2017 IEEE International Conference on Data Mining

GANG: Detecting Fraudulent Users in Online Social Networks via Guilt-by-Association on Directed Graphs

Binghui Wang, Neil Zhenqiang Gong
ECE Department, Iowa State University
{binghuiw, neilgong}@iastate.edu

Hao Fu
Microsoft Research Asia, China
fuha@microsoft.com

***Abstract*—Detecting fraudulent users in online social networks is a fundamental and urgent research problem as adversaries can use them to perform various malicious activities. Global social structure based methods, which are known as *guilt-by-association*, have been shown to be promising at detecting fraudulent users. However, existing guilt-by-association methods either assume symmetric (i.e., undirected) social links, which oversimplifies the asymmetric (i.e., directed) social structure of real-world online social networks, or only leverage labeled fraudulent users or labeled normal users (but not both) in the training dataset, which limits detection accuracies.**

In this work, we propose GANG, a *g*uilt-by-*a*ssociation method on directed *g*raphs, to detect fraudulent users in OSNs. GANG is based on a novel pairwise Markov Random Field that we design to capture the unique characteristics of the fraudulent-user-detection problem in directed OSNs. In the basic version of GANG, given a training dataset, we leverage Loopy Belief Propagation (LBP) to estimate the posterior probability distribution for each user and uses it to predict a user's label. However, the basic version is not scalable enough and not guaranteed to converge because it relies on LBP. Therefore, we further optimize GANG and our optimized version can be represented as a concise matrix form, with which we are able to derive conditions for convergence. We compare GANG with various existing guilt-by-association methods on a large-scale Twitter dataset and a large-scale Sina Weibo dataset with labeled fraudulent and normal users. Our results demonstrate that GANG substantially outperforms existing methods, and that the optimized version of GANG is significantly more efficient than the basic version.

I. Introduction

Online social networks (OSNs) have become indispensable platforms for interacting with people, processing information, and diffusing social influence. However, a large number of users on OSNs are *fraudulent*, e.g., spammers, fake users, and compromised normal users. For instance, it was reported that 10% of Twitter users were fake [1]. Adversaries use these fraudulent users to perform various malicious activities such as disrupting democratic election and influencing financial market via spreading rumors [2], [3], distributing malware [4], as well as harvesting private user data. Therefore, detecting fraudulent users is an urgent research problem.

Indeed, this research problem has attracted increasing attention from multiple communities including data mining, cybersecurity, and networking. Depending on the used information sources, we classify existing approaches into two categories, *global structure based methods* [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] and *local feature based methods* [17], [18], [19], [20], [21]. Global structure based methods leverage the global structure of a social graph and are often based on the intuition that a user is likely to be fraudulent (or normal) if it is linked with other fraudulent (or normal) users. In order to stress their application to detecting fraudulent users, we also call these methods *guilt-by-association*. Existing guilt-by-association methods either assume symmetric (i.e., undirected) social links [5], [6], [7], [8], [9], [10], [11], [12], [13], which oversimplifies the asymmetric (i.e., directed) social graph structure in real-world OSNs, or leverage only labeled fraudulent users or normal users (but not both) in the training dataset [14], [15], [16], which limits their detection accuracies. Local feature based methods leverage a user's local subgraph structure (e.g., ego-network) [17], side information (e.g., IP address, behaviors, and content) [18], [19], and possibly combine them with features from the global social structure [20], [21]. A key limitation of these methods is that they are not adversarially robust, i.e., fraudulent users can evade detection via modifying their side information to mimic normal users and colluding to manipulate their local subgraph structures as desired. Indeed, in our experiments, we observe such fraudulent users on Sina Weibo, one of the largest OSNs in China (See Figure 5).

Our work: In this work, we propose GANG, a guilt-by-association method on directed graphs, to detect fraudulent users in OSNs. In GANG, we associate a binary random variable with each user to model its label, and then we design a novel pairwise Markov Random Field (pMRF) to model the joint probability distribution of all these random variables based on the directed social graph. Our pMRF incorporates unique characteristics of the fraudulent-user-detection problem. Specifically, we call an edge (u,v) *unidirectional* if the edge (v,u) in the reverse direction does not exist, otherwise we call the edge *bidirectional*. If two users are linked by bidirectional edges and have the same label, then our pMRF produces a larger joint probability. However, suppose u and v are linked by a unidirectional edge (u,v), e.g., on Twitter, this means that u follows v, but v does not follow back to u. If u is fraudulent or v is normal, then whether the unidirectional edge (u,v) exists or not does not influence the joint probability under our pMRF, otherwise the edge (u,v) makes the joint probability larger. This is because a fraudulent user can follow arbitrary users without being followed back, while a normal user can be followed by arbitrary users without following them back.

In the basic version of GANG, given a training dataset, we use Loopy Belief Propagation (LBP) [22] to estimate the *posterior probability distribution* for each binary random variable and use it to predict label of the corresponding user. However, the basic version has two shortcomings: 1) it is not scalable enough because LBP needs to maintain messages on

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.56

each edge, and 2) it is not guaranteed to converge because LBP might oscillate on loopy graphs [22]. Therefore, we further optimize GANG to address these shortcomings. Our optimizations include eliminating message maintenance and approximating GANG by a concise matrix form. We also derive the conditions for our optimized GANG to converge.

We evaluate GANG and compare it with various existing guilt-by-association methods using a large-scale Twitter dataset (42M users and 1.5B directed edges) and a large-scale Sina Weibo dataset (3.5M users and 653M directed edges). Both datasets have labeled fraudulent and normal nodes. Our results demonstrate that GANG substantially outperforms existing guilt-by-association methods. Via a case study on Sina Weibo, we found that GANG can detect a large amount of fraudulent users that evaded Sina Weibo's detector. Moreover, we demonstrate that the optimized version of GANG is significantly more efficient than its basic version.

In summary, our key contributions are as follows:

- We propose GANG to detect fraudulent users in OSNs via guilt-by-association on directed graphs. GANG leverages a novel pMRF that captures the unique characteristics of the fraudulent-user-detection problem.
- We optimize GANG to make it scalable and convergent.
- We evaluate GANG and various existing guilt-by-association methods using a large-scale Twitter dataset and a large-scale Sina Weibo dataset with labeled fraudulent and normal users. Our results demonstrate that GANG significantly outperforms existing methods, and that the optimized GANG is significantly more efficient than its basic version.

II. Related Work

A. Using Global Graph Structure

These methods [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] often leverage a set of labeled fraudulent nodes and/or labeled normal nodes. Then they *propagate* these label information among the graph to predict labels of the remaining nodes. The key insight of these methods is that a node is fraudulent (or normal) if it links to other fraudulent (or normal) nodes. We call these methods *guilt-by-association* methods in order to stress their application to detecting fraudulent users.

Using directed social graphs: Fraudulent users detection in social networks can be modeled as binary classification on directed graphs. For instance, Twitter's follower-followee network is a directed graph, and detecting fraudulent users can be viewed as a binary classification problem for nodes in the directed graph. To the best of our knowledge, guilt-by-association methods [14], [15], [16] on directed graphs are all based on random walks. Specifically, DistrustRank [15] and CIA [16] assign an initial "badness" reputation score for each node based on a set of labeled fraudulent nodes, while TrustRank [14] assigns an initial "normalness" reputation score for each node based on a set of labeled normal nodes. Then, they use random walks to propagate the reputation scores to the remaining nodes. They also restart the random walks from the initial reputation scores with a certain probability called *restart probability*.

These methods can only leverage labeled normal nodes [14] or labeled fraudulent nodes [15], [16], but not both, which limits their detection accuracies. Our method is a guilt-by-association method on directed graphs, and it can leverage both labeled fraudulent nodes and labeled normal nodes in the training dataset. Our method is based on a novel pairwise Markov Random Field and optimized Loopy Belief Propagation.

Using undirected social graphs: Some guilt-by-association methods [5], [6], [7], [8], [9], [10], [11], [12], [13] assume symmetric relationships between nodes. They often assume the graph satisfies the *homophily* property, i.e., two linked nodes tend to share the same label. In principle, one can apply these methods to detect fraudulent nodes in directed social graphs via transforming them into undirected graphs. However, a directed graph has richer structural information than its undirected version [23]. Transforming a directed graph into an undirected one oversimplifies the graph structure and achieves limited accuracy (as we demonstrate in our experiments). Generally speaking, there are two ways to transform a directed graph to an undirected one. One way is to keep an undirected edge between two nodes once they are connected by directed edge(s). This way is not adversarially robust. In particular, fraudulent nodes can easily inject a large amount of edges with normal nodes in the undirected graph. For instance, on Twitter, a fraudulent user can follow many normal users, all of which will be kept in the undirected graph. As a result, fraudulent nodes well embed in the normal nodes and the undirected graph does not satisfy the homophily property, limiting the detecting accuracy of those guilt-by-association methods. The other way is to keep an undirected edge between two nodes if they are connected via bidirectional edges. However, such transformation cannot leverage unidirectional edges, which are useful for determining reputations of nodes.

B. Using Local Features

Local feature based methods leverage a user's local subgraph structure (e.g., dense subgraphs, a node's hop-2 neighborhood, and a node's ego-network) [17], side information (e.g., IP address, behaviors, and content) [18], [19], and possibly combine them with features from the global social structure [20], [21]. They rely on that fraudulent nodes have abnormal subgraph structures, behavioral analysis, linguistic analysis, and/or sentiment analysis.

A key limitation of these methods is that they are not adversarially robust. Specifically, fraudulent nodes can evade detection of subgraph based methods via creating many fake nodes (e.g., an adversary can create many fake accounts on Twitter [1]) and manipulating links between them to change their subgraph structures as desired. Fraudulent nodes can also modify their side information to mimic normal nodes to evade side information based methods. Indeed, we found such fraudulent nodes in Sina Weibo, and our method can detect them (See Figure 5).

Hooi et al. [24] proposed a dense-subgraph-based method to detect fraudulent accounts. Specially, they aimed to find dense subgraphs and treat nodes in them as fraudulent. Their method has theoretical guarantees on adversarial camouflage. However, their method cannot produce a suspiciousness score for every node and they assume the suspiciousness scores are

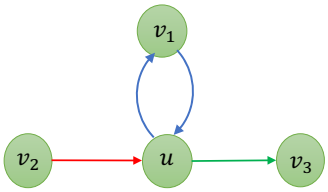


Fig. 1: Illustration of three types of neighbors. v_1, v_2, and v_3 are *bidirectional*, *unidirectional incoming*, and *unidirectional outgoing* neighbors of u, respectively.

given. In practice, OSN providers often hire human workers to manually inspect users and flag fraudulent ones. Producing a suspiciousness score can rank users, which serves as a priority list to aid human workers to find more fraudulent users within the same period of time. Our method produces a suspiciousness score for every user (i.e., the probability that a user is fraudulent).

III. Problem Definition

Suppose we are given a directed social graph $G=(V,E)$, where a node $v \in V$ represents a user, $|V|$ is the number of users, and a directed edge $(u,v) \in E$ indicates a certain relationship between u and v. For instance, such relationship could be that u follows v on Twitter, u sends a friend request to v on Facebook, or u accepts friend request from v on Facebook. Each node can be either *fraudulent* or *normal*. Fraudulent nodes include spammers, fake users, and compromised users.

Definition 1 (Directed Graph based Fraudster Detection)**.** *Suppose we are given a directed social graph and a training dataset consisting of labeled fraudulent and normal nodes. Fraudster detection is to predict the label of each remaining node in the social graph.*

Notations: We call an edge (u,v) *unidirectional* if the edge (v,u) does not exist. We denote by E_1 unidirectional edges in the graph, e.g., $E_1=\{(u,v)|(u,v)\in E \text{ and } (v,u)\notin E\}$. We call an edge (u,v) *bidirectional* if the edge (v,u) also exists. We denote by E_2 bidirectional edges in the graph, i.e., $E_2=\{(u,v)|u<v \text{ and } (u,v)\in E \text{ and } (v,u)\in E\}$. Note that, in our definition, either (u,v) or (v,u) (but not both) appears in E_2 if they are bidirectional edges.

We denote by $\Gamma_b(u)$, $\Gamma_i(u)$, and $\Gamma_o(u)$ the set of *bidirectional*, *unidirectional incoming*, *unidirectional outgoing* neighbors of a user u. Fig. 1 illustrates the three types of neighbors. Formally, we have $\Gamma_b(u)$=$\{v|(v,u)\in E \text{ and } (u,v)\in E\}$, $\Gamma_i(u)=\{v|(v,u)\in E \text{ and } (u,v)\notin E\}$, and $\Gamma_o(u)=\{v|(u,v)\in E \text{ and } (v,u)\notin E\}$. Furthermore, we denote by $\Gamma(u)$ the set of all neighbors of u, i.e., $\Gamma(u)$=$\Gamma_b(u)\cup\Gamma_i(u)\cup\Gamma_o(u)$. Please note that unidirectional incoming neighbors are different from conventional incoming neighbors, and unidirectional outgoing neighbors are different from conventional outgoing neighbors, because conventional incoming neighbors and outgoing neighbors also include bidirectional neighbors.

Table I includes these notations and other important notations used in the paper.

TABLE I: Important notations.

Notation	Explanation
$G=(V,E)$	Directed graph.
E_1	Unidirectional edges.
E_2	Bidirectional edges.
$\Gamma_b(u)$	Bidirectional neighbors of u.
$\Gamma_i(u)$	Unidirectional incoming neighbors of u.
$\Gamma_o(u)$	Unidirectional outgoing neighbors of u.
$\Gamma(u)$	All neighbors of u.
$\mathbf{A}_b$	Bidirectional adjacency matrix.
$\mathbf{A}_i$	Incoming adjacency matrix.
$\mathbf{A}_o$	Outgoing adjacency matrix.
x_u	Binary random variable modeling u.
q_u	Prior probability of u being fraudulent.
$\hat{q}_u$	Residual of q_u, i.e., $\hat{q}_u$=$q_u-0.5$.
p_u	Posterior probability of u being fraudulent.
$\hat{p}_u$	Residual of p_u, i.e., $\hat{p}_u$=$p_u-0.5$.
w	Homophily strength of every edge.
$\hat{w}$	Residual of w, i.e., $\hat{w}$=$w-0.5$.
m_{vu}	Message sent by v to u for $x_u=1$.
$\hat{m}_{vu}$	Residual of m_{vu}, i.e., $\hat{m}_{vu}=m_{vu}-0.5$.

IV. Design of GANG

We introduce a basic version of our GANG. Specifically, we first introduce intuitions on which GANG is based. Second, we design a novel customized pairwise Markov Random Field (pMRF) to capture the intuitions. Third, we discuss how we leverage the pMRF to detect fraudulent nodes.

A. Intuitions

We associate a binary random variable x_u with each user u in the graph, where $x_u=1$ and $x_u=-1$ mean that u is *fraudulent* and *normal*, respectively. We denote by $\bar{x}_u$ the observed label of a node u. We denote by x_S the set of binary random variables associated with the set of vertices in S, and we denote by $\bar{x}_S$ the observed labels of these random variables. In particular, $\bar{x}_{\Gamma(u)}$ are the observed labels of u's neighbors.

Different types of neighbors have different influences on a node u's label. Specifically, we have the following intuitions:

- **Intuition I: Bidirectional neighbors.** If a neighbor v is a bidirectional neighbor of u, then u tends to have the same label with v, e.g., both u and v tend to be fraudulent. This property is known as *homophily*. OSNs with fraudulent and normal nodes have the homophily property because normal nodes will not link to fraudulent nodes with bidirectional edges in most cases. Given the labels of u's bidirectional neighbors, we model the probability that u is fraudulent as the following *sigmoid function*:

$$Pr(x_u=1|\bar{x}_{\Gamma_b(u)})=\frac{1}{1+\exp(-\sum_{v\in\Gamma_b(u)}J_{vu}\bar{x}_v)}, \quad (1)$$

  where J_{vu} is the coupling strength of the edge (v,u), and we set J_{vu}=J_{uv} for bidirectional edges. Moreover, we set $J_{vu}>0$ to model the homophily property. In our model, u has a higher probability to be fraudulent if more of its bidirectional neighbors are fraudulent. We note that using a sigmoid function allows us to capture these intuitions using a customized pMRF.
- **Intuition II: Unidirectional incoming neighbors.** If v is an unidirectional incoming neighbor, then v is not informative for u's label if v is fraudulent. This is because a fraudulent node can link to many other nodes (fraudulent

or normal) in OSNs. For instance, in Twitter, the follower-followee network is a directed graph in which an edge (v,u) means that v follows u, and a fraudulent node could follow many other fraudulent or normal users. Therefore, being linked by a fraudulent node does not mean the node is fraudulent nor normal. However, when v is normal, u also tends to be normal. We model this intuition as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma_i(u)}) = \frac{1}{1+\exp(-\frac{1}{2}\sum_{v\in\Gamma_i(u)} J_{vu}(\bar{x}_v - 1))}, \quad (2)$$

where $J_{vu} > 0$. In our model, unidirectional incoming neighbors, which are known to be fraudulent, do not influence u's label; and u is less likely to be fraudulent if more of its unidirectional incoming neighbors are known to be normal.

- **Intuition III: Unidirectional outgoing neighbors.** If v is an unidirectional outgoing neighbor, then v is not informative for u's label if v is a normal node. This is because any node can link to a normal node in OSNs. For instance, in the Twitter example, any node can follow a normal user. However, if v is fraudulent, then u also tends to be fraudulent because a normal user rarely follows a fraudulent node. We model this intuition as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma_o(u)}) = \frac{1}{1+\exp(-\frac{1}{2}\sum_{v\in\Gamma_o(u)} J_{uv}(\bar{x}_v + 1))}, \quad (3)$$

where $J_{vu} > 0$. In our model, unidirectional outgoing neighbors, which are known to be normal, do not influence u's label; and u is more likely to be fraudulent if more of its unidirectional outgoing neighbors are known to be fraudulent.

Modeling prior knowledge about u's label: We could have some prior knowledge about u's label, which is independent with u's neighbors' labels. We model the prior knowledge as:

$$Pr(x_u = 1) = \frac{1}{1+\exp(-h_u)}, \quad (4)$$

where $h_u > 0$ and $h_u < 0$ indicate that u tends to be fraudulent or normal according to its prior knowledge, respectively; and $h_u = 0$ means that u's prior knowledge is not informative for determining u's label. Such prior knowledge can be obtained through a labeled training dataset (See Section IV-C). Moreover, in practice, we can learn the parameters h_u for each node through feature-based methods, which analyze local graph structure, content, and behaviors.

Unifying neighbor influences and prior knowledge: Suppose we already know the labels of u's neighbors and its prior knowledge, we model the probability that u is fraudulent as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma(u)}) = \frac{1}{1+\exp(-I_b(u) - I_i(u) - I_o(u) - h_u)}, \quad (5)$$

where $I_b(u) = \sum_{v\in\Gamma_b(u)} J_{vu}\bar{x}_v$ is the total influence of bidirectional neighbors, $I_i(u) = \frac{1}{2}\sum_{v\in\Gamma_i(u)} J_{vu}(\bar{x}_v - 1)$ is the total influence of unidirectional incoming neighbors, $I_o(u) = \frac{1}{2}\sum_{v\in\Gamma_o(u)} J_{uv}(\bar{x}_v + 1)$ is the total influence of unidirectional outgoing neighbors, and h_u models the prior knowledge about u.

B. A Novel Pairwise Markov Random Field

We introduce a novel pairwise Markov Random Field (pMRF) to capture our intuitions in Equation 5. A pMRF models the joint probability distribution of all binary random variables x_u for all $u \in V$. We denote by x_V the set of all binary random variables. Our proposed pMRF is as follows:

$$\begin{aligned} H(x_V) = &-\frac{1}{2}\sum_{(u,v)\in E_2} J_{uv}x_u x_v \\ &-\frac{1}{4}\sum_{(u,v)\in E_1} J_{uv}(x_u - 1)(x_v + 1) \\ &-\frac{1}{2}\sum_{u\in V} h_u x_u, \end{aligned} \quad (6)$$

$$Pr(x_V) \propto \exp(-H(x_V)) \quad (7)$$

where $H(x_V)$ is conventionally called *energy function*. Either (u,v) or (v,u), but not both, appears in $H(x_V)$ when (u,v) is a bidirectional edge. We can verify that our pMRF satisfies Equation 5. In particular, when u's neighbors' states are observed, the conditional probability that u is fraudulent is given by Equation 5. Note that an unidirectional edge (u,v) does not influence the value of our energy function nor the joint probability if u is a fraudulent node or v is a normal node. This is because of our Intuition II and III.

Next, we will transform Equation 7 into a product of a set of *node potentials* and *edge potentials*, which is a standard form of a pMRF. This form makes it easier for us to present our method to infer states of nodes. Specifically, we define a node potential $\phi_u(x_u)$ for a node u as

$$\phi_u(x_u) := \begin{cases} q_u & \text{if } x_u = 1 \\ 1 - q_u & \text{if } x_u = -1, \end{cases}$$

where $q_u := (1+\exp\{-h_u\})^{-1}$, which is the prior probability of u being fraudulent. For a bidirectional edge (u,v), we define its edge potential $\varphi_{uv}(x_u,x_v)$ as:

$$\varphi_{uv}(x_u,x_v) := \begin{cases} w_{uv} & \text{if } x_u x_v = 1 \\ 1 - w_{uv} & \text{if } x_u x_v = -1. \end{cases}$$

For an unidirectional edge, we define its edge potential as:

$$\varphi_{uv}(x_u,x_v) := \begin{cases} w_{uv} & \text{if } x_u = 1 \text{ or } x_v = -1 \\ 1 - w_{uv} & \text{otherwise.} \end{cases}$$

$w_{uv} := (1+\exp\{-J_{uv}\})^{-1}$ in both definitions of edge potentials. $w_{uv} > 0.5$ captures the homophily property. In our definitions, w_{uv} can be interpreted as the probability that two nodes have the same label when they are linked via bidirectional edges. In this work, we set $w_{uv} = w > 0.5$ for all edges, and we call w *homophily strength*. However, learning the parameters w_{uv} would be a valuable future work.

With node potentials and edge potentials, we can rewrite Equation 7 as follows:

$$Pr(x_V) = \frac{1}{Z}\prod_{v\in V}\phi_v(x_v)\prod_{(u,v)\in E_1\cup E_2}\varphi_{uv}(x_u,x_v), \quad (8)$$

where $Z = \sum_{x_V}\prod_{v\in V}\phi_v(x_v)\prod_{(u,v)\in E_1\cup E_2}\varphi_{uv}(x_u,x_v)$ is conventionally called the partition function and normalizes the probabilities. Note that either (u,v) or (v,u), but not both, appears in the above equation if (u,v) is a bidirectional edge.

C. Detecting Fraudulent Nodes

We leverage the above pMRF to detect fraudulent nodes. Suppose we are given a set of labeled fraudulent nodes denoted as L_f and a set of labeled normal nodes denoted as L_n. We set the parameter q_u in node potentials as follows:

$$q_u = \begin{cases} 0.5+\theta & \text{if } u \in L_f \\ 0.5-\theta & \text{if } u \in L_n \\ 0.5 & \text{otherwise,} \end{cases} \tag{9}$$

where $0 < \theta \leq 0.5$. Then, we compute the posterior probability distribution of a node u, i.e., $Pr(x_u) = \sum_{x_{V/u}} Pr(x_V)$. For simplicity, we denote by p_u the posterior probability that u is a fraudulent node, i.e., $p_u = Pr(x_u = 1)$. We predict u to be fraudulent if $p_u > 0.5$, otherwise we predict u to be normal.

Computing posterior probability distribution using Loopy Belief Propagation (LBP): In the basic version of GANG, we use LBP [22] to estimate the posterior probability distribution $Pr(x_u)$. LBP iteratively passes messages between neighboring nodes in the graph. Specifically, the message $m_{vu}^{(t)}(x_u)$ sent from v to u in the tth iteration is

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v,x_u) \prod_{k\in\Gamma(v)/u} m_{kv}^{(t-1)}(x_v), \tag{10}$$

where $\Gamma(v)/u$ is the set of all neighbors of v, except the receiver node u. This encodes that each node forwards a product over incoming messages of the last iteration and adapts this message to the respective receiver based on the homophily strength with the receiver. LBP stops when the changes of messages become negligible in two consecutive iterations (e.g., l_1 distance of changes becomes smaller than 10^{-3}) or it reaches the predefined maximum number of iterations T. After LBP halts, we estimate the posterior belief $Pr(x_u)$ as follows:

$$Pr^{(t)}(x_u) \propto \phi_u(x_u) \prod_{k\in\Gamma(u)} m_{ku}^{(t)}(x_u), \tag{11}$$

which is equivalent to

$$p_u^{(t)} = \frac{q_u \prod_{k\in\Gamma(u)} m_{ku}^{(t)}}{q_u \prod_{k\in\Gamma(u)} m_{ku}^{(t)} + (1-q_u)\prod_{k\in\Gamma(u)}(1-m_{ku}^{(t)})}, \tag{12}$$

where $m_{ku}^{(t)} = m_{ku}^{(t)}(x_u = 1)$ and $1 - m_{ku}^{(t)} = m_{ku}^{(t)}(x_u = -1)$. Note that normalizing $m_{ku}^{(t)}(x_u)$ does not affect the computation of posterior probability distribution of any node. Therefore, for simplicity, we have normalized $m_{ku}^{(t)}(x_u)$ such that $m_{ku}^{(t)}(x_u = 1)$ + $m_{ku}^{(t)}(x_u = -1) = 1$ in the above equation.

V. Optimizing GANG

The basic version of GANG has two shortcomings: 1) GANG is not scalable enough, and 2) GANG is not guaranteed to converge. These shortcomings are caused by LBP which estimates the posterior probability distribution for each node. Specifically, LBP is not scalable enough because it maintains messages on each edge, and LBP might oscillate on loopy graphs [22]. In this section, we optimize GANG to address these shortcomings.

A. Eliminating Message Maintenance

One of the major reasons why GANG is not scalable enough is that LBP maintains messages on each edge. We observe that the key reason why LBP needs to maintain messages on edges is that when a node v prepares a message to its neighbor u, it needs to exclude the message that u sends to v. Therefore, our first optimization step is to include the message that u sends to v when v prepares its message for u. Formally, we modify Equation 10 as follows:

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v,x_u) \prod_{k\in\Gamma(v)} m_{kv}^{(t-1)}(x_v). \tag{13}$$

Considering Equation 11, we have:

$$m_{vu}^{(t)}(x_u) \propto \sum_{x_v} \varphi_{vu}(x_v,x_u) Pr^{(t-1)}(x_v). \tag{14}$$

Recall that we normalize $m_{vu}^{(t)}(x_u)$ such that $m_{vu}^{(t)}(x_u = 1)$ + $m_{vu}^{(t)}(x_u = -1) = 1$, and we abbreviate $m_{vu}^{(t)}(x_u = 1)$ as $m_{vu}^{(t)}$. With such normalization, our modified messages become

$$m_{vu}^{(t)} = \begin{cases} 0.5, & \text{if } p_v^{(t-1)} > 0.5 \text{ and } v \in \Gamma_i(u) \\ 0.5, & \text{if } p_v^{(t-1)} < 0.5 \text{ and } v \in \Gamma_o(u) \\ p_v^{(t-1)} w + (1-p_v^{(t-1)})(1-w), & \text{otherwise.} \end{cases} \tag{15}$$

With our modified messages, GANG does not need to store messages on edges and computing posterior beliefs is much more scalable as we will demonstrate in our experiments.

B. Approximating GANG with a Matrix Form

GANG iteratively applies Equations 15 and 12 with our modified messages, which still cannot guarantee convergence. The key reason is that Equation 12 combines messages from a node's neighbors in a nonlinear fashion. We make GANG converge via linearizing Equation 12. The resulting optimized GANG can be represented in a concise matrix form.

We define the residual of a variable y as $\hat{y} = y - 0.5$; we define the residual vector $\hat{\mathbf{y}}$ of $\mathbf{y}$ as $\hat{\mathbf{y}} = [y_1 - 0.5, y_2 - 0.5, \cdots]$; and we define the residual matrix $\hat{\mathbf{Y}}$ of $\mathbf{Y}$ as each entry of $\mathbf{Y}$ substracting 0.5. With residual variables, we can represent the residual message $\hat{m}_{vu}^{(t)}$ in Lemma 1.

Lemma 1 (Residual Messages)**.** *The residual message $\hat{m}_{vu}^{(t)}$ can be represented as follows:*

$$\hat{m}_{vu}^{(t)} = \begin{cases} 0 & \text{if } \hat{p}_v^{(t-1)} > 0 \text{ and } v \in \Gamma_i(u) \\ 0 & \text{if } \hat{p}_v^{(t-1)} < 0 \text{ and } v \in \Gamma_o(u) \\ 2\hat{p}_v^{(t-1)}\hat{w} & \text{otherwise.} \end{cases} \tag{16}$$

Proof: By substituting variables in Equation 15 with their residuals. ■

We denote by $\mathbf{A}_b \in \mathbb{R}^{|V|\times|V|}$, $\mathbf{A}_i \in \mathbb{R}^{|V|\times|V|}$, and $\mathbf{A}_o \in \mathbb{R}^{|V|\times|V|}$ the bidirectional, unidirectional incoming, and unidirectional outgoing adjacency matrices of the social graph, respectively. The uth row of $\mathbf{A}_b$, $\mathbf{A}_i$, and $\mathbf{A}_o$ represents the bidirectional, unidirectional incoming, and unidirectional outgoing neighbors of u. Formally, if there exists a bidirectional edge between u and v, then the entry $A_{b,uv} = A_{b,vu} = 1$,

otherwise, $A_{b,uv} = A_{b,vu} = 0$; if there exists an unidirectional edge from u to v, then $A_{o,uv} = 1$ and $A_{i,vu} = 1$. We define $\mathbf{p}^{(t)} = [p_1^{(t)}; p_2^{(t)}; \cdots; p_{|V|}^{(t)}]$ as the column vector of all nodes' posterior beliefs in the tth iteration, and $\hat{\mathbf{p}}^{(t)}$ as its residual vector. Similarly, we denote by $\mathbf{q} = [q_1; q_2; \cdots; q_{|V|}]$ the column vector of all nodes' prior beliefs, and by $\hat{\mathbf{q}}$ its residual vector. Let $\hat{\mathbf{P}}^{(t)} \in \mathbb{R}^{|V| \times |V|}$ be a matrix consisting of $|V|$ repeats of the column vector $\hat{\mathbf{p}}^{(t)}$, i.e., $\hat{\mathbf{P}}^{(t)} = [\hat{\mathbf{p}}^{(t)}, \hat{\mathbf{p}}^{(t)}, \cdots]$. With these notations, we have the following theorem, which states that GANG can be approximated as a concise matrix form.

Theorem 1. *We can approximate Equations 15 and 12 as the following equation:*

$$\begin{cases} \mathbf{A}_i'^{(t-1)} = I(\mathbf{A}_i \circ \hat{\mathbf{P}}^{(t-1)^T}), \\ \mathbf{A}_o'^{(t-1)} = I(-\mathbf{A}_o \circ \hat{\mathbf{P}}^{(t-1)^T}), \\ \hat{\mathbf{p}}^{(t)} = \hat{\mathbf{q}} + 2 \cdot \hat{w} \cdot (\mathbf{A}_b + \mathbf{A}_i'^{(t-1)} + \mathbf{A}_o'^{(t-1)}) \cdot \hat{\mathbf{p}}^{(t-1)}, \end{cases} \tag{17}$$

where the operator $\circ$ represents element-wise product of two matrices, $\mathbf{Y}^T$ is the transpose of the matrix $\mathbf{Y}$, and the indicator function $I(\mathbf{Y})$ means that an entry is set to be 0 if the corresponding entry of the matrix $\mathbf{Y}$ is non-negative, otherwise it is set to be 1.

Proof: See Appendix A. ■

Theorem 1 demonstrates that posterior beliefs can be iteratively solved by matrix operations without explicitly modeling messages. We note that Gatterbauer et al. [25] and Jia et al. [26] recently linearized LBP over a pMRF on an undirected graph. In particular, Jia et al. also leveraged the two steps of eliminating message maintenance and approximating as matrix form. However, our work linearizes LBP over a new pMRF on a directed graph, which is more complex.

Computational complexity: We use sparse matrix representation to implement GANG. In each iteration, GANG traverses each unidirectional edge twice (once for each node of the edge) and each bidirectional edge once. Therefore, time complexity of GANG is $O(2 \cdot (|E_1| + |E_2|) \cdot t)$, where t is the number of iterations. We note that the basic version of GANG has the same asymptotic time complexity. However, the constants in their asymptotic representations are different, which results in their significantly different scalability performances.

VI. Convergence Analysis

We analyze the conditions when our optimized version of GANG converges. Suppose we are given an iterative linear process: $\mathbf{y}^{(t)} \leftarrow \mathbf{c} + \mathbf{M}\mathbf{y}^{(t-1)}$. A basic result from linear systems [27] says that the linear process converges with any initial condition $\mathbf{y}^{(0)}$ if and only if the spectral radius of $\mathbf{M}$ is smaller than 1, i.e., $\rho(\mathbf{M}) < 1$. We use this basic result to analyze the convergence conditions for our optimized GANG.

Theorem 2 (Sufficient and Necessary Convergence Condition for Optimized GANG)**.** *A sufficient and necessary condition for our optimized GANG to converge is that the residual of the homophily strength (i.e., $\hat{w}$) is bounded as follows:*

$$\hat{w} < \frac{1}{2\rho\left(\mathbf{A}_b + \mathbf{A}_i'^{(t_*)} + \mathbf{A}_o'^{(t_*)}\right)}, \tag{18}$$

where $t_* = \text{argmax}_t \, \rho\left(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)}\right)$.

Proof: By applying the result in linear systems [27]. ■

Theorem 2 gives a strong sufficient and necessary convergence condition for optimized GANG. However, setting $\hat{w}$ using Theorem 2 is computationally expensive in practice. Because it involves simulating the iterative process and computing spectral radius. Therefore, we derive a *sufficient condition* for optimized GANG to converge. We can easily set $\hat{w}$ using our sufficient condition. Our sufficient condition is based on the fact that any norm is an upper bound of the spectral radius [28], i.e., $\rho(\mathbf{M}) \leq \|\mathbf{M}\|$, where $\|\cdot\|$ indicates some matrix norm. In particular, we use the induced l_1 matrix norm $\|\cdot\|_1$, i.e., $\|\mathbf{M}\|_1 = \max_j \sum_i |\mathbf{M}_{ij}|$, which is the maximum absolute column sum of the matrix. Our sufficient condition for convergence is as follows:

Theorem 3 (Sufficient Convergence Condition for Optimized GANG)**.** *Let $\|\cdot\|_1$ stand for the induced l_1 norm of a matrix. The following inequality is a sufficient condition for our optimized GANG to converge.*

$$\hat{w} < \frac{1}{2\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1} = \frac{1}{2\max_{u \in V} d_u}, \tag{19}$$

where $d_u = |\Gamma_u|$ is the degree of u.

Proof: See Appendix B. ■

Theorem 3 provides an elegant way to guide us to set $\hat{w}$, i.e., once $\hat{w}$ is smaller than the inverse of 2 times of the maximum node degree, GANG is guaranteed to converge. In practice, some nodes (e.g., celebrities) could have orders of magnitude bigger degrees than the others (e.g., ordinary people), and such nodes make $\hat{w}$ very small. In our experiments, we find that GANG can still converge when replacing the maximum node degree with the average node degree.

Similar to analyzing convergence of the optimized LBP over a standard pMRF on an undirected graph in Jia et al. [26], our analysis relies on linear systems and matrix theory. However, the mathematical details are more complex for the convergence analysis of the optimized version of GANG.

VII. Evaluations

A. Experimental Setup

Dataset description: We compare GANG with existing methods on two large-scale OSN datasets with labeled fraudulent and normal nodes.

First, we obtained a Twitter follower-followee graph with 41,652,230 users and 1,468,364,884 edges from Kwak et al. [29]. In this graph, a directed edge (u, v) means that u follows v. We obtained ground truth labels for each node from Wang et al. [7]. Specifically, 205,355 users were suspended by Twitter and we treated them as fraudulent users; 36,156,909 users were active and we treated them as normal users; and the remaining 5,289,966 users were deleted. As deleted users could be deleted by Twitter or by users themselves, we could not distinguish the two cases without accessing to Twitter's internal data. Thus, we treat them as unlabeled users. We sample 500,000 labeled users uniformly at random as a training set and treat the remaining labeled users as the testing set.

Second, we obtained a Sina Weibo dataset with 3,538,487 users and 652,889,971 directed edges from Fu et al. [9]. Like

TABLE II: Dataset statistics.

Dataset	Twitter	Sina Weibo
#Nodes	41,652,230	3,538,487
#Edges	1,468,364,884	652,889,971
Ave. degree	71	369

Twitter, a directed edge (u,v) means that u follows v. Fu et al. also manually labeled 2000 users sampled uniformly at random. Among them, 482 were fraudulent users, 1,498 were normal users, and 20 were unknown users (we do not consider these users in our experiments). We split the fraudulent and normal users into two halves; one is treated as the training set and the other is treated as the testing set. Table II shows some statistics of our datasets.

Compared methods: We compare GANG with both undirected and directed graph based methods. By default, we will use the optimized version of GANG.

1) Using undirected graphs. We consider the following undirected graph based methods: the well known graph-based semi-supervised learning method (SSL) [30], SybilRank [12], SybilBelief [5], and SybilSCAR [7]. SSL and SybilRank are based on random walks and SybilBelief is based on pMRF. SybilSCAR unifies random walk based methods and pMRF based methods as a local rule based framework. Moreover, under the framework, SybilSCAR designs a new local rule which outperforms existing random walk and pMRF based local rules. SSL, SybilBelief, and SybilSCAR can leverage both fraudulent users and normal users in the training dataset, while SybilRank is only able to leverage labeled normal users. These methods transform a directed graph into an undirected one via keeping an edge between two nodes if they are connected via bidirectional edges. This is more robust than keeping both bidirectional and unidirectional edges because fraudulent nodes can create arbitrary number of unidirectional edges with normal nodes, making them well embedded among normal nodes. Since these methods require connected graphs, we evaluate them on the largest connected component in the transformed undirected graph.

We note that SybilWalk [8] is very accurate at detecting fraudulent users in the Twitter dataset when we transform the directed graph into an undirected one via keeping an edge between two nodes if they are connected via directional edge(s). However, such transformation is not adversarially robust. On the transformed undirected Twitter graph with bidirectional edges only, SybilWalk achieves performance close to SybilBelief. Therefore, we do not show results of SybilWalk for simplicity.

2) Using directed graphs. We consider the following directed graph based methods: TrustRank [14], DistrustRank [15], CIA [16], and CatchSync [20]. TrustRank, DistrustRank, and CIA are based on random walks, while CatchSync leverages HITS [31]. TrustRank and DistrustRank were originally designed to detect fraudulent webpages based on hyperlinks, but they can be applied to detect fraudulent users in OSNs. TrustRank leverages only labeled normal nodes in the training dataset; DistrustRank and CIA are essentially the same, and they only leverage labeled fraudulent nodes; and CatchSync does not leverage the training dataset.

TABLE III: AUCs of compared methods.

Methods		Twitter	Sina Weibo
Using undirected graphs	SSL [30]	0.55	0.68
	SybilRank [12]	0.57	0.61
	SybilBelief [5]	0.61	0.65
	SybilSCAR [7]	0.64	0.68
Using directed graphs	TrustRank [14]	0.60	0.66
	DistrustRank [15]	0.63	0.64
	CIA [16]	0.63	0.64
	CatchSync [20]	0.68	0.51
	GANG	**0.72**	**0.80**

Parameter setting: For GANG, we set $\theta = 0.4$ to consider possible label noises, i.e., we assign a prior probability of being fraudulent 0.9, 0.1, and 0.5 to labeled fraudulent nodes, labeled normal nodes, and unlabeled nodes, respectively; we set $\delta = 10^{-3}$; and we respectively set $\hat{w} = 0.01$ and $\hat{w} = 0.001$ on Twitter and Sina Weibo, considering their different average node degrees. For all other compared methods, we set their parameters according to the original papers. For instance, we set the *decay factor* to be 0.85 for TrustRank [14] as suggested by its authors. For CatchSync [20], we set the parameter $\alpha = 3$ as suggested by its authors.

B. Ranking Results

Each compared method essentially computes a score for each node. We rank the nodes in the testing dataset using the scores such that fraudulent nodes are supposed to rank higher than normal nodes.

Overall ranking performance: We first use AUC to measure the overall ranking performance of the compared methods. In our problem, AUC can be interpreted as the probability that a randomly sampled fraudulent node is ranked higher than a randomly sampled normal node in the testing dataset. The higher AUC, the better performance. Table III shows the AUCs of all compared methods on the Twitter and Sina Weibo datasets. We observe that GANG consistently outperforms all compared methods on both datasets. We note that CatchSync achieves a close AUC as GANG on the Twitter dataset. However, CatchSync's performance degrades substantially on the Sina Weibo dataset. CatchSync relies on node degrees and properties of a node's neighbors. Therefore, we suspect the reason for CatchSync's poor performance on Sina Weibo is that nodes in the Sina Weibo dataset have larger average node degrees and their neighbors have more diverse properties.

Fraudulent nodes in top-ranked nodes: In practice, the ranking of nodes can be used as a priority list to help OSNs' human workers manually inspect nodes and detect fraudulent nodes. Inspecting nodes according to their rankings could aid human workers to detect more fraudulent nodes than inspecting nodes picked uniformly at random, within the same amount of time. When ranking is used for such purpose, the number of fraudulent nodes in top-ranked nodes is important because human workers can only inspect a limited number of nodes.

AUC measures the overall ranking performance, but it cannot tell fraudulent nodes among the top-ranked nodes. Therefore, we further compare the considered methods using

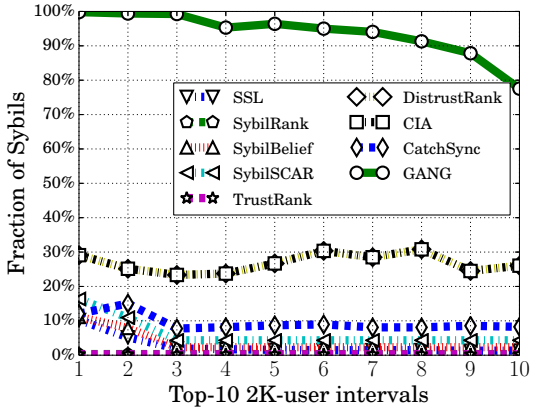
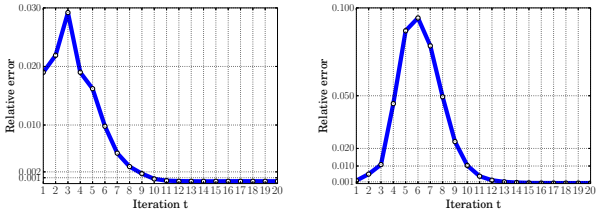


Fig. 2: Fraction of fraudulent nodes in each top ranked interval on the Twitter dataset.



(a) Twitter (b) Sina Weibo

Fig. 3: GANG's relative errors of residual posterior beliefs vs. number of iterations. GANG converges.

the fraction of fraudulent nodes in top-ranked nodes. In particular, we divide the top-20K nodes into 10 intervals, where each interval has 2K nodes. Figure 2 shows the fraction of fraudulent nodes in each interval for the Twitter dataset. Since the Sina Weibo dataset does not have enough labeled nodes to draw a similar graph, we omit its corresponding results. GANG achieves the best performance and substantially outperforms other methods. Specifically, the fraction of fraudulent nodes detected by GANG ranges from 77.5% to 99.8% in the top-10 2K-node intervals. The superiority of GANG comes from that GANG leverages unidirectional edges and GANG utilizes both labeled fraudulent and normal users.

Unbalanced vs. balanced training dataset: SSL, SybilBelief, SybilSCAR, and GANG leverage both labeled normal nodes and labeled fraudulent nodes in the training dataset. In real-world, human workers of an OSN would sample some nodes (e.g., 500,000 nodes in our Twitter dataset) and manually label them as a training dataset. In our Twitter dataset, such a training dataset is very unbalanced, i.e., fraudulent : normal = 1 : 176. We found that SSL, SybilBelief, SybilSCAR, and GANG achieve better results for fraudulent nodes in top-ranked nodes when using the unbalanced training dataset. However, they achieve much better AUCs when transforming the unbalanced training dataset into a balanced one. Specifically, among the 500,000 labeled nodes in the training dataset, we further sample some labeled normal nodes such that we

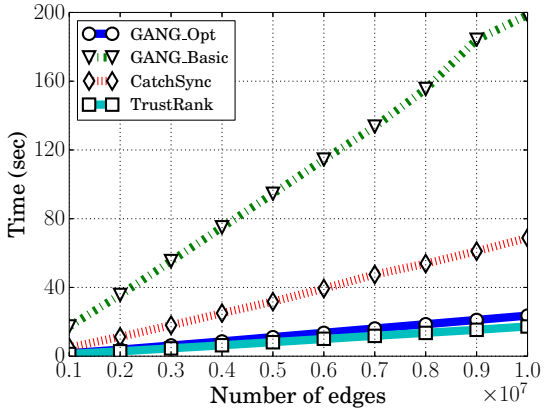


Fig. 4: Running time of directed graph based methods on synthesized graphs with increasing number of edges. DistrustRank and CIA have almost identical results with TrustRank, and thus we omit their results for conciseness.

have the same number of labeled fraudulent nodes and labeled normal nodes, which we treat as a balanced training dataset. Therefore, for the Twitter dataset, we obtained the AUC results in Table III for SSL, SybilBelief, SybilSCAR, and GANG using the sampled balanced training dataset; and we obtained the results in Figure 2 using the original unbalanced training dataset. For the Sina Weibo dataset, we did not observe the difference between unbalanced and balanced training dataset, and thus we use the original unbalanced training dataset for all methods. We suspect the reason is that Sina Weibo dataset has a much more balanced ratio between fraudulent nodes and normal nodes, i.e., fraudulent : normal = 1 : 3.

C. Convergence

Figure 3 shows GANG's relative errors of residual posterior beliefs in two consecutive iterations, i.e., $\|\hat{\mathbf{p}}^{(t)} - \hat{\mathbf{p}}^{(t-1)}\|_1 / \|\hat{\mathbf{p}}^{(t)}\|_1$, as a function of the number of iterations t. We observe that the relative error first increases, then decreases, and finally converges on both datasets.

D. Scalability

We measure the scalability of compared directed graph based methods with respect to the number of edges in the graph. Since we need graphs with different number of edges, but the Twitter and Sina Weibo datasets have fixed number of edges, we synthesize graphs according to a Preferential Attachment (PA) model [32]. We note that there are more advanced network models (e.g., the one proposed by Gong et al. [33]) to synthesize more realistic graphs. However, since the scalability does not depend on the graph structures, we use the simple PA model to synthesize graphs. All the compared methods involve iterative computing processes, e.g., TrustRank, DistrustRank, and CIA iteratively compute random walks, while CatchSync relies on the iterative HITS [31] algorithm. For fair comparison, we run the iterative processes with the same number of iterations. Figure 4 shows the running time used by the directed graph based methods (GANG_Basic

TABLE IV: Labeling results of the 1K nodes that are sampled from the top-ranked 100K nodes for Sina Weibo.

Category		Percentage	
Fraudulent users	Suspended users	41.5%	**92.0%**
	Spammers	42.5%	
	Compromised users	8.0%	
Normal users		**6.8%**	
Unknown users		**1.2%**	

and GANG_Opt are the basic and optimized versions of GANG, respectively) when we increase the number of edges in the synthesized graph.

First, GANG_Opt is slightly less efficient than random walk based methods TrustRank, DistrustRank, and CIA. This is because, in each iteration, these methods traverse each unidirectional edge once while GANG traverses twice. Second, GANG_Opt is more scalable than CatchSync. This is because CatchSync first uses HITS, which already has the same time complexity with GANG_Opt, to compute nodes' hubness and authoritativeness scores, and then CatchSync further computes each node's *synchronicity*, which involves going through node pairs between a node's outgoing neighbors, and *normality*, which involves going through node pairs between a node's outgoing neighbors and all nodes. Third, GANG_Opt is one order of magnitude more scalable than GANG_Basic.

E. Case Study on Sina Weibo

We apply our GANG to the Sina Weibo dataset and manually inspect the detected fraudulent nodes. Specifically, we use all the 1980 labeled nodes as a training dataset and produce a ranking list for the remaining nodes. Then we sample 1K nodes from the top-ranked 100K nodes uniformly at random, and we manually inspect them. Table IV shows the labeling results of the 1K nodes.

1) Suspended users. These users didn't exist any more at the time of our inspection. They could be suspended/deleted by Sina Weibo's detector or the users themselves.

2) Spammers. These users post or share a large amount of advertisements, e.g., to promote their products or to sell pirated products. These users violate Sina Weibo's policy,[1] and thus they should be suspended/deleted by Sina Weibo company. Interestingly, we found that some spammers also posted many *seemingly normal tweets* to camouflage themselves. For instance, Figure 5(a) shows an example spammer who posts seemingly normal tweets exactly every 9 hours and 13 minutes. We randomly sampled some tweets and used them as keywords to search on Baidu (the largest search engine in China). We found that these tweets were simply copied from Internet. Figure 5(b) shows the search results of one tweet. We suspect such spammers are controlled by software and are trying to evade content-based detection. Indeed, they have successfully evaded Sina Weibo's detector.

3) Compromised users. These users posted normal tweets about daily activities before a certain time point, and then they started to post or share a large amount of advertisements. We also randomly sampled some normal tweets of these users, but we could not find them on the Baidu search engine. Therefore, we suspect that these users could be compromised normal users. Our method can detect these compromised users because they link to other spammers to share their tweets.

[1] http://weibo.cn/dpool/ttt/h5/regagreement.php

(a) Periodic tweeting (b) Search results

Fig. 5: (a) A user performing periodic tweeting. (b) Search results of one of the user's tweet on Baidu.

4) Normal users. These users post normal tweets and comply with Sina Weibo's policy.

5) Unknown users. These users had no tweets at the time of inspection, so we cannot classify them through contents.

Comparing with Sina Weibo's detector: When Sina Weibo's detector detects a fraudulent user, the user will be suspended or deleted. In other words, the number of fraudulent nodes detected by Sina Weibo's detector is upper bounded by the category *suspended users*, and the users in the category *spammers* and *compromised users* have evaded Sina Weibo's detector. However, our method GANG can detect these fraudulent users.

VIII. Conclusion and Future Work

In this work, we propose GANG, a guilt-by-association method on directed graphs, to detect fraudulent users in OSNs. Based on the unique characteristics of the fraudulent-user-detection problem in directed graphs, we design a novel pairwise Markov Random Field to model the joint probability distribution of the states of all users. In the basic version of GANG, we use Loopy Belief Propagation to perform inference. Furthermore, we optimize GANG to make it convergent and more scalable via eliminating message maintenance and approximating GANG by a concise matrix form. We compare GANG with various existing guilt-by-association methods using a large-scale Twitter dataset and a large-scale Weibo dataset with labeled fraudulent users and normal users. Our results demonstrate that GANG substantially outperforms existing guilt-by-association methods. Moreover, we demonstrate that the optimized version of GANG is significantly more efficient than its basic version.

Future research direction includes applying GANG to detect other online frauds such as web spams, spamming reviews, and fake page likes.

References

[1] Fake Users in Twitter, "http://goo.gl/q1snms."

[2] Hacking Election. (2016, May). [Online]. Available: http://goo.gl/G8o9x0

[3] Hacking Financial Market. (2016, May). [Online]. Available: http://goo.gl/4AkWyt

[4] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *IEEE S & P*, 2011.

[5] N. Z. Gong, M. Frank, and P. Mittal, "SybilBelief: A semi-supervised learning approach for structure-based sybil detection," *IEEE TIFS*, vol. 9, no. 6, 2014.

[6] P. Gao, N. Z. Gong, S. Kulkarni, K. Thomas, and P. Mittal, "Sybilframe: A defense-in-depth framework for structure-based sybil detection," *CoRR*, 2015.

[7] B. Wang, L. Zhang, and N. Z. Gong, "SybilSCAR: Sybil detection in online social networks via local rule based propagation," in *IEEE INFOCOM*, 2017.

[8] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *IEEE DSN*, 2017.

[9] H. Fu, X. Xie, Y. Rui, N. Z. Gong, G. Sun, and E. Chen, "Robust spammer detection in microblogs: Leveraging user carefulness," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2017.

[10] 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*, 2007.

[11] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil nodes using social networks," in *NDSS*, 2009.

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

[13] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *KDD*, 2015.

[14] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB*, 2004.

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

[16] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammer's social networks for fun and profit," in *WWW*, 2012.

[17] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network Sybils in the wild," in *IMC*, 2011.

[18] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *SIGIR*, 2010.

[19] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *CCS*, 2014.

[20] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Catchsync: catching synchronized behavior in large directed graphs," in *KDD*, 2014.

[21] X. Hu, J. Tang, H. Gao, and H. Liu, "Social spammer detection with sentiment information," in *ICDM*, 2014.

[22] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, 1988.

[23] N. Z. Gong and W. Xu, "Reciprocal versus parasocial relationships in online social networks," *Social Network Analysis and Mining*, vol. 4, no. 1, pp. 1–14, 2014.

[24] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *KDD*, 2016.

[25] W. Gatterbauer, S. Günnemann, D. Koutra, and C. Faloutsos, "Linearized and single-pass belief propagation," *PVLDB*, vol. 8, no. 5, 2015.

[26] J. Jia, B. Wang, L. Zhang, and N. Z. Gong, "AttriInfer: Inferring user attributes in online social networks using markov random fields," in *WWW*, 2017.

[27] Y. Saad, *Iterative methods for sparse linear systems*. Siam, 2003.

[28] N. Derzko and A. Pfeffer, "Bounds for the spectral radius of a matrix," *Mathematics of Computation*, vol. 19, no. 89, 1965.

[29] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *WWW*, 2010.

[30] X. Zhu, Z. Ghahramani, J. Lafferty *et al.*, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003.

[31] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, 1999.

[32] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, 1999.

[33] N. Z. Gong, W. Xu, L. Huang, P. Mittal, E. Stefanov, V. Sekar, and D. Song, "Evolution of social-attribute networks: Measurements, modeling, and implications using google+," in *IMC*, 2012.

APPENDIX A
PROOF OF THEOREM 1

Our idea is to first linearize Equation 12 and then combine the linearized version with Equation 16. Optimizing LBP for a standard pMRF on undirected graphs in Jia et al. [26] also involves linearizing Equation 12. In particular, they linearized Equation 12 as follows:

$$\hat{p}_u^{(t)} = \hat{q}_u + \sum_{v \in \Gamma(u)} \hat{m}_{vu}^{(t)}. \tag{20}$$

Next, we combine Equation 20 with Equation 16 to represent posterior beliefs as matrix form without explicitly modeling messages. Recall that there are three types of neighbors for each node and we can expand Equation 20 as follows:

$$\hat{p}_u^{(t)} = \hat{q}_u + \sum_{v \in \Gamma_b(u)} \hat{m}_{vu}^{(t)} + \sum_{v \in \Gamma_i(u)} \hat{m}_{vu}^{(t)} + \sum_{v \in \Gamma_o(u)} \hat{m}_{vu}^{(t)}. \tag{21}$$

Therefore, we expect to solve $\hat{m}_{vu}^{(t)}$ for u over its bidirectional neighbors, unidirectional incoming neighbors, and unidirectional outgoing neighbors. Towards this goal, we leverage the bidirectional adjacency matrix $\mathbf{A}_b$, the unidirectional incoming adjacency matrix $\mathbf{A}_i$, and unidirectional outgoing adjacency matrix $\mathbf{A}_o$. For a bidirectional neighbor v, $\hat{m}_{vu}^{(t)}$ can be rewritten as $\hat{m}_{vu}^{(t)} = 2\hat{p}_v^{(t-1)}\hat{w}$ according to Lemma 1. For an unidirectional incoming neighbor, $\hat{m}_{vu}^{(t)} = 0$ when $\hat{p}_v^{(t-1)} > 0$. For an unidirectional outgoing neighbor, $\hat{m}_{vu}^{(t)} = 0$ when $\hat{p}_v^{(t-1)} < 0$. Therefore, we define an indicator function $I(\mathbf{Y})$ over a matrix $\mathbf{Y}$, where an entry is set to be 0 if the corresponding entry of the matrix $\mathbf{Y}$ is non-negative, otherwise it is set to be 1. With these notations, we can obtain Equation 17 via combining Equation 21 with Equation 16.

APPENDIX B
PROOF OF THEOREM 3

$\mathbf{A}_i'^{(t)}$ and $\mathbf{A}_i$ are binary matrices. For any iteration t, we have $\mathbf{A}_i'^{(t)} = I(\mathbf{A}_i \circ \hat{\mathbf{P}}^{(t)^T}) \leq \mathbf{A}_i \circ I(\hat{\mathbf{P}}^{(t)^T}) \leq \mathbf{A}_i$, where the comparison between two matrices is element-wise. Similarly, we have $\mathbf{A}_o'^{(t)} \leq \mathbf{A}_o$. Therefore, we have $\|\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)}\|_1 \leq \|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1$ for any iteration t.

As $\rho(\mathbf{M}) \leq \|\mathbf{M}\|_1$, we achieve a sufficient condition via enforcing $\|\mathbf{M}\|_1 < 1$, where $\mathbf{M} = 2\hat{w}(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)})$ in optimized GANG. Specifically, we have the following derivations:

$$\rho(2\hat{w}(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)})) < 1 \tag{22}$$

$$\Longleftarrow \|2\hat{w}(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)})\|_1 < 1 \tag{23}$$

$$\Longleftarrow 2\hat{w}\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1 < 1 \tag{24}$$

$$\Longleftarrow \hat{w} < \frac{1}{2\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1} \tag{25}$$

Finally, $\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1 = \max_{u \in V} d_u$.