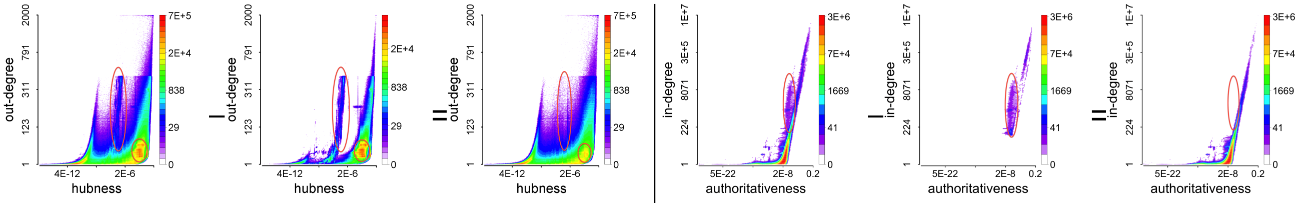
(a) TwitterSG **(b) WeiboSG**

Figure 6: CatchSync+SPOT is the best at ranking the suspiciousness: it reaches the highest precision and recall.

Observations in the feature space. We provide interesting observations from the change of feature space before and after we operate CatchSync on WeiboSG. Figure 7a, 7b and 7c are OutF-plots arranged as an equation: all nodes *minus* suspicious nodes with synchronized behaviors *equals* normal nodes. Figure 7b 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 7c after we remove them from the graph. Figure 7d, 7e and 7f show a similar equation of InF-plots. Figure 7e shows that the suspicious targets are in a purple cluster in Figure 7f 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.

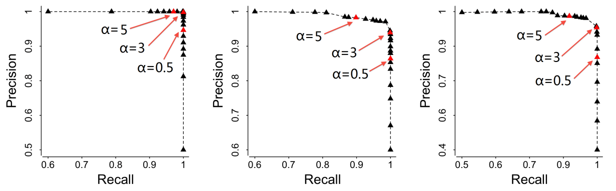
5.5 CatchSync Properties

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 labelled 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 3 different sizes. Figure 8 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 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 3 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.



(a) All sources (b) Caught sources (c) Normal sources (d) All targets (e) Caught targets (f) Normal targets

Figure 7: 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

Figure 8: Robustness is perfect: the performance of CATCHSYNC is rather insensitive. We suggest $\alpha = 3.0$ as default.

Speed and scalability. We measure the run time on synthetic graphs with 1-3 million nodes. Figure 9 plots processor time vs graph size, showing that CATCHSYNC (the red filled triangles) scales linearly with the graph size and runs faster than alternatives. 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.

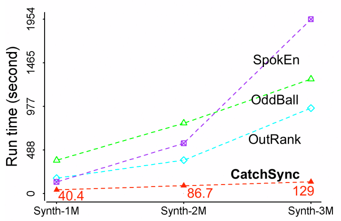


Figure 9: CATCHSYNC is fast and scalable: run time to detect injected nodes as the graph grows.

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 [1, 2]. As a result, this has created marketplaces for buying Twitter followers, which besides providing politicians the appearance of popularity, also raises the value of the "Tweeter." Additionally there are marketplaces, e.g. buytwitteraccounts.org [2] and socialsellouts.com [3] 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 a 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 10 shows a tiny subset (3 followers and 4 followees), from a large, suspicious group of 91K followers and about 700 followees), that was caught by CATCHSYNC. We see 3 accounts on the left that follow the 4 accounts on the right (and many others). Overall, each account, on its own, raises a few small suspicions, but our point is that, collectively, these accounts raise many more suspicions. Below we break down the types of accounts we find:

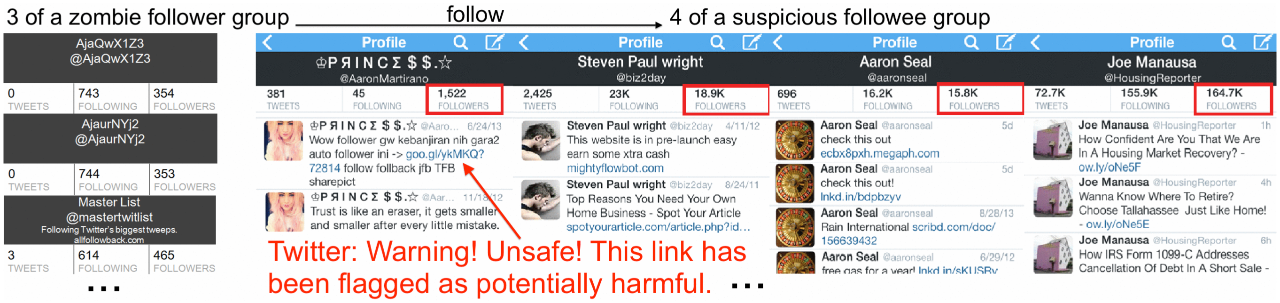
Dedicated Followers: Looking in Figure 10, we see on the left three followers: @AjaQwX1Z3, @AjaurNYj2 and @mastertwitlist. All three accounts have a slightly unusual name, few or no tweets, follow approximately 700 other accounts, and are surprisingly followed by approximately 400 accounts. Alone, each 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.

Surprising Followees: On the right side of Figure 10 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


3 of a zombie follower group
follow
4 of a suspicious followee group
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

Profile

♔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.

Profile

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...

Profile

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/156639432

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

Profile

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 6h
How IRS Form 1099-C Addresses Cancellation Of Debt In A Short Sale -

Figure 10: CATCHSYNC at work: using *only* structure information, we illustrate the biggest group that CATCHSYNC flagged (91,035 followers, 667 followees); we show 3 of the former and 4 of the latter. Side information *corroborates* our findings, raising several red flags: (a) the 3 shown followers have ∼0 tweets, and near-identical counts of followers and followees, (b) the 4 shown followees mainly tweet urls, one of which is flagged by Twitter as unsafe.

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.

7. ACKNOWLEDGEMENT

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, No. CNS-1314632; 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; and the National Science Foundation Graduate Research Fellowship, Grant No. DGE-1252522.

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.

8. REFERENCES

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

[2] Buy Twitter Accounts. http://buytwitteraccounts.org.

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

[4] C. C. Aggarwal. *An introduction to social network data analytics*. Springer, 2011.

[5] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46, 2001.

[6] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *AKDDM*, pages 410–421. 2010.

[7] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW'13*, pages 119–130.

[8] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer networks*, 33(1), 2000.

[9] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI'12*.

[10] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD'04*, pages 112–124.

[11] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15, 2009.

[12] D. H. Chau, S. Pandit, and C. Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. In *PKDD'06*, pages 103–114.

[13] J. Chen and Y. Saad. Dense subgraph extraction with application to community detection. *TKDE*, 24(7):1216–1230, 2012.

[14] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99(25), 2002.

[15] D. J. Cook and L. B. Holder. *Mining graph data*. Wiley-Interscience, 2006.

[16] W. Eberle and L. Holder. Discovering structural anomalies in graph-based data. In *ICDM'07*, pages 393–398.

[17] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM*, 29(4):251–262, 1999.

[18] T. Fawcett. An introduction to roc analysis. *PR letters*, 27(8):861–874, 2006.

[19] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[20] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. D-cores: measuring collaboration of directed graphs based on degeneracy. *KAIS*, 35(2):311–343, 2013.

[21] X. Hu, J. Tang, Y. Zhang, and H. Liu. Social spammer detection in microblogging. In *IJCAI'13*, pages 2633–2639.

[22] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Detecting suspicious following behavior in multimillion-node social networks. In *WWW Companion*, pages 305–306, 2014.

[23] 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*, volume 8443, pages 126–138. 2014.

[24] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[25] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*, pages 337–357. 2010.

[26] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW'10*, pages 591–600.

[27] C. Liu, X. Yan, H. Yu, J. Han, and S. Y. Philip. Mining behavior graphs for "backtrace" of noncrashing bugs. In *SDM'05*.

[28] H. Moonesinghe and P.-N. Tan. Outrank: a graph-based outlier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17(01):19–36, 2008.

[29] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD'03*, pages 631–636.

[30] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW'07*, pages 201–210.

[31] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD'05*, pages 228–238.

[32] C. Perez, M. Lemercier, B. Birregah, and A. Corpel. Spot 1.0: Scoring suspicious profiles on twitter. In *ASONAM'11*, pages 377–381.

[33] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *AKDDM*, pages 435–448. 2010.

[34] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *KDD'01*, pages 371–376, 2001.

[35] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM'05*.

[36] D. M. Tax and R. P. Duin. Outlier detection using classifier instability. In *Advances in Pattern Recognition*. 1998.

[37] C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. A. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *KDD'13*.

[38] X. Wang and I. Davidson. Active spectral clustering. In *ICDM'10*, pages 561–568.

[39] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *KDD'03*, pages 286–295.

[40] Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *TKDE*, 22(9):1203–1218, 2010.

APPENDIX

Here we provide the proof of Theorem 1.

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

Given G grids, f_g and b_g counts of points ($f_g \leq b_g$) from the foreground/background cloud in grid g ($g = 1, \ldots, G$), the normality n of $\vec{f}$, **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 M entries ($M \leq G$), **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 $s = \vec{\hat{f}} \cdot \vec{\hat{f}} = \sum \hat{f}_g^2$, and **report** both the optimal such vector $\vec{\hat{f}}_{opt}$, 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

$$\mathcal{F}(\hat{f}_g, \lambda, \mu) = (\sum_g \hat{f}_g^2) + \lambda(\sum_g \hat{f}_g - 1) + \mu(\sum_g (\hat{f}_g * \hat{b}_g) - n) \quad (5)$$

The gradients of the function are

$$\partial\mathcal{F}/\partial\hat{f}_g = 2\hat{f}_g + \lambda + \mu\hat{b}_g = 0 \quad g = 1, \ldots, M \qquad (6)$$

and the two initial conditions are

$$\begin{aligned} \partial\mathcal{F}/\partial\lambda &= \sum_g \hat{f}_g - 1 = 0 \qquad (7) \\ \partial\mathcal{F}/\partial\mu &= \sum_g (\hat{f}_g * \hat{b}_g) - n = 0 \qquad (8) \end{aligned}$$

From Eq 6 we have, after summing them all up:

$$2 + M\lambda + \mu = 0 \qquad (9)$$

From Eq 6 we have, after multiplying each with $\hat{b}_g$ and summing them all up:

$$2 * n + \lambda + \mu s_b = 0 \qquad (10)$$

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

$$\mu = -2 - M\lambda \qquad (11)$$

and substituting μ and for λ we get

$$\lambda = 2(s_b - n)/(1 - M * s_b) \qquad (12)$$

We can substitute the values of μ and λ into Eq 6 and solve for each $\hat{f}_g$, or, even faster, we multiply each of Eq 6 with the corresponding $\hat{f}_g$ and we add, obtaining:

$$2 * s + \lambda + \mu n = 0 \qquad (13)$$

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

$$s_{opt} = 1/2(-\lambda - \mu n) \qquad (14)$$

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

$$\boxed{s_{min} = (-Mn^2 + 2n - s_b)/(1 - Ms_b)} \qquad (15)$$

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