2019

Graph-based security and privacy analytics via collective classification

Binghui Wang
Iowa State University

Graph-based security and privacy analytics via collective classification

by

Binghui Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Iowa State University

Ames, Iowa

2019

DEDICATION

I would like to dedicate this dissertation to my wife and my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Neil Zhenqiang Gong, for his continuous support for my Ph.D. study and research. His guidance and encourgement helped me all the time when I was stuck and depressed. His research enthusiasm and research style have influenced me profoundly on my research career.

In addition, I would like to thank my committee members, including Prof. Daji Qiao, Prof. Yong Guan, Prof. Doug Jacobson, and Prof. Jin Tian, for their insightful comments and suggestions that have broadened my research horizon from different perspectives. A specical thanks goes to Prof. Daji Qiao. Without his help, I would have no chance to get to know my supervisor.

I also want to thank my collaborators. During my Ph.D. study, I have collaborated with Yinzhi Cao, Yiu-ming Cheung, Hao Fu, Peng Gao, Neil Gong, Jinyuan Jia, Sanjeev Kulkarni, Song Li, Chuang Lin, Risheng Liu, Prateek Mittal, Luke Myers, Meng Pang, Daji Qiao, Kurt Thomas, Le Zhang, and Tianchen Zhou. I have benefited a lot from my collaborators.

I would additionally like to thank my friends: Xiaoyu Cao, Xiaoyi Dou, Yesta Izenov, Jinyuan Jia, Yueyi Jiao, Siyao Li, Zhonghao Liao, Zhaoyu Liu, Liqun Ma, Shou Tang, Pengyuan Wang, Ling Xiang, Songzhe Xu, Bo Yang, Shuo Yang, Anji Yu, Le Zhang, and Zhikun Zhong. I have spent a wonderful time with them at Iowa State University.

Last but not least, I am so grateful to my wife Yun Dong and my parents for their endless love and support.

ABSTRACT

Graphs are a powerful tool to represent complex interactions between various entities. A particular family of graph-based machine learning techniques called *collective classification* have been applied to various security and privacy problems, e.g., malware detection, Sybil detection in social networks, fake review detection, malicious website detection, auction fraud detection, APT infection detection, attribute inference attacks, etc.. Moreover, some collective classification methods have been deployed in industry, e.g., Symantec deployed collective classification to detect malware; Tuenti, the largest social network in Spain, deployed collective classification to detect Sybils.

In this dissertation, we aim to systematically study graph-based security and privacy problems that are modeled via collective classification. In particular, we focus on collective classification methods that leverage random walk (RW) or loopy belief propagation (LBP).

First, we propose a local rule-based framework to unify existing RW-based and LBP-based methods. Under our framework, existing methods can be viewed as iteratively applying a different local rule to every node in the graph. know about the node.

Second, we design a novel local rule for undirected graphs. Based on our local rule, we propose a collective classification method that can maintain the advantages and overcome the disadvantages of state-of-the-art undirected graph-based collective classification methods for Sybil detection.

Third, many security and privacy problems are modeled using directed graphs. Directed graph-based security and privacy problems have their unique characteristics. Existing undirected graph-based collective classification methods (e.g., LBP-based methods) cannot be applied to directed graphs and existing directed graph-based methods (e.g., RW-based methods) cannot make full use of the labeled training set. To address the issue, we develop a novel local rule for directed graph-based Sybil detection and propose a collective classification method that captures unique characteristics of directed graph-based Sybil detection.

Finally, one key issue of *all* collective classification methods is that they either assign small weights to a large number of edges whose two corresponding nodes have the same label or/and assign large weights to a large number of edges whose two corresponding nodes have different labels. Although collective classification has been studied and applied for security and privacy problems for more than a decade, it is still challenging to assign edge weights such that an edge has a large weight if the two corresponding nodes have the same label, and a small weight otherwise. We develop a novel collective classification framework to address this long-standing challenge. Specifically, we first formulate learning edge weights as an optimization problem, which, however, is computationally challenging to solve. Then, we relax the optimization problem and design an efficient joint weight learning and propagation algorithm to solve this approximate optimization problem.

CHAPTER 1. OVERVIEW

1.1 Collective Classification

Graphs are a powerful tool to represent complex interactions between various entities. A particular family of graph-based machine learning techniques called *collective classification* Sen et al. (2008) have been studied from multiple communities such as security, data mining, and networking. Figure 1.1 illustrates the setting of collective classification for binary classes: given 1) a graph, which can be either undirected or directed, and 2) a training dataset, which consists of some labeled positive nodes (+) and/or labeled negative nodes (-), collective classification is to classify the remaining unlabeled nodes (?) to be positive or negative *simultaneously*.

State-of-the-art collective classification methods Gyöngyi et al. (2004); Wu et al. (2006); Li et al. (2013); Pandit et al. (2007); Chau et al. (2011); Mohaisen et al. (2011); Cao et al. (2012); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015); Stringhini et al. (2017); Wang et al. (2017b); Jia et al. (2017a); Wang et al. (2017a); Boshmaf et al. (2015); Wang et al. (2018a); Gao et al. (2018) have three key steps. In Step I, they assign a *prior reputation score* to every node in the graph based on the training dataset. In Step II, they assign weights to edges in the graph, where a large weight of an edge (u, v) indicates that u and v are likely to have the same label. In Step III, they iteratively propagate the prior reputation scores among the weighted graph to obtain a *posterior reputation score* for every node. The posterior reputation scores are used to classify or rank nodes. The details of one or more of the three steps could be different for different collective classification methods. For instance, some methods leverage random walks for propagation in Step III, while some methods leverage loopy belief propagation in Step III, which we call *random walk (RW)-based methods* and *loopy belief propagation (LBP)-based methods*, respectively.

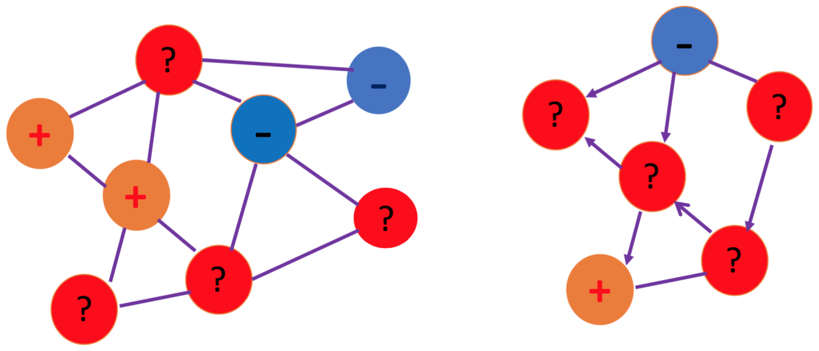


Figure 1.1 Illustration of collective classification.

1.1.1 RW-based methods

Depending on which labeled nodes in the training dataset are used to assign the prior reputation scores in Step I, we classify RW-based methods into three categories, i.e., RW that only leverages labeled negative nodes (RW-N), RW that only leverages positive nodes (RW-P), and RW that leverages both labeled positive nodes and labeled negative nodes (RW-B). RW-N Wu et al. (2006); Gyöngyi et al. (2004); Cao et al. (2012); Li et al. (2013); Boshmaf et al. (2015); Gong and Liu (2016) leverages labeled negative nodes to assign prior reputation scores, e.g., every labeled negative node in the training dataset has a prior reputation score of -1 (the absolute value of the reputation score does not matter since these RW-based methods rely on relative reputation scores) and the remaining nodes have prior reputation scores of 0. RW-P Wu et al. (2006); Yang et al. (2012) leverages labeled positive nodes to assign prior reputation scores, e.g., every labeled positive node has a prior reputation score of 1 and the remaining nodes have prior reputation scores of 0. RW-B Zhu et al. (2003); Ye et al. (2011); Jia et al. (2017a) leverages both labeled positive nodes and labeled negative nodes to assign prior reputation scores, e.g., labeled positive nodes, labeled negative nodes, and unlabeled nodes have prior reputation scores of 1, -1, and 0, respectively.

In Step II, most RW-based methods simply set a constant weight to all edges. Íntegro Boshmaf et al. (2015), developed under the context of Sybil detection in social networks, learns edge weights using features extracted from nodes. Specifically, Íntegro first trains a node classifier based on nodes' features to predict the probability that a node is a victim (a victim is a negative node connecting to at least one positive node). Then, Íntegro assigns small weights to all edges of the nodes that are predicted to be victims. For notation convenience, we denote Íntegro as *RW-FLW (Fixed Learnt Weights)* since it learns the edge weights and fixes them in Step II.

In Step III, different methods use different random walks to propagate the reputation scores. For instance, on an undirected graph, RW-N, RW-P, and RW-FLW iteratively distribute a node's current reputation score to its neighbors and a node sums the reputation scores distributed from its neighbors as its new reputation score. In particular, a node u shares a fraction of its reputation score to a neighbor v, where the fraction is the edge weight divided by the weighted degree of u. On a directed graph, RW-N Gyöngyi et al. (2004) (or RW-P Wu et al. (2006); Yang et al. (2011)) shares a fraction of node u's current reputation score to each outgoing (or incoming) neighbor, where the fraction is the edge weight divided by the total weights of u's outgoing (or incoming) edges. RW-B is applicable to undirected graphs and uses a different random walk to propagate reputation scores. Specifically, the fraction is the edge weight divided by the weighted degree of v instead of u when u shares its reputation score to a neighbor v.

1.1.2 LBP-based methods

These methods Pandit et al. (2007); Chau et al. (2011); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015); Wang et al. (2017b,a); Gao et al. (2018); Wang et al. (2018a) leverage both labeled positive nodes and labeled negative nodes in the training dataset to assign prior reputation scores in Step I, e.g., labeled positive nodes, labeled negative nodes, and unlabeled nodes have prior reputation scores of 1, -1, and 0, respectively. In Step II, most LBP-based methods simply set a constant weight to all edges in the graph. SybilFuse Gao et al. (2018), developed under the context of Sybil detection, learns edge weights using features extracted from the graph structure. Specifically, SybilFuse first trains a node classifier using structure-based node features (e.g., local clustering coefficient) to predict the prior reputation score for each node. Then, SybilFuse uses the prior reputation scores to assign edge weights in a way that a larger edge weight is assigned if the two corresponding nodes are more likely to have the same label based on the prior reputation scores. We note that SybilFuse can also extract structure-based edge features (e.g., common neighbors between two nodes) to train an edge classifier to assign the edge weights. To stress the fact that SybilFuse learns edge weights and fixes them in Step II, we denote it as *LBP-FLW*.

In Step III, these LBP-based methods Pandit et al. (2007); Chau et al. (2011); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015) leverage either a standard LBP Pearl (1988) or an optimized LBP Gatterbauer et al. (2015); Wang et al. (2017b,a); Gatterbauer (2017); Wang et al. (2018a) to propagate the reputation scores among the weighted graph. The optimized LBP is an order of magnitude more efficient than the standard LBP. On an undirected graph, the optimized LBP shares a fraction of a node u's reputation score to its neighbor v, where the fraction is simply the edge weight, and a node sums its prior reputation score and the reputation scores shared from its neighbors as its new reputation score in each iteration. On a directed graph, the optimized LBP shares a node u's reputation score to its neighbor v like on an undirected graph if both the directed edges (u, v) and (v, u) exist. However, when u connects v with only an outgoing edge, the optimized LBP shares u's reputation score with v like on an undirected graph only if u's current reputation score is negative. When u connects v with only an incoming edge, the optimized LBP shares u's reputation score with v only if u's current reputation score is positive.

1.2 Modeling Security/Privacy Problems As Collective Classification

Studies from multiple communities–such as security, data mining, and networking–have shown that various security and privacy problems can be modeled as collective classification on graphs. Such security and privac problems include malware detection Chau et al. (2011); Ye et al. (2011); Rajab et al. (2013); Tamersoy et al. (2014); Stringhini et al. (2017), Sybil detection in social networks Yu et al. (2006, 2008); Danezis and Mittal (2009); Viswanath et al. (2010); Mohaisen et al. (2011); Yang et al. (2011); Cao et al. (2012); Boshmaf et al. (2015); Gong et al. (2014a); Fu

Table 1.1 Representative methods developed for different security and privacy applications. *Graph Type*: "Undir" and "Dir" respectively indicate a collective classification method uses an undirected graph or a directed graph. *Training Dataset*: "Neg", "Pos", or "Both" means that a method leverages labeled negative nodes, labeled positive nodes, or both to assign prior reputation scores. *Weight Learning*: "Y" or "N" means that a method learns edge weights or does not. We note that all these methods assign edge weights and fix them in Step II.

Method		Application	Graph Type	Training Dataset	Weight Learning
RW	**SybilGuard Yu et al. (2006)**	Sybil Detection	Undir	Neg	N
	SybilLimit Yu et al. (2008)	Sybil Detection	Undir	Neg	N
	SybilInfer Danezis and Mittal (2009)	Sybil Detection	Undir	Neg	N
	SybilRank Cao et al. (2012)	Sybil Detection	Undir	Neg	N
	CIA Yang et al. (2012)	Sybil Detection	Dir	Pos	N
	Íntegro Boshmaf et al. (2015)	Sybil Detection	Undir	Neg	Y
	SmartWalk Liu et al. (2016)	Sybil Detection	Undir	Neg	N
	SybilWalk Jia et al. (2017a)	Sybil Detection	Undir	Both	N
	ELSIEDET Zheng et al. (2018)	Sybil Detection	Undir	Pos	N
	TrustRank Gyöngyi et al. (2004)	Mal. Web. Detection	Dir	Neg	N
	DistrustRank Wu et al. (2006)	Mal. Web. Detection	Dir	Pos	N
	Li et al. Li et al. (2013)	Mal. Web. Detection	Dir	Neg	N
	Ye et al. Ye et al. (2011)	Malware Detection	Undir	Both	N
	Marmite Stringhini et al. (2017)	Malware Detection	Undir	Both	N
	RWwR-SAN Gong et al. (2014b)	Attribute Inference	Undir	Pos	N
	VAIL Gong and Liu (2016)	Attribute Inference	Undir	Pos	N
	EdgeExplain Chakrabarti et al. (2017)	Attribute Inference	Undir	Both	N
LBP	**SybilBelief Gong et al. (2014a)**	Sybil Detection	Undir	Both	N
	SybilSCAR Wang et al. (2017b, 2018a)	Sybil Detection	Undir	Both	N
	GANG Wang et al. (2017a)	Sybil Detection	Dir	Both	N
	SybilFuse Gao et al. (2018)	Sybil Detection	Undir	Both	Y
	Burst Fei et al. (2013)	Fake Review Detection	Undir	Both	N
	FraudEagle Akoglu et al. (2013)	Fake Review Detection	Undir	Both	N
	SpEagle Rayana and Akoglu (2015)	Fake Review Detection	Undir	Both	N
	Polonium Chau et al. (2011)	Malware Detection	Undir	Both	N
	AESOP Tamersoy et al. (2014)	Malware Detection	Undir	Both	N
	NetProbe Pandit et al. (2007)	Auction Fraud Detection	Undir	Both	N
	Olteanu et al. Olteanu et al. (2017)	Attribute Inference	Undir	Both	N
	AttriInfer Jia et al. (2017b)	Attribute Inference	Undir	Both	N

Table 1.2 Possible meanings of nodes, edges, positive, and negative in example security and privacy application scenarios.

Problem	Nodes	Edges	Positive	Negative
Sybil Detection	Users	Friendship, Following	Sybil	Benign User
Fake Review Detection	Users, Reviews, Products	Reviewing	Fake Review	Genuine Review
Malware Detection	Files, Hosts	Appearance	Malware	Benign File
Malicious Website Detection	Websites	Redirection	Malicious Website	Benign Website
Attribute Inference Attack	Users	Friendship	Having the Attribute	Not Having the Attribute

et al. (2017); Wang et al. (2017b, 2018a); Jia et al. (2017a); Wang et al. (2017a); Zheng et al. (2018); Gao et al. (2018), fake review detection Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015), malicious website detection Gyöngyi et al. (2004); Wu et al. (2006); Li et al. (2013), auction fraud detection Pandit et al. (2007), APT infection detection Oprea et al. (2015), and attribute inference attacks Zheleva and Getoor (2009); Gong et al. (2014b); Chaabane et al. (2012); Gong and Liu (2016); Jia et al. (2017b); Olteanu et al. (2017); Gong and Liu (2018). Moreover, some collective classification methods have been deployed in industry, e.g., Symantec deployed collective classification to detect malware Tamersoy et al. (2014) and Tuenti, the largest social network in Spain, deployed collective classification to detect Sybils Cao et al. (2012); Boshmaf et al. (2015).

For different graph-based security and privacy applications, nodes represent different entities, edges represent different relationships, and the labels "positive" and "negative" have different semantic meanings. Table 1.1 list representative methods developed for different security and privacy applications. Table 1.2 summarizes several security and privacy problems that can be modeled as collective classification on graphs. We note that some security problems Kwon et al. (2015); Feng et al. (2016); Xu et al. (2017) can be modeled as graph analytics other than collective classification. However, we focus on collective classification based security/privacy analytics.

- **Sybil detection in social networks.** Social networks (e.g., Facebook, Twitter, Sina Weibo) are fundamentally vulnerable to an attack called *Sybil attack*, in which an attacker manages a large number of malicious users (also called Sybils). An attacker can use these Sybils perform various malicious activities such as disrupting democratic election Hacking Election.

(2016), influencing financial market Hacking Financial Market. (2016), distributing spams and phishing attacks Thomas et al. (2011), as well as harvesting private user data Bilge et al. (2009). Therefore, detecting Sybils in social networks is a fundamental security research problem. Various graph-based collective classification methods Gong et al. (2014a); Gao et al. (2015); Wang et al. (2017b); Jia et al. (2017a); Fu et al. (2017); Pandit et al. (2007); Danezis and Mittal (2009); Cao et al. (2012); Rayana and Akoglu (2015); Gyöngyi et al. (2004); Wu et al. (2006); Yang et al. (2012); Boshmaf et al. (2015) have been proposed to detect Sybils on social graphs. An example graph-based method called SybilRank Cao et al. (2012) and its enhanced version called Integro Boshmaf et al. (2015) have been deployed by Tuenti, the largest social network in Spain, to detect a large amount of Sybils. In a social graph, nodes are users and edges represent friendships or following relationships between users; a positive node means a Sybil and a negative node means a benign user.

- **Malware detection.** Imagine many end hosts have installed a certain antivirus software from a certain security company (e.g., Symantec). The security company has access to the executable files on all these hosts. Therefore, the security company can construct a host-file graph to represent the relationships between hosts and executable files. Specifically, in this graph, a node is a host or an executable file; an edge between a host and an executable file means that the file is on the host.

  In malware detection, a host-file graph is constructed to represent the relationships between hosts and executable files Chau et al. (2011); Ye et al. (2011); Rajab et al. (2013); Tamersoy et al. (2014); Stringhini et al. (2017). Specifically, in this graph, a node is a host or an executable file; an edge between a host and an executable file means that the file appears on the host; a positive node means a malware or compromised host, while a negative node means a benign file or normal host.

- **Fake review detection.** Online reviews capture the testimonials of "real" people and help shape the decisions of other consumers. Financial incentives associated with reviews, however, have created a market of (often paid) users to fabricate *fake reviews* to either unjustly hype (for promotion) or defame (under competition) a product or business. Therefore, detecting fake reviews in social graphs is important. In fake review detection, a user–review–product graph is constructed to represent the relationships between users, reviews, and products Rayana and Akoglu (2015). In this graph, a node is a user, a review, or a product; an edge between a user and a review means that the user writes the review; an edge between a review and a product means that the review is for the product; a positive node means a fake review while a negative node means a genuine review.

- **Attribute inference attacks in social networks.** Attribute inference has serious implications for Internet privacy as well as applications for targeted advertisements and personalized recommendation. Therefore, various parties (e.g., cyber criminal, online social network provider, advertiser, data broker, and surveillance agency) are motivated to perform attribute inference. In the attribute inference attack, an attacker aims to infer users' private attributes (e.g., locations, sexual orientation, and interests) using their public data (e.g., page likes on Facebook and friend list) in social networks. The graph is the social graph between users; a node is positive if the node has a given attribute (e.g., republican), otherwise the node is negative.

1.3 Pros and Cons of Existing Collective Classification Methods

RW-based methods have the following pros: 1) they are scalable; 2) they can guarantee to converge; 3) they have theoretically guaranteed performance; and 4) they are applicable to both undirected and directed graphs. However, they suffer from several key limitations: 1) they are not robust to label noise in the training dataset. The label of a user is noisy if the label is incorrect. Label noise often exists in practice due to human mistakes when manually labeling users Thomas et al. (2011); 2) they have limited performance in graphs with weak homophily. Strong homophily means that two connected nodes are likely to share the same label with high probability. LBP-based methods can 1) leverage both labeled positive nodes and labeled negative nodes; 2) be robust to label noise; and 3) tolerate graphs with a relative weak homophily. However, 1) they cannot guarantee convergence on real-world graphs; 2) they are not scalable; 3) they do not have theoretically guaranteed performance; and 4) they cannot be directly applied to directed graphs. Furthermore, a common limitation of both RW-based methods and LBP-based methods is that they assign edge weights inappropriately.

1.4 Organization of The Dissertation

The focus of this dissertation is to study security and privacy problems that are modeled via collective classification, and address the limitations of existing RW-based and LBP-based collective classification methods.

In Chapter 2, we develop a local rule-based framework to unify existing RW-based and LBP-based methods. Under our framework, different collective classification methods reduce to designing different local rules.

In Chapter 3, we design a novel local rule for *undirected graph*-based Sybil detection and propose a method that integrates advantages of existing RW-based and LBP-based Sybil detection methods, while overcoming their limitations. For instance, our method can guarantee to converge and has theoretically guaranteed performance.

In Chapter 4, we design a novel local rule for *directed graph*-based Sybil detection. In contrast to undirected graphs, directed graph-based Sybil detection has its unique characteristics. We propose a customized pairwise Markov Random Field model to capture these unique characteristics, and infer the model via LBP. Furthermore, we optimize our method to make it scalable and convergent.

In Chapter 5, we design an edge weight learning framework for graph-based collective classification methods. Our framework formulates learning edge weights as an optimization problem and learns edge weights such that an edge has a large weight if the two corresponding nodes have the same label, and a small weight otherwise.

Finally, we conclude the dissertation and discuss several future directions in Chapter 6.

CHAPTER 2. COLLECTIVE CLASSIFICATION UNDER A LOCAL RULE-BASED FRAMEWORK

2.1 Our Local Rule-based Framework

State-of-the-art collective classification methods Gyöngyi et al. (2004); Wu et al. (2006); Li et al. (2013); Pandit et al. (2007); Chau et al. (2011); Mohaisen et al. (2011); Cao et al. (2012); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015); Stringhini et al. (2017); Wang et al. (2017b); Jia et al. (2017a); Wang et al. (2017a); Boshmaf et al. (2015); Wang et al. (2018a); Gao et al. (2018) in social graphs can be grouped into Random Walk (RW)-based methods and Loop Belief Propagation (LBP)-based methods. Given a training dataset, these methods iteratively propagate label information among the social graph to predict labels for users. RW-based methods implement the propagation using random walks, while LBP-based methods implement the propagation using Loopy Belief Propagation Pearl (1988).

We can unify existing RW-based methods Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012); Boshmaf et al. (2015); Liu et al. (2016) and LBP-based methods Gong et al. (2014a); Gao et al. (2018) into a local rule-based framework, although RW-based methods and LBP-based methods use very different mathematical foundations (i.e., RW vs. LBP). Specifically, these methods first assign the *prior knowledge* of all nodes using a training dataset. Then, they propagate the prior knowledge among the social network to obtain the *posterior knowledge* via iteratively applying their *local rules* to every node. *A local rule is to update the posterior knowledge of a node by combining the influences from its neighbors with its prior knowledge.* We call the influence from a neighbor *neighbor influence*. Figure 2.1 shows our unified framework. Under our framework, designing a new collective classification method is reduced to designing a new local rule.

Notations: We denote by w_{uv} the weight of the edge (u, v), Γ_u the set of neighbors of node u, and d_u the total weights of edges linked to u, i.e., $d_u = \sum_{v \in \Gamma_u} w_{uv}$. In RW-based methods Mohaisen

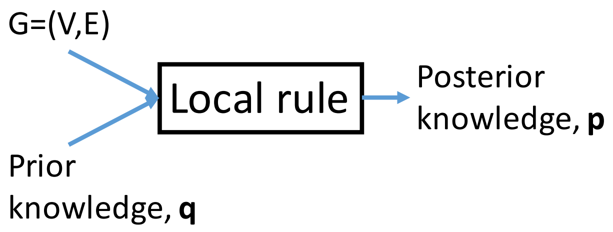


Figure 2.1 Our proposed framework to unify state-of-the-art RW-based and LBP-based Sybil detection methods.

et al. (2011); Yang et al. (2012); Boshmaf et al. (2015), edge weights model the relative importance (e.g., level of trust) of edges. In LBP-based method Gong et al. (2014a), an edge weight w_{uv} models the tendency that u and v share the same label. We denote by q_u and p_u the prior knowledge and posterior knowledge of the node u, respectively. In RW-based methods, q_u and p_u are the prior and posterior reputation scores of u, respectively, and they represent relative benignness of nodes. In LBP-based methods, q_u and p_u are the prior and posterior probabilities that node u is a Sybil, respectively.

2.1.1 Additive Local Rule of RW-based Methods

State-of-the-art RW-based methods Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012); Boshmaf et al. (2015) first assign prior reputation scores for every node using a training dataset. Then they iteratively apply the following local rule to every node:

$$p_u = (1-\alpha) \sum_{v \in \Gamma_u} \underbrace{p_v \frac{w_{uv}}{d_v}}_{\text{neighbor influence}} + \alpha \underbrace{q_u}_{\text{prior knowledge}}, \tag{2.1}$$

where $\alpha \in [0, 1]$ is called a restart probability of the random walk. We note that SybilRank uses a restart probability of 0 and normalizes the final reputation scores by node degrees.

We have two observations for the additive local rule. First, the neighbor influence from a neighbor v to u is a fraction of v's current reputation score p_v, and the fraction is proportional to the edge weight w_{uv}. Second, this local rule combines the prior knowledge and the neighbor influences *linearly* to update the posterior knowledge about a node.

2.1.2 Multiplicative Local Rule of LBP-based Methods

SybilBelief Gong et al. (2014a), a LBP-based method, associates a binary random variable x_u with each node u, where $x_u = 1$ indicates that u is Sybil while $x_u = -1$ indicates that u is benign. Then, q_u and p_u are the prior and posterior probabilities that $x_u = 1$, respectively. SybilBelief first assigns the prior probabilities for nodes using a set of labeled benign nodes and/or a set of labeled Sybils, and then it iteratively applies the following local rule Gong et al. (2014a):

$$\underbrace{m_{vu}(x_u)}_{\text{neighbor influence}} = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v, x_u) \prod_{z \in \Gamma_v / u} m_{zv}(x_v) \tag{2.2}$$

$$p_u = \frac{\overbrace{q_u}^{\text{prior knowledge}} \prod_{v \in \Gamma_u} m_{vu}(1)}{q_u \prod_{v \in \Gamma_u} m_{vu}(1) + (1 - q_u) \prod_{v \in \Gamma_u} m_{vu}(-1)}, \tag{2.3}$$

where node potential $\phi_v(x_v)$ and edge potential $\varphi_{vu}(x_v, x_u)$ are defined as follows:

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

We also have two observations for the multiplicative local rule. First, *this local rule explicitly models neighbor influences*. Specifically, the neighbor influence from a neighbor v to u (i.e., $m_{vu}(x_u)$) is defined in Equation 2.2. To compute the neighbor influence $m_{vu}(x_u)$, u's neighbor v needs to multiply the neighbor influences from all its neighbors except u. Second, according to Equation 2.3, this local rule combines the neighbor influences with the prior probability *nonlinearly*.

2.1.3 Comparing Additive Local Rule with Multiplicative Local Rule

LBP-based multiplicative local rule can tolerate a relatively larger fraction of label noise because of its nonlinearity Gong et al. (2014a), and it can leverage both labeled benign nodes and labeled Sybils. However, LBP-based multiplicative local rule is space and time inefficient because it requires a large amount of space and time to maintain the neighbor influences associated with every edge, and methods using this local rule are not guaranteed to converge.

CHAPTER 3. COLLECTIVE CLASSIFICATION ON UNDIRECTED GRAPHS FOR SYBIL DETECTION IN SOCIAL NETWORKS

3.1 Introduction

Most collective classification methods Yu et al. (2006, 2008); Danezis and Mittal (2009); Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012); Boshmaf et al. (2015); Liu et al. (2016); Jia et al. (2017a); Zhang et al. (2016); Gong et al. (2014a); Gao et al. (2018); Fu et al. (2017); Wang et al. (2017a) for Sybil detection can be grouped into Random Walk (RW)-based methods and Loop Belief Propagation (LBP)-based methods. RW-based methods Yu et al. (2006, 2008); Danezis and Mittal (2009); Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012); Boshmaf et al. (2015); Liu et al. (2016); Jia et al. (2017a) suffer from the major limitations: 1) they can only leverage either labeled benign users or labeled Sybils in the training dataset, but not both, which limits their detection accuracies; and 2) they are not robust to label noise in the training dataset. The label of a user is noisy if the label is incorrect. Label noise often exists in practice due to human mistakes when manually labeling users Thomas et al. (2011); Wang et al. (2013). LBP-based methods Gong et al. (2014a); Gao et al. (2018); Fu et al. (2017) suffer from the major limitations: 1) they cannot guarantee convergence on real-world social networks; 2) they are not scalable; and 3) they do not have theoretically guaranteed performance. The first limitation makes LBP-based methods sensitive to the number of iterations that the methods run.

Our work: We propose a novel collective classification method, called SybilSCAR, to perform Sybil detection in undirected social networks. SybilSCAR combines the advantages of existing RW-based methods and LBP-based methods, while overcoming their limitations. Specifically, SybilSCAR is Scalable, Convergent, Accurate, and Robust to label noise.

First, we design a novel local rule that integrates the advantages of both RW-based methods and LBP-based methods, while overcoming their limitations. SybilSCAR iteratively applies our local rule to every user. Our local rule, like RW-based methods and LBP-based methods, leverages the *homophily property* of social networks. Homophily means that two linked users share the same label with a high probability. In our local rule, we associate a weight with each edge, which represents the probability that the two corresponding users have the same label. For a neighbor v of u, our local rule models v's influence (we call it *neighbor influence*) to u's label as the probability that u is a Sybil, given v's information alone. Our local rule combines neighbor influences and prior knowledge about a user in a multiplicative way to update knowledge about the user's label. Moreover, we linearize the multiplicative local rule in order to make SybilSCAR convergent.

Second, we evaluate SybilSCAR and compare it with state-of-the-art RW-based methods and LBP-based methods both theoretically and empirically. Theoretically, we derive a bound on the number of Sybils that are accepted into a social network for SybilSCAR. Our bound is tighter than those of the existing methods. Moreover, we analyze the condition when SybilSCAR is guaranteed to converge. Empirically, we compare SybilSCAR with SybilRank Cao et al. (2012), a state-of-the-art RW-based method, and SybilBelief Gong et al. (2014a), a state-of-the-art LBP-based method, using 1) three real-world social networks with synthesized Sybils and 2) a large-scale Twitter dataset (41.7M users and 1.2B edges) with real Sybils. Our empirical results demonstrate that 1) SybilSCAR achieves better detection accuracies than SybilRank and SybilBelief, 2) SybilSCAR is robust to larger label noise than SybilRank, and is as robust as SybilBelief; 3) SybilSCAR is as space and time efficient as SybilRank, but is several times more space efficient and one order of magnitude more time efficient than SybilBelief; 4) SybilSCAR and SybilRank are convergent, but SybilBelief is not. For instance, in the large Twitter dataset, among the top-10K users that are predicted to be most likely Sybils by SybilRank, SybilBelief, and SybilSCAR, 0.33%, 77.5%, and 95.8% of them are real Sybils, respectively.

In summary, our key contributions are as follows:

- We design a novel local rule for undirected graphs and develop a novel collective classification method, called SybilSCAR, to detect Sybils in social networks. SybilSCAR is convergent, scalable, robust to label noise, and more accurate than existing methods.

- We evaluate SybilSCAR both theoretically and empirically, and compare it with a state-of-the-art RW-based method and a state-of-the-art LBP-based method. Our theoretical results show that our method has a tighter bound on the number of Sybils that are falsely accepted into a social network than existing methods. Our empirical results on multiple social network datasets demonstrate that SybilSCAR significantly outperforms the state-of-the-art RW-based method in terms of accuracy and robustness to label noise, and that SybilSCAR outperforms the state-of-the-art LBP-based method in terms of accuracy, scalability, and convergence.

3.2 Problem Definition

We formally define our undirected graph-based Sybil detection problem in social networks, introduce our design goals, and describe the threat model we consider in the paper.

3.2.1 Undirected Graph-based Sybil Detection

Suppose we are given an undirected social network $G = (V, E)$, where a node $v \in V$ represents a user and an edge $(u, v) \in E$ indicates a (mutual) relationship between u and v. $|V|$ and $|E|$ are number of nodes and edges, respectively. For instance, on Facebook, an edge (u, v) could mean that u is in v's friend list and vice versa. On Twitter, an edge (u, v) could mean that u follows v. Our purpose is to design the local rule defined on the social networks to perform Sybil detection. Accordingly, we call this problem local rule-based Sybil detection.

Definition 1 (Undirect Graph-based Sybil Detection)**.** *Suppose we are given an undirected social network and a training dataset consisting of some labeled Sybils and labeled benign nodes. Undirect graph-based Sybil detection is to predict the label of each unlabeled node by leveraging the graph structure of the undirected social network.*

3.2.2 Design Goals

1) Leveraging both labeled benign users and Sybils: Social network service providers often have a set of labeled benign users and labeled Sybils. For instance, verified users on Twitter or Facebook can be treated as labeled benign users; users spreading spam or malware can be treated as labeled Sybils, which can be obtained through manual inspection Cao et al. (2012) or crowdsourcing Wang et al. (2013). Our method should be able to leverage both labeled benign users and labeled Sybils to enhance detection accuracy.

2) Robust to label noise: A given label of a user is noisy if it does not match the user's true label. Labeled users may have noisy labels. For instance, an adversary could compromise a labeled benign user or make a Sybil whitelisted as a benign user. In addition, labels obtained through manual inspection, especially crowdsourcing, often contain noises due to human mistakes Wang et al. (2013). We target a method that is robust to a minority fraction of incorrect labels.

3) Scalable: Real-world social networks often have hundreds of millions of users and edges. Therefore, our method should be scalable and easily parallelizable.

4) Convergent: Existing methods and our method are iterative methods. Convergence makes it easy to determine when to stop an iterative method. It is hard to set the best number of iterations for an iterative method that is not convergent. Therefore, our method should be convergent.

5) Theoretical guarantee: Our method should have a theoretical guarantee on the number of Sybils that can be falsely accepted into a social network. This theoretical guarantee is important for security-critical applications that leverage social networks, e.g., social network based Sybil defense in peer-to-peer and distributed systems Yu et al. (2006), and social network based anonymous communications Danezis et al. (2010).

Existing RW-based SybilGuard Yu et al. (2006) and SybilLimit Yu et al. (2008) do not satisfy 1), 2), and 3); SybilInfer Danezis and Mittal (2009) only satisfies the requirement 4); SybilRank Cao et al. (2012) and Íntegro Boshmaf et al. (2015) do not satisfy 1) and 2); CIA Yang et al. (2012) does not satisfy 1), 2), and 5). Existing LBP-based SybilBelief Gong et al. (2014a) and SybilFuse Gao et al. (2018) do not satisfy 3), 4), and 5).

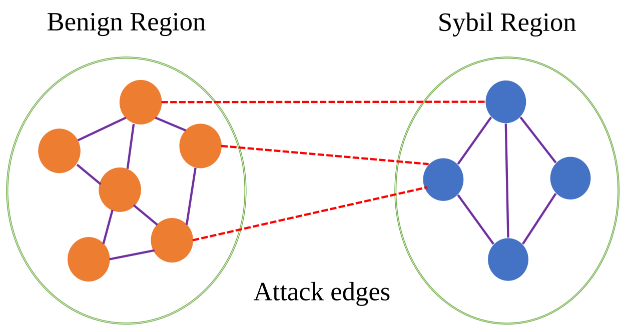


Figure 3.1 Benign region, Sybil region, and attack edges.

3.2.3 Threat Model

We call the subgraph containing all benign nodes and edges between them the *benign region*, and call the subgraph containing all Sybil nodes and edges between them the *Sybil region*. Edges between the two regions are called *attack edges*. Figure 3.1 illustrates these concepts.

One basic assumption under structure-based Sybil detection methods is that the benign region and the Sybil region are sparsely connected (i.e., the number of attack edges is relatively small), compared with the edges among the two regions. In other words, most benign users would not establish trust relationships with Sybils. We note that this assumption is equivalent to requiring that the social network follows *homophily*, i.e., two linked nodes share the same label with a high probability. For an extreme example, if the benign region and the Sybil region are separated from each other, then the social network has a perfect homophily, i.e., every two linked nodes have the same label. Note that, it is of great importance to obtain social networks that satisfy this assumption, otherwise the detection accuracies of collective classification methods are limited. For instance, Yang et al. Yang et al. (2014) showed that RenRen *friendship* social network does not satisfy this assumption, and thus the performance of collective classification methods are unsatisfactory. However, Cao et al. Cao et al. (2012) found that Tuenti, the largest online social network in Spain, satisfies the homophily assumption, and thus SybilRank can detect a large amount of Sybils in Tuenti.

Generally speaking, there are two ways for service providers to construct a social network that satisfies homophily. One way is to approximately obtain trust relationships between users by looking into user interactions Wilson et al. (2009), predicting tie strength Gilbert and Karahalios (2009), asking users to rate their social contacts Wei et al. (2012), etc. The other way is to preprocess the network structure so that collective classification methods are suitable to be applied. Specifically, analysts could detect and remove compromised benign nodes (e.g., front peers) Wang and Nakao (2010), or employ feature-based classifier to filter Sybils, so as to decrease the number of attack edges and enhance the homophily. For instance, Alvisi et al. Alvisi et al. (2013) showed that if the attack edges are established randomly, simple feature-based classifiers are sufficient to enforce Sybils to be suitable for structure-based Sybil detection. We note that the reason why the RenRen friendship social network did not satisfy homophily in the study of Yang et al. is that RenRen even didn't deploy simple feature-based classifiers at that time Yang et al. (2014).

Formally, we measure homophily as the fraction of edges in the social network that are not attack edges. For the same benign region and Sybil region, more attack edges indicate weaker homophily. As we will demonstrate in our empirical evaluations, our SybilSCAR can tolerate weaker homophily than existing methods.

When analyzing the theoretical bound on the falsely accepted Sybils, SybilSCAR further assumes that the iterative process of SybilSCAR converges fast in the benign region, which is similar to the fast mixing assumption of RW-based methods.

3.3 Design of SybilSCAR

Under our local rule-based framework in Chapter 2, we know that designing a new Sybil detection method is reduced to designing a new local rule. Therefore, we aim to design a novel local rule in this Section. Then, we describe how we design SybilSCAR based on the new local rule.

3.3.1 Our New Local Rule

We aim to design a local rule that integrates the advantages of both RW-based and LBP-based local rules, while overcoming their limitations. Roughly speaking, our idea is to leverage the *multiplicativeness* like LBP-based local rule to be robust to label noise, while *avoiding maintaining neighbor influences* to be as space and time efficient as RW-based local rule. Next, we show how we model neighbor influences and combine neighbor influences with prior knowledge.

Neighbor influence: We denote w_{vu}, which ranges from 0 to 1, as the *homophily strength* of the edge (u, v). $w_{uv} > 0.5$ means that u and v are in a *homogeneous relationship*, i.e., they tend to share the same label; $w_{uv} < 0.5$ means that u and v are in a *heterogeneous relationship*, i.e., they tend to have the opposite labels; and $w_{uv} = 0.5$ means that u and v are not correlated. We associate a binary random variable x_u with a node u, where $x_u = 1$ and $x_u = -1$ mean that u is a Sybil and benign node, respectively. We denote by f_{vu} the neighbor influence from v to u, where v is a neighbor of u. We model f_{vu} as the influence v has on u's label, given v's status and w_{vu}. Specifically, $f_{vu} > 0.5$ means that v has a positive influence on u (i.e., v tends to make u to be positive); $f_{vu} < 0.5$ means that v has a negative influence on u (i.e., v tends to make u to be negative); $f_{vu} < 0.5$ means that v has no influence on u (i.e., v cannot decide us label).

Based on the homophily property, f_{vu} should meet the following constraints:

$$p_v = 0.5 \text{ or } w_{vu} = 0.5 \Longrightarrow f_{vu} = 0.5 \tag{3.1}$$

$$p_v > 0.5 \text{ and } w_{vu} > 0.5 \Longrightarrow f_{vu} > 0.5 \tag{3.2}$$

$$p_v < 0.5 \text{ and } w_{vu} > 0.5 \Longrightarrow f_{vu} < 0.5 \tag{3.3}$$

$$p_v > 0.5 \text{ and } w_{vu} < 0.5 \Longrightarrow f_{vu} < 0.5 \tag{3.4}$$

$$p_v < 0.5 \text{ and } w_{vu} < 0.5 \Longrightarrow f_{vu} > 0.5, \tag{3.5}$$

where Equation 3.1 means that we cannot learn anything about u's label if v's label is undecidable (i.e., $p_v = 0.5$) or u and v are uncorrelated (i.e., $w_{vu} = 0.5$); Equations 3.2 and 3.3 mean that u and v tend to share the same label if they are in a homogeneous relationship; and Equations 3.4 and 3.5 mean that u and v tend to have opposite labels if they are in a heterogeneous relationship.

To satisfy all the above constraints in Equations 3.1-3.5, we model f_{vu} as follows:

$$f_{vu} = p_v w_{vu} + (1 - p_v)(1 - w_{vu}). \tag{3.6}$$

We can verify that f_{vu} in Equation 3.6 indeed satisfies the above constraints. Another way to interpret our model of f_{vu} is that w_{vu} is the probability that u and v have the same label and f_{vu} is the probability that u is a Sybil conditioned on w_{vu} and p_v. In our model, it is straightforward to compute a neighbor influence f_{vu}.

Combining neighbor influences with prior: In our local rule, a node's posterior probability of being Sybil is updated by combining its neighbor influences with its prior probability of being Sybil. In order to tolerate label noise, we leverage the multiplicative local rule in LBP-based methods. Specifically, we have:

$$p_u = \frac{q_u \prod_{v \in \Gamma_u} f_{vu}}{q_u \prod_{v \in \Gamma_u} f_{vu} + (1 - q_u) \prod_{v \in \Gamma_u} (1 - f_{vu})}. \tag{3.7}$$

However, methods that iteratively apply the above multiplicative local rule to every user are not guaranteed to converge. Therefore we further *linearize* Equation 3.7. We first define two concepts *residual variable* and *residual vector*.

Definition 2 (Residual Variable and Vector)**.** *We define the residual of a variable y as $\hat{y} = y - 0.5$; and we define the residual vector $\hat{\mathbf{y}}$ of $\mathbf{y}$ as $\hat{\mathbf{y}} = [y_1 - 0.5, y_2 - 0.5, \cdots]$.*

With above definition, we denote $\hat{w}_{vu}$ as the residual homophily strength. Moreover, by substituting variables in Equation 3.6 with their corresponding residuals, we have the residual neighbor influence $\hat{f}_{vu}$ as follows:

$$\hat{f}_{vu} = 2\hat{p}_v \hat{w}_{vu}. \tag{3.8}$$

Based on the approximations $\ln(1 + x) \approx x$ and $\ln(1 - x) \approx -x$ when x is small, we have the following theorem, which linearizes Equation 3.7.

Theorem 3. *The residual posterior probability of being Sybil for a node u can be linearized as:*

$$\hat{p}_u = \hat{q}_u + \sum_{v \in \Gamma(u)} \hat{f}_{vu}. \tag{3.9}$$

Proof. See Appendix . □

By combining Equation 4.16 and Equation .23, we obtain our new local rule as follows:

$$\textbf{Our local rule:} \quad \hat{p}_u = \hat{q}_u + 2 \sum_{v \in \Gamma(u)} \hat{p}_v \hat{w}_{vu}. \tag{3.10}$$

3.3.2 SybilSCAR Algorithm

Our SybilSCAR iteratively applies our local rule to every node to compute the posterior probabilities. Suppose we are given a set of labeled Sybils which we denote as L_s and a set of labeled benign nodes which we denote as L_b. SybilSCAR first utilizes L_s and L_b to assign a prior probability of being a Sybil for all nodes. Specifically,

$$q_u = \begin{cases} 0.5 + \theta & \text{if } u \in L_s \\ 0.5 - \theta & \text{if } u \in L_b \\ 0.5 & \text{otherwise,} \end{cases} \tag{3.11}$$

where $\theta > 0$ indicates that we assign a higher prior probability of being a Sybil to labeled Sybils. Considering that the labels might have noise, we will set θ to be smaller than 0.5. In practice, these prior probabilities can also be obtained from feature-based methods. Specifically, for each user we can leverage a binary classifier, trained using user's local features, to produce the probability of being a Sybil, which can then be treated as the user's prior probability. With such prior probabilities, SybilSCAR iteratively applies our local rule in Equation 3.10 to update residual posterior probabilities of all nodes.

Representing SybilSCAR as a matrix form: For convenience, we denote by a vector $\mathbf{q}$ the prior probability of being a Sybil for all nodes, i.e., $\mathbf{q} = [q_1; q_2; \cdots; q_{|V|}]$. Similarly, we denote by a vector $\mathbf{p}$ the posterior probability of all nodes, i.e., $\mathbf{p} = [p_1; p_2; \cdots; p_{|V|}]$. Moreover, we denote $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}$ as the residual prior probability vector and residual posterior probability vector of all nodes, respectively. We denote $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ as the adjacency matrix of the social graph, where the uth row represents the neighbors of u. Formally, if there exists an edge (u, v) between nodes u and v, then the entry $A_{uv} = A_{vu} = 1$, otherwise $A_{uv} = A_{vu} = 0$. Moreover, we denote $\hat{\mathbf{W}}$ as the corresponding residual homophily strength matrix , where $\hat{w}_{vu} = 0$ if $A_{vu} = 0$. With these notations, we can represent our SybilSCAR as iteratively applying the following equation:

$$\hat{\mathbf{p}}^{(t)} = \hat{\mathbf{q}} + 2\hat{\mathbf{W}}\hat{\mathbf{p}}^{(t-1)}, \tag{3.12}$$

where $\hat{\mathbf{p}}^{(t)}$ is the residual posterior probability vector in the tth iteration. Initially, we set $\hat{\mathbf{p}}^{(0)} = \hat{\mathbf{q}}$.

We stop running SybilSCAR when the relative errors of residual posterior probabilities between two consecutive iterations is smaller than some threshold δ or it reaches the predefined number of maximum iterations T. After SybilSCAR halts, we predict u to be a Sybil if $p_u > 0.5$, otherwise we predict u to be benign.

In this paper, we consider the following two cases for the homophily strength $\hat{\mathbf{W}}$. Although we study these two settings, we believe that learning the homophily strength for each edge would be a valuable future work.

SybilSCAR-C: In this variant, we use a constant homophily strength for all edges, i.e., $\hat{w}_{vu} = \hat{w}$.

SybilSCAR-D: In this variant, we use a degree-normalized homophily strength for each edge, i.e., $\hat{w}_{vu} = \frac{1}{2d_u}$, where d_u is the degree of node u. The intuition is that when a node has many neighbors, each neighbor has a small influence on the node. In this variant, a node's residual posterior probability is the sum of its residual prior probability and the average residual posterior probability of its neighbors. We note that SybilSCAR-D essentially uses the RW-based *neighbor influence* proposed by SybilWalk Jia et al. (2017a). The differences with SybilWalk include 1) SybilWalk resets the residual posterior probabilities of labeled nodes to their residual prior probabilities in each iteration, and 2) SybilWalk does not consider prior probabilities of unlabeled nodes.

3.4 Theoretical Analysis

We first analyze the convergence condition of SybilSCAR. Then we analyze its peformance bound. Finally, we analyze its computational complexity.

3.4.1 Convergence Condition

We analyze the condition when SybilSCAR converges.

Lemma 4 (Sufficient and Necessary Convergence Condition for a Linear System Saad (2003))**.** *Suppose we are given an iterative linear process:* $\mathbf{y}^{(t)} \leftarrow \mathbf{c} + \mathbf{M}\mathbf{y}^{(t-1)}$*. The linear process converges with any initial choice* $\mathbf{y}^{(0)}$ *if and only if the spectral radius*[1] *of* $\mathbf{M}$ *is smaller than 1, i.e.,* $\rho(\mathbf{M}) < 1$*.*

Proof. See Saad (2003). □

Based on Equation 4.17 and Lemma 4, we are able to analyze the convergence condition of SybilSCAR.

Theorem 5 (Sufficient and Necessary Convergence Condition of SybilSCAR)**.** *The sufficient and necessary condition that makes SybilSCAR converge is equivalent to*

$$\rho(\hat{\mathbf{W}}) < \frac{1}{2}. \tag{3.13}$$

Proof. By directly using Lemma 4 in Equation 4.17. □

Theorem 5 provides a strong sufficient and necessary convergence condition. However, in practice using Theorem 5 is computationally expensive, as it involves computing the largest eigenvalue with respect to spectral radius of $\hat{\mathbf{W}}$. Hence, we instead derive a *sufficient condition* for SybilSCAR's convergence, which enables us to set $\hat{w}$ with cheap computation. Specifically, our sufficient condition is based on the fact that any norm is an upper bound of the spectral radius Derzko and Pfeffer (1965), i.e., $\rho(\mathbf{M}) \leq \|\mathbf{M}\|$, where $\|\cdot\|$ indicates some matrix norm. In particular, we use the induced l_∞ matrix norm $\|\cdot\|_\infty$ [2]. In this way, our sufficient condition for convergence is as follows:

Theorem 6 (Sufficient Convergence Condition of SybilSCAR)**.** *A sufficient condition that makes SybilSCAR converge is*

$$\|\hat{\mathbf{W}}\|_\infty < \frac{1}{2}. \tag{3.14}$$

Proof. As $\rho(\hat{\mathbf{W}}) \leq \|\hat{\mathbf{W}}\|_\infty$, we achieve the sufficient condition by enforcing $2\|\hat{\mathbf{W}}\|_\infty < 1$, and thus $\|\hat{\mathbf{W}}\|_\infty < \frac{1}{2}$. □

[1] *The spectral radius of a square matrix is the maximum of the absolute values of its eigenvalues.*
[2] $\|\mathbf{M}\|_\infty = \max_i \sum_j |\mathbf{M}_{ij}|$, the maximum absolute row sum of the matrix.

Table 3.1 Summary of theoretical guarantees of various collective classification methods. g is the number of attack edges (sum of weights on the attack edge for Íntegro) and $d(B)_{\min}$ is the minimum node degree in the benign region. SybilGuard requires $g = o(\sqrt{|V|}/\log|V|)$. "–" means the corresponding bound is unknown.

Method	#Accepted Sybils
SybilGuard Yu et al. (2006)	$O(g\sqrt{\lvert V\rvert}\log\lvert V\rvert)$
SybilLimit Yu et al. (2008)	$O(g\log\lvert V\rvert)$
SybilInfer Danezis and Mittal (2009)	–
SybilRank Cao et al. (2012)	$O(g\log\lvert V\rvert)$
CIA Yang et al. (2012)	–
Íntegro Boshmaf et al. (2015)	$O(g\log\lvert V\rvert)$
SybilBelief Gong et al. (2014a)	–
SybilSCAR-D	$O(\frac{g\log\lvert V\rvert}{d(\mathcal{S})})$

Theorem 6 provides a guideline for us to set $\hat{w}$, i.e., if $\hat{w}$ is smaller than the inverse of 2 times of the maximum node degree, SybilSCAR converges. In practice, however, 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 note that SybilSCAR can still converge when replacing the maximum node degree with the average node degree.

Next, we derive the sufficient convergence condition for SybilSCAR-C and SybilSCAR-D, respectively.

SybilSCAR-C: By applying Theorem 6, we have the sufficient condition for SybilSCAR-C to converge as $\hat{w} < \frac{1}{2\|\mathbf{A}\|_\infty} = \frac{1}{2\max_{u\in V} d_u}$. Note that our result provides a guideline to set $\hat{w}$, i.e., once $\hat{w}$ is smaller than the inverse of 2 times of the maximum node degree, SybilSCAR-C is guaranteed to converge. In practice, however, 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 found that SybilSCAR-C can still converge when replacing the maximum node degree with the average node degree.

SybilSCAR-D: In this case, the summation of each row of $\hat{\mathbf{W}}$ has a fixed value $\frac{1}{2}$. Therefore, $\|\mathbf{W}\|_\infty = \frac{1}{2}$. In practice, it is often $\rho(\hat{\mathbf{W}}) < \|\hat{\mathbf{W}}\|_\infty$, which implies that $\rho(\hat{\mathbf{W}}) < \frac{1}{2}$. Therefore, SybilSCAR-D is also convergent.

3.4.2 Security Guarantee

Existing RW-based methods: Some existing RW-based Sybil detection methods Yu et al. (2006, 2008); Danezis and Mittal (2009); Cao et al. (2012); Boshmaf et al. (2015) have theoretical guarantees on the number of Sybils that are falsely accepted into a social network. For instance, Table 3.1 shows the theoretical guarantees of some representative methods. These guarantees are achieved based on the assumption that the benign region of the social network is *fast-mixing* Levin et al. (2009). Roughly speaking, a graph is fast mixing if a random walk on the graph converges to its stationary distribution in $O(\log |V|)$ iterations.

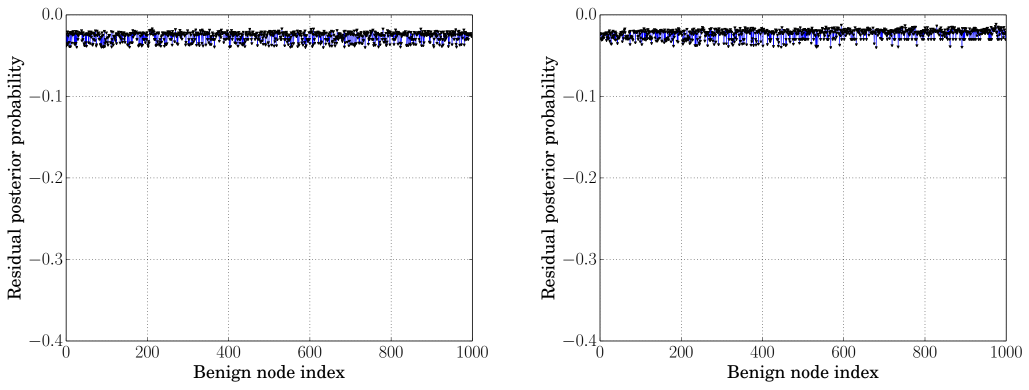
SybilSCAR: We will derive security guarantee for a "weaker" version of SybilSCAR-D. In SybilSCAR-D, in each iteration, a node's residual posterior probability is the sum of its residual prior probability and the average residual posterior probability of its neighbors. In other words, in each iteration, a node's prior probability is injected to influence the dynamics of nodes' residual posterior probabilities, which makes it harder to analyze the dynamics of residual posterior probabilities. Therefore, we consider a weaker version of SybilSCAR-D, in which the nodes' residual prior probabilities are only injected in the initialization step. In other words, we have

$$\hat{\mathbf{p}}^{(0)} = \hat{\mathbf{q}} \tag{3.15}$$

$$\hat{p}_u^{(t+1)} = \sum_{v \in \Gamma(u)} \frac{\hat{p}_v^{(t)}}{d_u}. \tag{3.16}$$

This version of SybilSCAR-D has a converged solution that every node has the same residual posterior probability, i.e., $\hat{p}_u = \pi$ for every node u is a solution for SybilSCAR-D. We have the following security guarantee for this version of SybilSCAR-D:

Theorem 7. *Suppose SybilSCAR-D only leverages the nodes' prior probabilities in the initialization step, residual posterior probabilities in the benign region converge in $O(\log |V|)$ iterations (this is similar to the fast mixing assumption of RW-based methods), the attacker randomly establishes g attack edges, and we are only given some labeled benign nodes. Then, the total number of Sybils whose residual posterior probabilities of being Sybil are lower than those of certain benign nodes is bounded by $O(\frac{g \log |V|}{d(S)})$, where $d(S)$ is the average node degree in the Sybil region.*



(a) ER model (b) PA model

Figure 3.2 Residual posterior probabilities of unlabeled benign nodes.

Proof. See Appendix . □

Theorem 7 implies that when Sybils are more densely connected among themselves (i.e., the average degree $d(S)$ is larger), it is easier for SybilSCAR to detect them. An intuitional explanation is that, when the Sybil region is more dense, a larger proportion of the residual posterior probabilities would be propagated among the Sybil region. Table 3.1 summarizes the theoretical performance bound of existing collective classification methods. For SybilRank, Íntegro, and CIA, the metric *#accepted Sybils* means the number of Sybils that are ranked lower than certain benign nodes. For the rest of methods, #accepted Sybils means the number of Sybils that are classified as benign. As we can see, our SybilSCAR achieves the tightest bound on the number of falsely accepted Sybils. We note that deriving security guarantee for SybilSCAR-C is still an open challenge. However, as we will demonstrate in our experiments, SybilSCAR-C outperforms SybilSCAR-D.

One key assumption of Theorem 7 is that residual posterior probabilities in the benign region converge after $O(\log |V|)$ iterations. In other words, nodes in the benign region have similar residual posterior probabilities after $O(\log |V|)$ iterations. We validate this assumption via simulations. Specifically, we synthesize a benign region and a Sybil region with 1,000 nodes and an average degree of 40 via the Erdos–Renyi (ER) model Erdos and Rényi (1960) or the Preferential Attachment (PA) model Barabási and Albert (1999); we randomly add 1,000 attack edges between the two regions;

and we randomly label 10 benign nodes as the training set. Figure 3.2 shows the residual posterior probabilities of unlabeled benign nodes after $\log|V|$ iterations. We observe that unlabeled benign nodes have similar residual posterior probabilities.

3.4.3 Complexity Analysis

SybilSCAR (both SybilSCAR-C and SybilSCAR-D), state-of-the-art RW-based methods Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012), and LBP-based method Gong et al. (2014a) have the same space complexity, i.e., $O(|E|)$, and their time complexity is $O(t|E|)$, where t is the number of iterations. Although SybilSCAR and SybilBelief (a LBP-based method) have the same asymptotic space and time complexity, SybilSCAR is several times more space efficient and significantly more time efficient than SybilBelief in practice, as we demonstrate in our experiments. This is because SybilBelief needs to store neighbor influences (i.e., $m_{vu}(x_u)$) in both directions of every edge and update them in every iteration.

Parallel implementation: SybilSCAR, state-of-the-art RW-based methods Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012), and LBP-based methods Gong et al. (2014a); Gao et al. (2018) can be implemented in parallel. Specifically, we can divide nodes into groups, and a thread or computer applies the corresponding local rule to a group of nodes iteratively.

3.5 Empirical Evaluations

We compare our SybilSCAR-C and SybilSCAR-D with SybilRank Cao et al. (2012), a state-of-the-art RW-based method, and SybilBelief Gong et al. (2014a), a state-of-the-art LBP-based method, in terms of accuracy, robustness to label noise, scalability, and convergence.

3.5.1 Dataset Description

We use 1) three real-world social networks with synthesized Sybils and 2) a large-scale Twitter dataset with real Sybils for evaluations. Table 3.2 shows some basic statistics about our datasets.

Table 3.2 Dataset statistics.

Dataset	#Nodes	#Edges	Ave. degree
Facebook	4,039	88,234	43.69
Enron	33,696	180,811	10.73
Epinions	75,877	811,478	21.39
Twitter	41,652,230	1,202,513,046	57.74

Social networks with synthesized Sybils: We use a real social graph as the benign region while synthesizing the Sybil region and adding attack edges between the two regions uniformly at random. There are different ways to synthesize the Sybil region. For instance, we can use a network model (e.g., Preferential Attachment (PA) model Barabási and Albert (1999)) to generate a Sybil region. A Sybil region that is synthesized by a network model might be structurally very different from the benign region, e.g., although the PA model can generate graphs that have similar degree distribution with real social networks, the generated graphs have very small clustering coefficients, which is very different from real-world social networks. Such structural difference could bias Sybil detection results Alvisi et al. (2013). Moreover, a Sybil region synthesized by a network model like PA does not have community structures, making it unrealistic. Therefore, following recent studies Alvisi et al. (2013); Gong et al. (2014a), we consider a Sybil attack in which the Sybil region is a replicate of the benign region. This way of synthesizing the Sybil region can avoid the structural difference between the two regions, and both Sybil region and benign region have complex community structures.

We utilize three social networks, i.e., Facebook (4,039 nodes and 88,234 edges), Enron (33,696 nodes and 180,811 edges), and Epinions (75,877 nodes and 811,478 edges), to represent different application scenarios. We obtained these datasets from SNAP (http://snap.stanford.edu/data/index.html). A node in Facebook dataset represents a user in Facebook, and two nodes are connected if they are friends. A node in Enron dataset represents an email address, and an edge between two nodes indicate at least one email was exchanged between the two corresponding email addresses. Epinions is a who-trust-whom online social network of a general consumer review site Epinions.com. The nodes in Epinions denote members of the site. And in order to maintain quality, Epinsons encourages users to specify which other users they trust, and uses the resulting web of the trust to order the product reviews seem by each person. For each social network, we use it as the benign region and replicate it as a Sybil region. Moreover, without otherwise mentioned, we add 1,000 attack edges uniformly at random.

Twitter dataset with real Sybils: We obtained a snapshot of a large-scale Twitter follower-followee network crawled by Kwak et al. Kwak et al. (2010). We transformed the follower-followee network into an undirected one via keeping an edge between two users if there are at least one directed edge between them. The undirected Twitter graph has 41,652,230 nodes and 1,202,513,046 edges, with an average degree of 57.74. To perform evaluation, we need ground truth labels of the users. Since the Twitter network includes users' Twitter IDs, we wrote a crawler to visit each user's profile using Twitter's API, which told us the status (i.e., active, suspended, or deleted) of each user. We found that 205,355 users were suspended by Twitter and we treated them as Sybils; 36,156,909 users were still active and we treated them as benign users. 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. Therefore, we treat them as unlabeled users. The average number of attack edges per Sybil is 181.55. Therefore, the Twitter network has a very weak homophily. Note that the number of benign users and the number of Sybils are very unbalanced, i.e., the number of labeled benign users is 176 times larger than the number of labeled Sybils.

Training and testing sets: For a social network with synthesized Sybils, we select 200 nodes uniformly at random and use them as a training dataset. For the Twitter dataset, we select 500,000 nodes uniformly at random and use them as a training dataset. The remaining benign and Sybil nodes are used as testing data.

3.5.2 Compared Methods

We compare SybilSCAR-C and SybilSCAR-D with SybilRank Cao et al. (2012), a state-of-the-art RW-based method, and SybilBelief Gong et al. (2014a), a state-of-the-art LBP-based method. In addition, we use random guessing as a baseline.

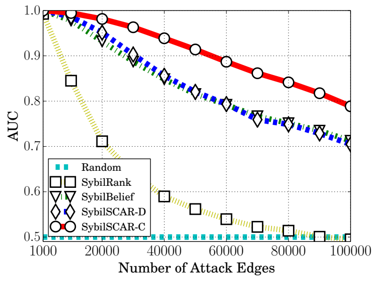
For SybilSCAR-C and SybilSCAR-D, we set $\theta = 0.1$ to consider possible label noises, i.e., we assign a prior probability 0.6, 0.4, and 0.5 to labeled Sybils, labeled benign users, and unlabeled nodes, respectively. We set $\delta = 10^{-3}$ and $T = 20$. Considering different average degrees of Facebook, Enron, Epinions, and Twitter, we set $\hat{w} = 0.01$, 0.04, 0.02, and 0.01 for SybilSCAR-C, respectively. We set the parameters of SybilRank and SybilBelief according to their papers. For instance, for SybilBelief, the edge weight is set to be 0.9 for all edges; SybilRank requires early termination, and we set the number of iterations as $\lceil \log(|V|) \rceil$.

We implemented SybilRank, SybilSCAR-C, and SybilSCAR-D in C++. We obtained a basic implementation of SybilBelief (in C++) from its authors and optimized the implementation. We performed all experiments on a Linux machine with 16GB memory and 8 cores.

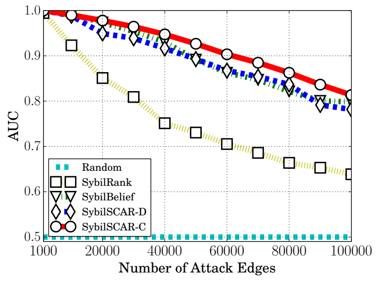
3.5.3 Results on Social Networks with Synthesized Sybils

Viswanath et al. (2010) demonstrated that Sybil detection methods can be treated as ranking mechanisms, and they can be evaluated using Area Under the Receiver Operating Characteristic Curve (AUC). Therefore, we adopt AUC to evaluate ranking accuracy. Suppose we rank nodes with respect to their posterior reputation/probability of being a Sybil in a descending order. AUC is the probability that a randomly selected Sybil ranks higher than a randomly selected benign node. Note that random guessing, which ranks all nodes uniformly at random, has an AUC of 0.5.

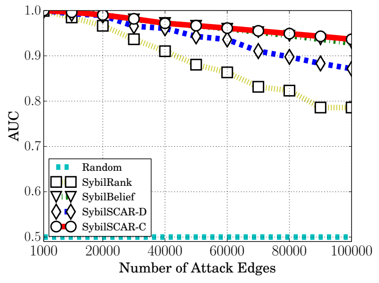
Figure 3.3 shows AUCs of the compared methods as we increase the number of attack edges from 1,000 to 100,000. We have three observations. First, when a social network has strong homophily, i.e., the number of attack edges is small, all the compared methods achieve very high AUCs. For instance, SybilRank, SybilBelief, SybilSCAR-C, and SybilSCAR-D all achieve AUCs that are close to 1 when the number of attack edges is less than 1,000. Second, SybilSCAR-C, SybilSCAR-



(a) Facebook



(b) Enron



(c) Epinions

Figure 3.3 AUCs of compared methods as the number of attack edges becomes large. SybilSCAR-C and SybilSCAR-D are substantially more accurate than SybilRank, and SybilSCAR-C is slightly more accurate than SybilBelief and SybilSCAR-D, when the number of attack edges is large.

D, and SybilBelief are substantially more accurate than SybilRank when the number of attack edges becomes large, i.e., the social networks have weak homophily. A possible reason is that SybilSCAR-C, SybilSCAR-D, and SybilBelief can leverage both labeled benign users and labeled Sybils in the training dataset. Third, SybilSCAR-C achieves slightly larger AUCs than SybilBelief and SybilSCAR-D. Compared with SybilBelief, SybilSCAR-C uses a new neighbor influence by directly modeling the homophily property of the social network. Compared with SybilSCAR-D, SybilSCAR-C uses a constant weight for all edges, which may make the labeled nodes have larger influence to their neighbors.

3.5.4 Results on the Large-scale Twitter Dataset

The AUCs of SybilRank, SybilBelief, SybilSCAR-D, and SybilSCAR-C are 0.37, 0.76, 0.77, and 0.80, respectively. SybilRank performs worse than random guessing. SybilSCAR-C performs better than SybilSCAR-D, which is comparable with SybilBelief. These results are consistent with those in social networks with synthesized Sybils, because the number of attack edges in the Twitter dataset is very large. We note that these AUCs are obtained via a *balanced* training dataset. Specifically, among the 500,000 nodes in the training dataset, benign nodes are much more than Sybils; we subsample some benign nodes such that we have the same number of benign nodes and Sybils,

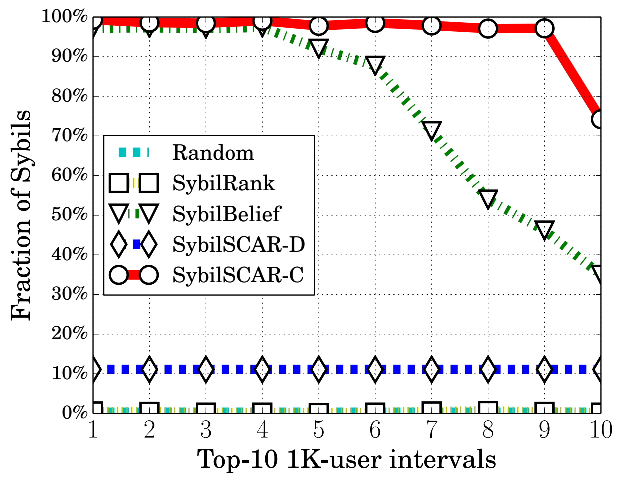


Figure 3.4 Fraction of Sybils in top ranked intervals on the Twitter dataset. SybilSCAR-C performs better than SybilBelief and SybilSCAR-D, while SybilSCAR-D and SybilBelief perform much better than SybilRank.

and we use them as a balanced training dataset. We found that all the methods have very low AUCs (worse than random guessing) if we use the original unbalanced training dataset consisting of the randomly sampled 500,000 nodes. It would be an interesting future work to theoretically understand the impact of balanced/unbalanced training dataset on the accuracy of these methods.

In practice, the ranking of users can be used as a priority list to guide human workers to manually inspect users and detect Sybils. In particular, inspecting users according to their rankings could aid human workers to detect more Sybils than inspecting randomly picked users, within the same amount of time. When ranking is used for such purpose, the number of Sybils in top-ranked users is important because human workers can only inspect a limited number of users. AUC measures the overall ranking performance, but it cannot tell Sybils among the top-ranked users. Therefore, we further compare the considered methods using the fraction of Sybils in top-ranked users. Specifically, for each method, we divide the top-10K users obtained by the method into 10 intervals, where each interval has 1K users. Figure 3.4 shows the fraction of Sybils in each 1K-user interval for the compared methods. First, SybilSCAR-C performs better than SybilBelief. Specifically, the fraction of Sybils ranges from 74.2% to 99.3% in the top-10 1K-user intervals for

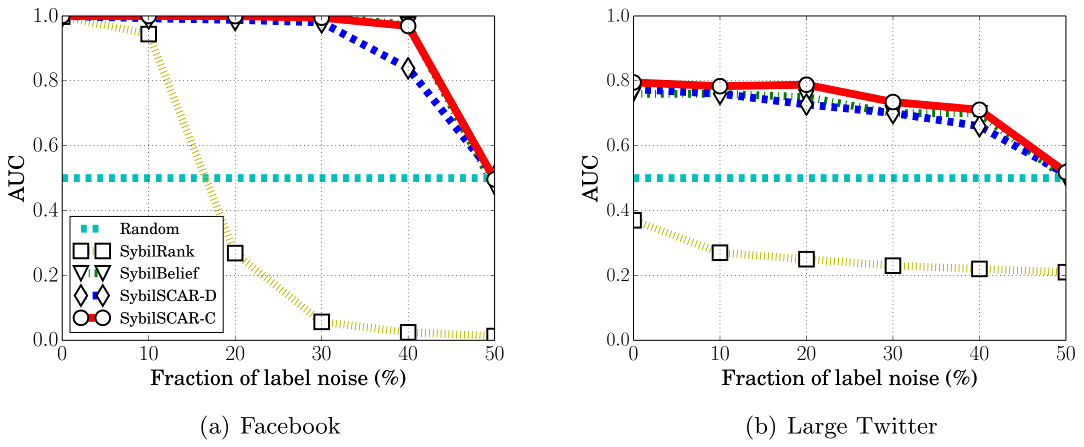


Figure 3.5 AUCs of SybilRank, SybilBelief, SybilSCAR-D, and SybilSCAR-C vs. level of label noise. SybilSCAR-C is slightly better than SybilSCAR-D and SybilBelief, while they are much more robust to label noise than SybilRank. Note that SybilBelief and SybilSCAR-C overlap in (a).

SybilSCAR-C, while the range is from 35.0% to 97.4% for SybilBelief. Second, SybilSCAR-C and SybilBelief outperform SybilSCAR-D, which demonstrates that the predefined constant edge weight is more informative than degree-normalized edge weight for ranking Sybils. Third, SybilRank is close to random guessing.

3.5.5 Robustness to label noise

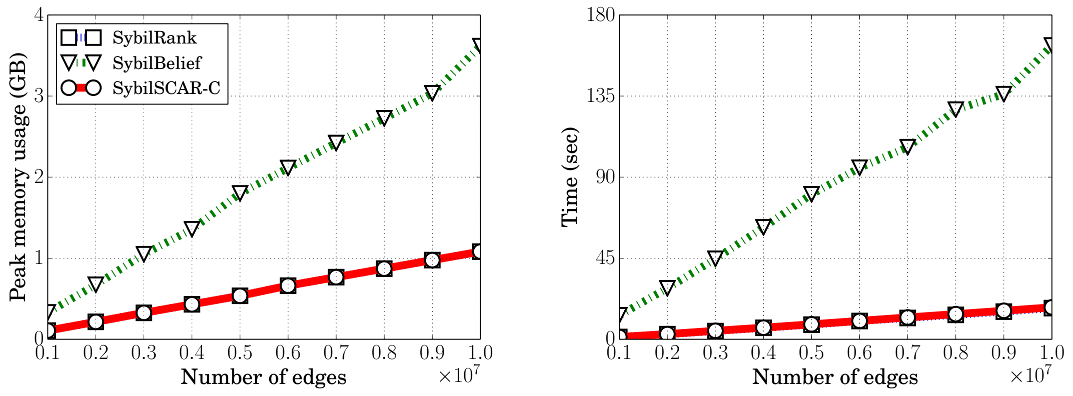
In practice, a training dataset might have noises, i.e., some labeled benign users are actually Sybils and some labeled Sybils are actually benign. Such noises could be introduced by human mistakes Wang et al. (2013). Thus, one natural question is how label noise impacts the accuracy of detection methods.

For a given level of noise $\tau\%$, we randomly choose $\tau\%$ of labeled Sybils in the training dataset and mislabel them as benign users; and we also sample $\tau\%$ of labeled benign users in the training dataset and mislabel them as Sybils. We vary $\tau\%$ from 10% to 50% with a step size of 10%. Note that we didn't perform experiments for $\tau\% > 50\%$ as all these methods cannot detect Sybils when a majority of labels are incorrect. Figure 3.5 shows the AUCs of SybilRank, SybilBelief, SybilSCAR-C, and SybilSCAR-D on Facebook (note that we have similar results on Enron and Epinions, and thus omit them for simplicity) and Twitter datasets against different levels of label noises. We observe that 1) SybilSCAR-C has the best robustness against label noise; 2) SybilBelief is slightly more robust than SybilSCAR-D to label noise; 3) SybilSCAR-C, SybilSCAR-D, and SybilBelief are more robust to label noise than SybilRank. For instance, on the Facebook dataset, SybilSCAR-C and SybilBelief can tolerate label noise up to 40%, SybilSCAR-D can tolerate label noise up to 30%, while SybilRank performs worse than random guessing when label noise is higher than 20%.

We believe it is an interesting future work to theoretically understand the robustness to label noise of different methods. In the following, we provide a possible explanation on why SybilSCAR-C, SybilSCAR-D, and SybilBelief are more robust to label noise than SybilRank. When there are label noises, some benign nodes are treated as Sybils. The edges between these nodes and the rest of benign nodes become attack edges, while the original attack edges that connect with these nodes become edges in the new Sybil region. Similarly, some Sybils are mislabeled as benign nodes. The edges between these nodes and the rest of Sybils are treated as attack edges, while the original attack edges that connect with these nodes become edges in the new benign region. Since we randomly sample mislabeled nodes, the new attack edges are likely to be more than the original attack edges that become edges within the new benign region or Sybil region. Therefore, SybilSCAR-C, SybilSCAR-D, and SybilBelief outperform SybilRank, because they can tolerate a larger number of attack edges. Moreover, when label noise is larger than a certain threshold (this threshold is graph-dependent), the new attack edges are more than what SybilSCAR-C, SybilSCAR-D, and SybilBelief can tolerate, and thus their performance degrades significantly.

3.5.6 Scalability

We evaluate scalability in terms of the peak memory and time used by each method. As evaluating scalability requires social networks with varying number of edges, we choose synthesized graphs with different number of edges. Note that our purpose here is not to concern about the accuracy, which depends on the number of iterations of each method. Thus, to avoid the bias introduced by the number of iterations, we run all methods with the same number of iterations.



(a) Peak memory usage (b) Time

Figure 3.6 Space and time efficiency of SybilRank, SybilBelief, and SybilSCAR-C (SybilSCAR-D has the same space and time efficiency with SybilSCAR-C), vs. number of edges. SybilSCAR-C and SybilRank have almost the same space and time efficiency, while SybilSCAR-C is several times more space efficient and one order of magnitude more time efficient than SybilBelief.

Figure 3.6 exhibits the peak memory and time used by SybilRank, SybilBelief, and SybilSCAR-C (SybilSCAR-D has the same complexity with SybilSCAR-C, and thus we omit its results for simplicity) for different number of edges with 20 iterations. We observe that: 1) all methods have linear space and time complexity, which is consistent with our theoretical analysis in Section 3.4.3; 2) SybilRank and SybilSCAR-C use almost the same space and time; 3) SybilSCAR-C requires a few times less memory than SybilBelief and is one order of magnitude faster than SybilBelief. The reason is that SybilBelief needs a large amount of resources to store and maintain the neighbor influence on every edge. We note that we optimized the implementation of SybilBelief provided by its authors, and our optimized version is one order of magnitude faster than the unoptimized version. Using the unoptimized implementation, SybilSCAR is around two orders of magnitude faster than SybilBelief, which was reported in Wang et al. (2017b).

3.5.7 Convergence

We define a relative error of residual posterior probability vectors of SybilSCAR-C (or SybilSCAR-D) as $\frac{\|\hat{\mathbf{p}}^{(t)}-\hat{\mathbf{p}}^{(t-1)}\|_1}{\|\hat{\mathbf{p}}^{(t)}\|_1}$, where $\hat{\mathbf{p}}^{(t)}$ is the residual vector of posterior probability produced by SybilSCAR-

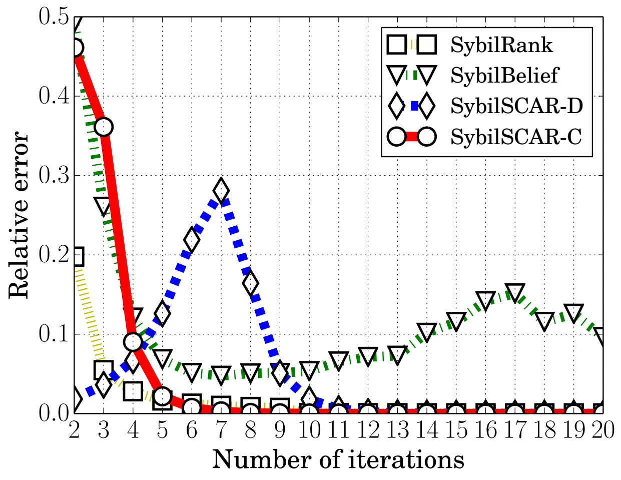
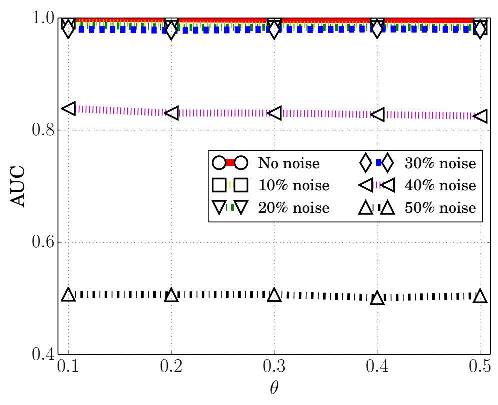


Figure 3.7 Relative errors of SybilRank, SybilBelief, SybilSCAR-D, and SybilSCAR-C vs. the number of iterations on Facebook with 1,000 attack edges. SybilRank, SybilSCAR-D, and SybilSCAR-C can converge, but SybilBelief cannot.

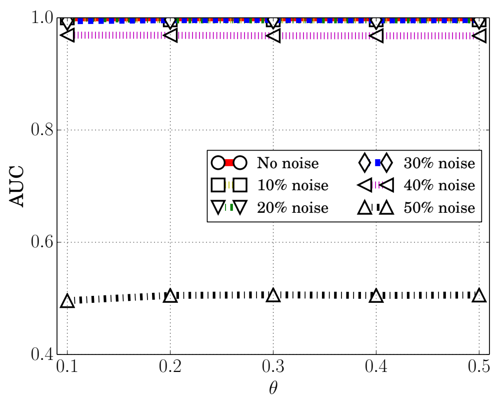
C (or SybilSCAR-D) in the tth iteration. Similarly, we can define relative errors for SybilRank and SybilBelief using their vectors of posterior reputation/probability. Figure 3.7 shows the relative errors vs. the number of iterations on Facebook with 1,000 attack edges. We observe that 1) SybilSCAR-C, SybilSCAR-D, and SybilRank converge after several iterations; and 2) the relative errors of SybilBelief oscillate. SybilBelief does not converge because there exists many loops in real-world social networks and LBP may oscillate on graphs with loops, as pointed out by the author of LBP Pearl (1988).

3.5.8 Impact of the Parameter θ

The parameter θ is the residual prior probability of labeled nodes. Figure 3.8 shows the AUCs of SybilSCAR-D and SybilSCAR-C for different θ and different levels of label noise, where the dataset is Facebook. Note that θ should be in the range (0, 0.5]. Therefore, we explored the values 0.1, 0.2, 0.3, 0.4, and 0.5. We observe that both SybilSCAR-D and SybilSCAR-C are stable with respect to the choice of θ.



(a) SybilSCAR-D



(b) SybilSCAR-C

Figure 3.8 AUCs of (a) SybilSCAR-D and (b) SybilSCAR-C for different levels of label noise and different values for the parameter θ.

3.6 Summary

In this chapter, we first design a novel local rule based on the local rule-based framework in Chapter 2. Our local rule integrates advantages of existing RW-based and LBP-based methods, while overcoming their limitations.

Then, we perform both theoretical and empirical evaluations. Theoretically, SybilSCAR has a tighter asymptotical bound on the number of Sybils that are falsely accepted into the social network than existing collective classification methods. Moreover, SybilSCAR can guarantee to converge. Empirically, our experimental results on both synthesized Sybils and real-world Sybils demonstrate that SybilSCAR is more accurate and more robust to label noise than SybilRank, while SybilSCAR is more accurate and significantly more scalable than SybilBelief.

CHAPTER 4. COLLECTIVE CLASSIFICATION ON DIRECTED GRAPHS FOR SYBIL DETECTION IN SOCIAL NETWORKS

4.1 Introduction

Many methods Yu et al. (2006, 2008); Danezis and Mittal (2009); Viswanath et al. (2010); Mohaisen et al. (2011); Cao et al. (2012); Boshmaf et al. (2015); Liu et al. (2016); Gong et al. (2014a) assume undirected graphs, in which edges represent strong trust relationships between users. However, many social networks are intrinsically directed. For instance, on Twitter, one user can follow another user without being followed back. In a directed social network, we call an edge *unidirectional* if the edge in the reverse direction does not exist, otherwise we call the edge *bidirectional*.

To apply these undirected graph based methods, we need to transform a directed social network to an undirected one, which oversimplifies the social structure, resulting in limited performance. Specifically, there could be two ways to transform a directed social network to an undirected one. One way is to keep an undirected edge between two users once they are connected by directed edge(s). Sybils often aggressively link to many benign users Ghosh et al. (2012); Yang et al. (2012), which well embeds Sybils among benign users and evades detection in this transformation. Indeed, in our Twitter dataset, Sybils proactively follow a large number of benign users while only a small fraction of them follow back. The other way is to keep an undirected edge between two users only if they are connected via bidirectional edges. However, such transformation cannot leverage unidirectional edges. These unidirectional edges could be very informative for a user's reputation. For instance, if many benign users follow a user on Twitter, then the user might also be benign even if he/she does not follow back to those followers. To the best of our knowledge, collective classification methods Gyöngyi et al. (2004); Yang et al. (2012) using directed graphs are all based on *random walks*. They suffer from three limitations. First, they only leverage users' either incoming edges or outgoing edges, but not both. Second, they only leverage manually labeled benign users Gyöngyi et al. (2004) or manually labeled Sybils Yang et al. (2012) (but not both) when a training dataset is available. Third, they are not robust to noisy labels. A training dataset often has noisy labels due to human mistakes Wang et al. (2013). These limitations make them insufficient for real-world social networks.

Our work: We propose GANG, a new collective classification method using directed graphs, to detect Sybils in social networks. 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 Sybil 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 Sybil or v is benign, 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 Sybil can follow arbitrary users without being followed back, while a benign 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) Pearl (1988) 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 each edge, and 2) it is not guaranteed to converge because LBP might oscillate on loopy graphs Pearl (1988). 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 directed 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 Sybils and benign 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 Sybils 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 Sybils in social networks via guilt-by-association on directed graphs. GANG leverages a novel pMRF that captures the unique characteristics of the Sybil 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 Sybils and labeled benign users. Our results demonstrate that GANG significantly outperforms existing methods, and that the optimized GANG is significantly more efficient than its basic version.

4.2 Problem Definition

4.2.1 Directed Graph-based Sybil Detection

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 *Sybil* or *benign*. Sybils include spammers, fake users, and compromised users.

Definition 8 (Directed Graph-based Sybil Detection)**.** *Suppose we are given a directed social graph and a training dataset consisting of labeled Sybils and labeled benign nodes. Directed Graph-based Sybil detection is to predict the label of each remaining node in the social graph.*

4.2.2 Design Goals

1) Leveraging unidirectional edges: A large portion of edges in a directed social graph are unidirectional. For instance, in our large-scale Twitter and Weibo social graphs, 36.2% and 25.4% of edges are unidirectional, respectively. Our method should be able to leverage unidirectional edges to enhance ranking performance.

2) Incorporating both labeled benign users and Sybils: When a manually labeled training dataset is available, our method should be able to incorporate both labeled benign users and labeled Sybils.

3) Robust to label noise: A given label of a user is noisy if it does not match the user's true label. A training dataset might have noisy labels due to human mistakes Wang et al. (2013) or labeled benign users being compromised. We aim to be robust to a minority fraction of given labels being noisy.

4) Scalable: Our method should be scalable to real-world social networks that have hundreds of millions of users and billions of edges.

5) Convergent: If the method is an iterative algorithm, then the iterative process should be convergent, making it easy to determine when to stop.

Existing methods only satisfy a subset of these design goals. SybilGuard Yu et al. (2006), SybilLimit Yu et al. (2008), and SybilInfer Danezis and Mittal (2009) do not satisfy goals 1)-4). SybilRank Cao et al. (2012) and Íntegro Boshmaf et al. (2015) do not satisfy goals 1)-3). SybilBelief Gong et al. (2014a) does not satisfy goals 1), 4), and 5). CIA Yang et al. (2012) and TrustRank Gyöngyi et al. (2004) do not satisfy goals 2) and 3).

4.3 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 pMRF to capture the intuitions. Third, we discuss how we leverage the pMRF to detect Sybils.

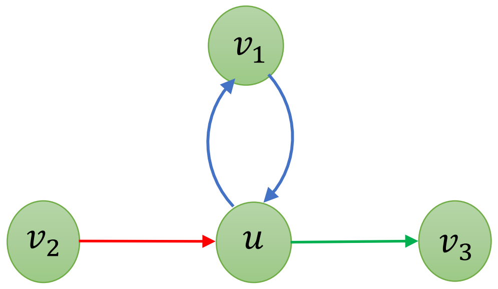


Figure 4.1 Illustration of three types of neighbors. v_1, v_2, and v_3 are *bidirectional*, *unidirectional incoming*, and *unidirectional outgoing* neighbors of u, respectively.

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. 4.1 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.

4.3.1 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 *Sybil* and *benign*, 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 Sybil. This property is known as *homophily.* social networks with Sybil and benign nodes have the homophily property because benign nodes will not link to Sybils with bidirectional edges in most cases. Given the labels of u's bidirectional neighbors, we model the probability that u is Sybil as the following *sigmoid function*:

$$Pr(x_u = 1|\bar{x}_{\Gamma_b(u)}) = \frac{1}{1 + \exp(-\sum_{v \in \Gamma_b(u)} J_{vu}\bar{x}_v)}, \tag{4.1}$$

  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 Sybil if more of its bidirectional neighbors are Sybils. 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 Sybil. This is because a Sybil can link to many other nodes (Sybil or benign) in social networks. 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 Sybil could follow many other Sybil or benign users. Therefore, being linked by a Sybil does not mean the node is Sybil nor benign. However, when v is benign, u also tends to be benign. We model this intuition as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma_i(u)}) = \frac{1}{1 + \exp(-\frac{1}{2}\sum_{v \in \Gamma_i(u)} J_{vu}(\bar{x}_v - 1))}, \tag{4.2}$$

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

- **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 benign node. This is because any node can link to a benign node in social networks. For instance, in the Twitter example, any node can follow a benign user. However, if v is Sybil, then u also tends to be Sybil because a benign user rarely follows a Sybil. We model this intuition as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma_o(u)}) = \frac{1}{1 + \exp(-\frac{1}{2}\sum_{v\in\Gamma_o(u)} J_{uv}(\bar{x}_v + 1))}, \tag{4.3}$$

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

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:

$$Pr(x_u = 1) = \frac{1}{1 + \exp(-h_u)}, \tag{4.4}$$

where $h_u > 0$ and $h_u < 0$ indicate that u tends to be Sybil or benign 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 4.3.3). 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 Sybil as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma(u)}) = \frac{1}{1 + \exp(-I_b(u) - I_i(u) - I_o(u) - h_u)}, \tag{4.5}$$

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;
- h_u models the prior knowledge about u.

4.3.2 A Novel Pairwise Markov Random Field

We introduce a novel pairwise Markov Random Field (pMRF) to capture our intuitions in Equation 4.5. 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:

$$H(x_V) = -\frac{1}{2}\sum_{(u,v)\in E_2} J_{uv}x_u x_v - \frac{1}{4}\sum_{(u,v)\in E_1} J_{uv}(x_u - 1)(x_v + 1) - \frac{1}{2}\sum_{u\in V} h_u x_u, \tag{4.6}$$

$$Pr(x_V) \propto \exp(-H(x_V)) \tag{4.7}$$

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 4.5. In particular, when u's neighbors' states are observed, the conditional probability that u is Sybil is given by Equation 4.5. Note that an unidirectional edge (u, v) does not influence the value of our energy function nor the joint probability if u is a Sybil or v is a benign node. This is because of our Intuition II and III.

Next, we will transform Equation 4.7 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 Sybil. 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 4.7 as follows:

$$Pr(x_V) = \frac{1}{Z} \prod_{v \in V} \phi_v(x_v) \prod_{(u,v) \in E_1 \cup E_2} \varphi_{uv}(x_u, x_v), \tag{4.8}$$

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 benignizes 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.

4.3.3 Detecting Sybils

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

$$q_u = \begin{cases} 0.5 + \theta & \text{if } u \in L_f \\ 0.5 - \theta & \text{if } u \in L_n \\ 0.5 & \text{otherwise,} \end{cases} \tag{4.9}$$

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 Sybil, i.e., $p_u = Pr(x_u = 1)$. We predict u to be Sybil if $p_u > 0.5$, otherwise we predict u to be benign.

Computing posterior probability distribution using Loopy Belief Propagation (LBP): In the basic version of GANG, we use LBP Pearl (1988) 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

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v, x_u) \prod_{k\in\Gamma(v)/u} m_{kv}^{(t-1)}(x_v), \tag{4.10}$$

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:

$$Pr^{(t)}(x_u) \propto \phi_u(x_u) \prod_{k\in\Gamma(u)} m_{ku}^{(t)}(x_u), \tag{4.11}$$

which is equivalent to

$$p_u^{(t)} = \frac{q_u \prod_{k\in\Gamma(u)} m_{ku}^{(t)}}{q_u \prod_{k\in\Gamma(u)} m_{ku}^{(t)} + (1-q_u)\prod_{k\in\Gamma(u)}(1-m_{ku}^{(t)})}, \tag{4.12}$$

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

4.4 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 Pearl (1988). In this section, we optimize GANG to address these shortcomings.

4.4.1 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 4.10 as follows:

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v, x_u) \prod_{k \in \Gamma(v)} m_{kv}^{(t-1)}(x_v). \tag{4.13}$$

Considering Equation 4.11, we have:

$$m_{vu}^{(t)}(x_u) \propto \sum_{x_v} \varphi_{vu}(x_v, x_u) Pr^{(t-1)}(x_v). \tag{4.14}$$

Recall that we benignize $m_{vu}^{(t)}(x_u)$ 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 benignization, our modified messages become

$$m_{vu}^{(t)} = \begin{cases} 0.5, & \text{if } p_v^{(t-1)} > 0.5 \text{ and } v \in \Gamma_i(u) \\ 0.5, & \text{if } p_v^{(t-1)} < 0.5 \text{ and } v \in \Gamma_o(u) \\ p_v^{(t-1)} w + (1 - p_v^{(t-1)})(1 - w), & \text{otherwise.} \end{cases} \tag{4.15}$$

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.

4.4.2 Approximating GANG with a Matrix Form

GANG iteratively applies Equations 4.15 and 4.12 with our modified messages, which still cannot guarantee convergence. The key reason is that Equation 4.12 combines messages from a node's neighbors in a nonlinear fashion. We make GANG converge via linearizing Equation 4.12. 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 9.

Lemma 9 (Residual Messages)**.** *The residual message $\hat{m}_{vu}^{(t)}$ can be represented as follows:*

$$\hat{m}_{vu}^{(t)} = \begin{cases} 0 & \text{if } \hat{p}_v^{(t-1)} > 0 \text{ and } v \in \Gamma_i(u) \\ 0 & \text{if } \hat{p}_v^{(t-1)} < 0 \text{ and } v \in \Gamma_o(u) \\ 2\hat{p}_v^{(t-1)}\hat{w} & \text{otherwise.} \end{cases} \tag{4.16}$$

Proof. By substituting variables in Equation 4.15 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 10. *We can approximate Equations 4.15 and 4.12 as the following equation:*

$$\begin{cases} \mathbf{A}_i'^{(t-1)} = I(\mathbf{A}_i \circ \hat{\mathbf{P}}^{(t-1)^T}), \\ \mathbf{A}_o'^{(t-1)} = I(-\mathbf{A}_o \circ \hat{\mathbf{P}}^{(t-1)^T}), \\ \hat{\mathbf{p}}^{(t)} = \hat{\mathbf{q}} + 2 \cdot \hat{w} \cdot (\mathbf{A}_b + \mathbf{A}_i'^{(t-1)} + \mathbf{A}_o'^{(t-1)}) \cdot \hat{\mathbf{p}}^{(t-1)}, \end{cases} \tag{4.17}$$

where the operator $\circ$ represents element-wise product of two matrices; $\mathbf{Y}^T$ is the transpose of the matrix $\mathbf{Y}$; 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 . □

Theorem 10 demonstrates that posterior beliefs can be iteratively solved by matrix operations without explicitly modeling messages.

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 scalabilities.

4.5 Convergence Analysis

We analyze the conditions when our optimized 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 Saad (2003) 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 11 (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.*

$$\hat{w} < \frac{1}{2\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1} = \frac{1}{2\max_{u \in V} d_u}, \tag{4.18}$$

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

Proof. See Appendix . □

Theorem 11 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.

Table 4.1 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

4.6 Evaluations

4.6.1 Experimental Setup

Dataset description: We compare GANG with existing methods on two large-scale social graph datasets with labeled Sybil and benign nodes.

First, we obtained a Twitter follower-followee graph with 41,652,230 users and 1,468,364,884 edges from Kwak et al. Kwak et al. (2010). 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. Wang et al. (2017b). Specifically, 205,355 users were suspended by Twitter and we treated them as Sybils; 36,156,909 users were active and we treated them as benign 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. Fu et al. (2017). Like 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 Sybils, 1,498 were benign users, and 20 were unknown users (we do not consider these users in our experiments). We split the Sybil and benign users into two halves; one is treated as the training set and the other is treated as the testing set. Table 4.1 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) Zhu et al. (2003), SybilRank Cao et al. (2012), SybilBelief Gong et al. (2014a), and SybilSCAR Wang et al. (2017b). 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 Sybils and benign users in the training dataset, while SybilRank is only able to leverage labeled benign 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 Sybils can create arbitrary number of unidirectional edges with benign nodes, making them well embedded among benign nodes. Since these methods require connected graphs, we evaluate them on the largest connected component in the transformed undirected graph.

2) Using directed graphs. We consider the following directed graph based methods: TrustRank Gyöngyi et al. (2004), DistrustRank Wu et al. (2006), CIA Yang et al. (2012), and CatchSync Jiang et al. (2014). TrustRank, DistrustRank, and CIA are based on random walks, while CatchSync leverages HITS Kleinberg (1999). TrustRank and DistrustRank were originally designed to detect Sybil webpages based on hyperlinks, but they can be applied to detect Sybils in social networks. TrustRank leverages only labeled benign nodes in the training dataset; DistrustRank and CIA are essentially the same, and they only leverage labeled Sybils; and CatchSync does not leverage the training dataset.

Parameter setting: For GANG, we set $\theta = 0.4$ to consider possible label noises, i.e., we assign a prior probability of being Sybil 0.9, 0.1, and 0.5 to labeled Sybils, labeled benign 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 com-

Table 4.2 AUCs of compared methods.

Methods		Twitter	Sina Weibo
Using undirected graphs	SSL Zhu et al. (2003)	0.55	0.68
	SybilRank Cao et al. (2012)	0.57	0.61
	SybilBelief Gong et al. (2014a)	0.61	0.65
	SybilSCAR Wang et al. (2017b)	0.64	0.68
Using directed graphs	TrustRank Gyöngyi et al. (2004)	0.60	0.66
	DistrustRank Wu et al. (2006)	0.63	0.64
	CIA Yang et al. (2012)	0.63	0.64
	CatchSync Jiang et al. (2014)	0.68	0.51
	GANG	**0.72**	**0.80**

pared methods, we set their parameters according to the original papers. For instance, we set the *decay factor* to be 0.85 for TrustRank Gyöngyi et al. (2004) as suggested by its authors. For CatchSync Jiang et al. (2014), we set the parameter $\alpha = 3$.

4.6.2 Ranking Results

Each compared method computes a score for each node. We rank the nodes in the testing dataset using the scores such that Sybils are supposed to rank higher than benign 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 Sybil is ranked higher than a randomly sampled benign node in the testing dataset. The higher AUC, the better performance. Table 4.2 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.

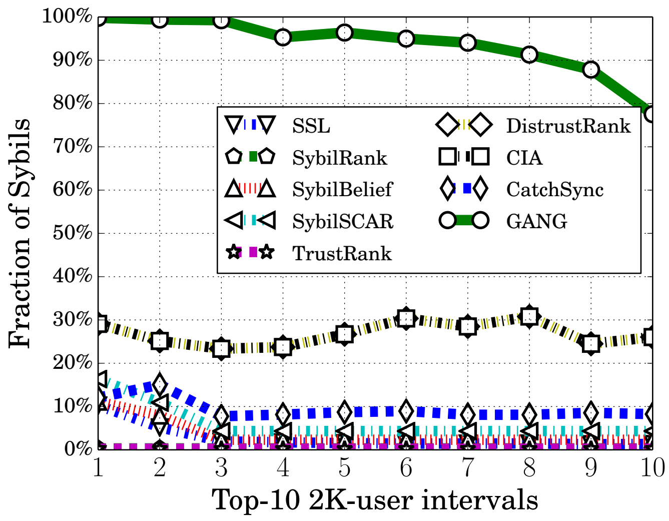
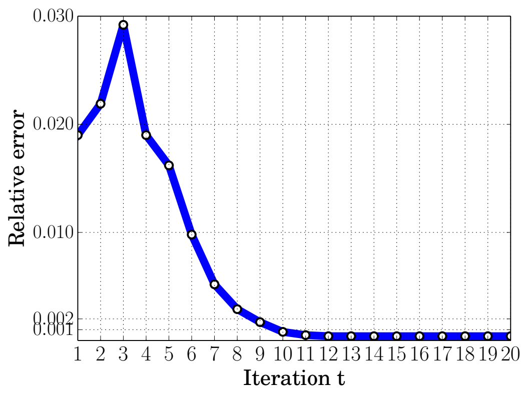


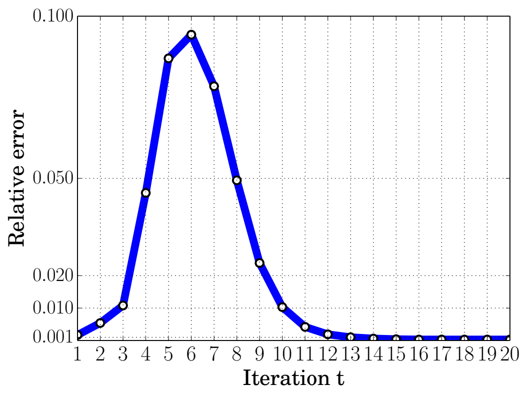
Figure 4.2 Fraction of Sybils in each top ranked interval on Twitter.

Sybils in top-ranked nodes: In practice, the ranking of nodes can be used as a priority list to help social networks' human workers manually inspect nodes and detect Sybils. Inspecting nodes according to their rankings could aid human workers to detect more Sybils than inspecting nodes picked uniformly at random, within the same amount of time. When ranking is used for such purpose, the number of Sybils 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 Sybils among the top-ranked nodes. Therefore, we further compare the considered methods using the fraction of Sybils in top-ranked nodes. In particular, we divide the top-20K nodes into 10 intervals, where each interval has 2K nodes. Figure 4.2 shows the fraction of Sybils 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 Sybils 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 Sybil and benign users.



(a) Twitter



(b) Sina Weibo

Figure 4.3 GANG's relative errors of residual posterior beliefs vs. number of iterations. GANG converges.

4.6.3 Convergence

We evaluate the convergence of our proposed GANG by observing the residual posterior beliefs in two consecutive iterations. Figure 4.3 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.

4.6.4 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 Barabási and Albert (1999). We note that there are more advanced network models (e.g., the one proposed by Gong et al. Gong et al. (2012)) 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,

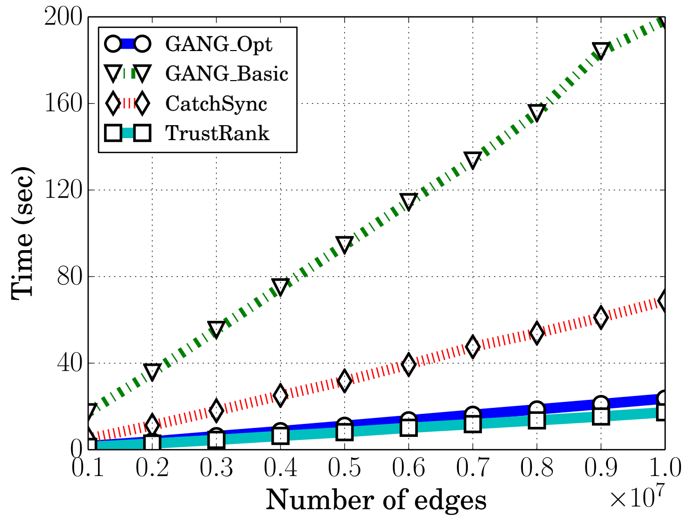


Figure 4.4 Running time of directed graph based methods on synthesized graphs with increasing number of edges. DistrustRank and CIA have almost identical results with TrustRank, and thus we omit their results for conciseness.

while CatchSync relies on the iterative HITS Kleinberg (1999) algorithm. For fair comparison, we run the iterative processes with the same number of iterations. Figure 4.4 shows the running time used by the directed graph based methods (GANG_Basic 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 *benignity*, 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.

Table 4.3 Labeling results of the 1K nodes that are sampled from the top-ranked 100K nodes for Sina Weibo.

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

4.6.5 Case Study on Sina Weibo

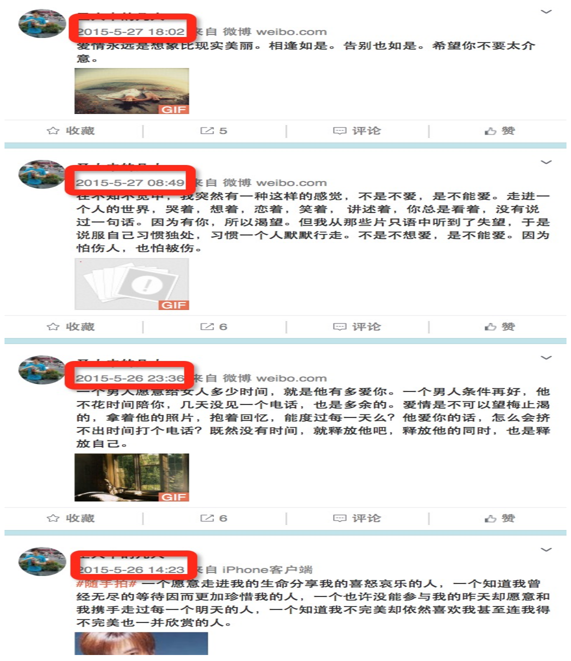
We apply our GANG to the Sina Weibo dataset and manually inspect the detected Sybils. Specifically, we use all the 1980 labeled nodes as a training dataset and produce a ranking list for the remaining nodes. Then we sample 1K nodes from the top-ranked 100K nodes uniformly at random and manually inspect them. Table 4.3 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 benign tweets* to camouflage themselves. For instance, Figure 4.5(a) shows an example spammer who posts seemingly benign 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 4.5(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 benign tweets about daily activities before a certain time point, and then they started to post or share a large amount of advertisements. We also

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



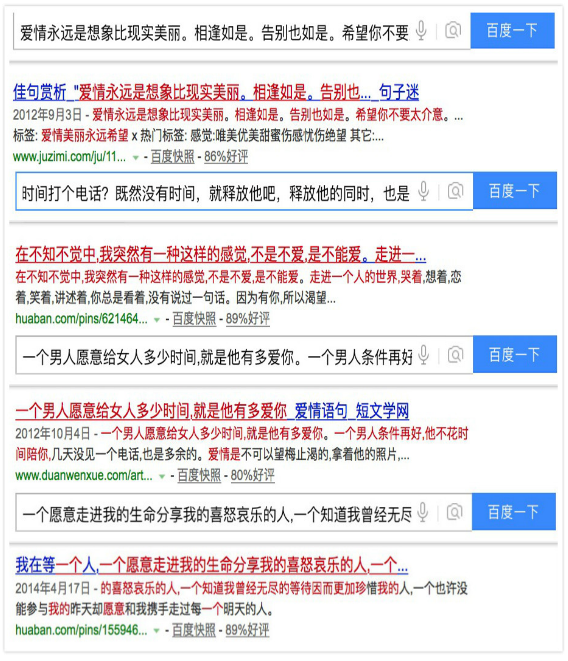
2015-5-27 18:02 来自 微博 weibo.com
爱情永远是想象比现实美丽。相逢如是。告别也如是。希望你不要太介意。

2015-5-27 08:49 来自 微博 weibo.com
在不知不觉中，我突然有一种这样的感觉，不是不爱，是不能爱。走进一个人的世界，哭着，想着，恋着，笑着， 讲述着，你总是看着，没有说过一句话。因为有你，所以渴望。但我从那些片只语中听到了失望，于是说服自己习惯独处，习惯一个人默默行走。不是不想爱，是不能爱。因为怕伤人，也怕被伤。

2015-5-26 23:36 来自 微博 weibo.com
一个男人愿意给女人多少时间，就是他有多爱你。一个男人条件再好，他不花时间陪你，几天没见一个电话，也是多余的。爱情是不可以望梅止渴的，拿着他的照片，抱着回忆，能度过每一天么？他爱你的话，怎么会挤不出时间打个电话？既然没有时间，就释放他吧，释放他的同时，也是释放自己。

2015-5-26 14:23 来自 iPhone客户端
#随手拍# 一个愿意走进我的生命分享我的喜怒哀乐的人，一个知道我曾经无尽的等待因而更加珍惜我的人，一个也许没能参与我的昨天却愿意和我携手走过每一个明天的人，一个知道我不完美却依然喜欢我甚至连我得不完美也一并欣赏的人。

(a) Periodic tweeting



爱情永远是想象比现实美丽。相逢如是。告别也如是。希望你不要

佳句赏析_"爱情永远是想象比现实美丽。相逢如是。告别也..._句子迷
2012年9月3日 - 爱情永远是想象比现实美丽。相逢如是。告别也如是。希望你不要太介意。...
标签: 爱情美丽永远希望 x 热门标签: 感觉:唯美优美甜蜜伤感忧伤绝望 其它:...
www.juzimi.com/ju/11... - 百度快照 - 86%好评

时间打个电话？既然没有时间，就释放他吧，释放他的同时，也是

在不知不觉中,我突然有一种这样的感觉,不是不爱,是不能爱。走进一...
在不知不觉中,我突然有一种这样的感觉,不是不爱,是不能爱。走进一个人的世界,哭着,想着,恋着,笑着,讲述着,你总是看着,没有说过一句话。因为有你,所以渴望...
huaban.com/pins/621464... - 百度快照 - 89%好评

一个男人愿意给女人多少时间,就是他有多爱你。一个男人条件再好

一个男人愿意给女人多少时间,就是他有多爱你_爱情语句_短文学网
2012年10月4日 - 一个男人愿意给女人多少时间,就是他有多爱你。一个男人条件再好,他不花时间陪你,几天没见一个电话,也是多余的。爱情是不可以望梅止渴的,拿着他的照片,...
www.duanwenxue.com/art... - 百度快照 - 80%好评

一个愿意走进我的生命分享我的喜怒哀乐的人,一个知道我曾经无尽

我在等一个人,一个愿意走进我的生命分享我的喜怒哀乐的人,一个...
2014年4月17日 - 的喜怒哀乐的人,一个知道我曾经无尽的等待因而更加珍惜我的人,一个也许没能参与我的昨天却愿意和我携手走过每一个明天的人。
huaban.com/pins/155946... - 百度快照 - 89%好评

(b) Search results

Figure 4.5 (a) A user performing periodic tweeting. (b) Search results of one of the user's tweet on Baidu.

randomly sampled some benign tweets of these users, but we could not find them on the Baidu search engine. Therefore, we suspect that these users could be compromised benign users. Our method can detect these compromised users because they link to other spammers to share their tweets.

4) Normal users. These users post benign 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 Sybil, the user will be suspended or deleted. In other words, the number of Sybils 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 Sybils.

4.7 Summary

This chapter has introduced GANG, a guilt-by-association method on directed graphs, to detect Sybils in directed social networks. Based on the unique characteristics of the Sybil 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 LBP 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 Sybils and benign 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.

CHAPTER 5. COLLECTIVE CLASSIFICATION WITH JOINT WEIGHT LEARNING AND PROPAGATION

5.1 Introduction

State-of-the-art collective classification methods have three key steps. In Step I, they assign a *prior reputation score* to every node in the graph based on the training dataset. In Step II, they assign weights to edges in the graph, where a large weight of an edge (u, v) indicates that u and v are likely to have the same label. In Step III, they iteratively propagate the prior reputation scores among the weighted graph to obtain a *posterior reputation score* for every node. The posterior reputation scores are used to classify or rank nodes. The details of one or more of the three steps could be different for different collective classification methods. For instance, some methods leverage random walks for propagation in Step III, while some methods leverage loopy belief propagation in Step III.

An edge (u, v) is said to be *homogeneous* if u and v have the same label, otherwise it is said to be *heterogeneous*. The propagation in Step III requires that homogeneous edges have large weights and heterogeneous edges have small weights. However, existing methods face a key limitation: they assign small weights to a large number of homogeneous edges and/or large weights to a large number of heterogeneous edges in Step II.

Specifically, most existing methods Pandit et al. (2007); Chau et al. (2011); Ye et al. (2011); Cao et al. (2012); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015); Wang et al. (2017b, 2018a); Jia et al. (2017a); Wang et al. (2017a) simply set a large constant weight to all edges, ignoring the difference between homogeneous and heterogeneous edges. A few methods Boshmaf et al. (2015); Gao et al. (2018) proposed to learn edge weights using features extracted from nodes' characteristics and the graph structure. For instance, to detect Sybils in social networks, Íntegro Boshmaf et al. (2015) first learns a classifier based on users' profiles to predict the probability that a node is a victim (a victim is a benign user connecting to at least one Sybil) and then assigns small weights to all edges of the nodes that are victims with high probabilities. However, Íntegro assigns small weights to a large number of homogeneous edges, because a large number of nodes are victims Gao et al. (2018). SybilFuse Gao et al. (2018) extracts features from the graph structure for each node, learns a classifier to predict the prior reputation score for each node using the training dataset, and uses the prior reputation scores to assign edge weights. In particular, an edge has a larger weight if the two nodes of the edge are more likely to have the same label, where a node's label is determined by its prior reputation score. Since the prior reputation scores are inaccurate at determining nodes' labels (otherwise we do not need to propagate the prior reputation scores to obtain posterior reputation scores in Step III), a large number of heterogeneous edges are assigned large weights while a large number of homogeneous edges are assigned small weights.

As a result, existing methods have a limited success in the security and privacy problems that have a large amount of heterogeneous edges, e.g., Sybil detection in weak-trust social networks like Twitter, fake review detection, and attribute inference.

Our work: We propose a new framework to learn edge weights for graph-based security and privacy analytics. Our framework is applicable to both RW-based and LBP-based methods. Our key intuition is that the edge weights and the final posterior reputation scores produced by a collective classification method should satisfy two goals. First, the labeled positive nodes and the labeled negative nodes in the training dataset should have high and low posterior reputation scores, respectively. Second, the edge weights and the posterior reputation scores should be *consistent*. Specifically, we use the final posterior reputation scores to predict labels of nodes; an edge is predicted to be homogeneous if the two nodes of the edge are predicted to have the same label, otherwise the edge is predicted to be heterogeneous. Consistency between edge weights and posterior reputation scores means that an edge that is predicted to be homogeneous should have a large weight, while an edge that is predicted to be heterogeneous should have a small weight.

We formulate learning edge weights as minimizing an *objective function*, where the objective function is small if the two goals are achieved. However, it is computationally challenging to solve the formulated optimization problem (we will discuss more details in Section 5.3.2) because the final posterior reputation scores depend on the edge weights in a very complex way, e.g., every edge weight could influence the final posterior reputation score of every node.

To address the computational challenge, we propose to jointly learn the edge weights and propagate the posterior reputation scores, which can be viewed as an approximate solution to our formulated optimization problem. Our key idea is that the posterior reputation scores are iteratively updated in Step III and thus we iteratively learn edge weights using the current posterior reputation scores instead of the final posterior reputation scores. Specifically, given the posterior reputation scores in the tth iteration, we learn new edge weights and then use the learnt edge weights to update the posterior reputation scores in the $(t+1)$th iteration. We aim to learn the edge weights to satisfy the two goals: 1) the posterior reputation scores in the $(t+1)$th iteration should be large and small for the labeled positive nodes and negative nodes in the training dataset, respectively; and 2) the learnt edge weights should be consistent with the posterior reputation scores in the tth iteration. Learning edge weights is efficient under our framework because the posterior reputation scores of the nodes in the training dataset in the $(t+1)$th iteration only depend on weights of the edges of the nodes in the training dataset.

We compare our framework with state-of-the-art RW-based and LBP-based methods using multiple large-scale real-world datasets, which are from different application scenarios including Sybil detection in social networks, fake review detection, and attribute inference attacks. For instance, the Twitter dataset for Sybil detection has 42M users and 1.5B edges, while the Google+ dataset for attribute inference attacks has 5M users and 31M edges. Our results demonstrate that our framework has significantly higher accuracies than state-of-the-art methods, with an acceptable computational overhead (around 2-3 times slower than state-of-the-art methods). Moreover, we apply our framework to detect Sybils in a large-scale directed Sina Weibo (a large social network in China) dataset with 3.5M users and 653M edges. We manually verify the detected Sybils and our results show that our framework can accurately detect Sybils, e.g., 95% of the top-ranked 100K users are Sybils.

We summarize our contributions as follows:

- We propose a novel framework to learn edge weights for graph-based security and privacy analytics.
- We formulate learning edge weights as solving an optimization problem. Moreover, we design efficient algorithms to solve the optimization problem.
- We compare our framework with state-of-the-art methods using multiple large-scale real-world datasets from different security and privacy application scenarios.

5.2 Problem Definition

5.2.1 Collective Classification

Suppose we are given a graph (either undirected or directed) $G = (V, E)$, where $u \in V$ is a node and $(u, v) \in E$ is an edge, which indicates a certain relationship between u and v. $|V|$ and $|E|$ are the number of nodes and edges, respectively. We are also given a training dataset L, which consists of a set of labeled positive nodes L_P and a set of labeled negative nodes L_N. We denote by y_u the label of node u, where $y_u = 1$ means that u is positive and $y_u = -1$ means that u is negative. We formally define the collective classification problem as follows:

Definition 12 (Collective Classification Problem)**.** *Suppose we are given 1) a graph (either undirected or directed) and 2) a training dataset including both labeled positive nodes and labeled negative nodes. Collective classification is to classify or rank the unlabeled nodes in the graph.*

5.2.2 Evaluation Metrics

Like previous studies Viswanath et al. (2010); Mohaisen et al. (2011); Chau et al. (2011); Cao et al. (2012); Boshmaf et al. (2015); Gong et al. (2014a); Tamersoy et al. (2014); Wang et al. (2017b,a), we adopt the following metrics to evaluate a collective classification method for graph-based security and privacy analytics.

Accuracy: One way to measure the accuracy of a collective classification method is to use its posterior reputation scores to classify the unlabeled nodes, i.e., a node is predicted to be positive if its posterior reputation score is bigger than a certain *threshold*, and negative otherwise. Then, accuracy is the fraction of unlabeled nodes whose labels are correctly predicted. However, such accuracy highly relies on the threshold. Moreover, it is challenging to select a good threshold for various collective classification methods. Therefore, *Area Under the Receiver Operating Characteristic Curve (AUC)* is widely used to measure the accuracy of a collective classification method. AUC does not depend on a threshold and can be consistently used to evaluate all collective classification methods that produce posterior reputation scores. In particular, we rank all unlabeled nodes in a decreasing order with respect to their posterior reputation scores. AUC is the probability that a randomly selected positive node ranks higher than a randomly selected negative node. If a method ranks all positive nodes before negative nodes, then the method has an AUC of 1 (there exists a threshold such that all nodes are correctly classified); if a method ranks all negative nodes before positive nodes, then the method has an AUC of 0 (there exists a threshold such that all nodes are incorrectly classified); a method has an AUC of 0.5 if the method ranks the unlabeled nodes uniformly at random.

Scalability: In real-world security and privacy problems, the graph often has a very large scale, e.g., hundreds of millions of nodes and edges. Therefore, a collective classification method should be scalable. In particular, we use the computational *time* a method consumes to measure its scalability/efficiency.

Robustness to label noise: The training dataset is often obtained via manual inspection of nodes, or is crowdsourced from users or workers in a crowdsourcing platform such as Amazon Mechanical Turk Wang et al. (2013). For instance, online social networks often provide users an option to report a certain user as a spammer or fraudulent user, which could be incorporated into the training dataset. However, due to human mistakes Wang et al. (2013); Freeman (2017), the training dataset could have noise, i.e., some negative nodes are falsely labeled as positive and/or some positive nodes are falsely labeled as negative. Therefore, we also use robustness to label noise in the training dataset to evaluate collective classification methods. In particular, suppose $\alpha\%$ of the nodes in the training dataset are falsely labeled and a method achieves an AUC of β with such training dataset. Then, we say the method has a robustness of β when the label noise is at the level of $\alpha\%$.

5.3 Our Framework

We first summarize state-of-the-art collective classification methods. In particular, the posterior reputation scores are essentially solutions to a *system of equations*. Then, we formulate learning edge weights as an optimization problem, which quantifies our goals on the posterior reputation scores. However, it is computationally challenging to solve this optimization problem because the posterior reputation scores depend on the edge weights in very complex ways. Finally, we introduce our strategy to alternately learn edge weights and propagate posterior reputation scores. Our strategy is an approximate solution to our formulated optimization problem.

5.3.1 Background

In state-of-the-art collective classification methods Gyöngyi et al. (2004); Wu et al. (2006); Li et al. (2013); Pandit et al. (2007); Chau et al. (2011); Mohaisen et al. (2011); Cao et al. (2012); Yang et al. (2012); Gong et al. (2014a); Tamersoy et al. (2014); Akoglu et al. (2013); Li et al. (2014); Rayana and Akoglu (2015); Wang et al. (2017b); Jia et al. (2017b,a); Wang et al. (2017a); Boshmaf et al. (2015); Gao et al. (2018), the posterior reputation scores are solutions to the following system of equations:

$$\mathbf{p} = f(\mathbf{q}, \mathbf{W}, \mathbf{p}), \tag{5.1}$$

where the column vector $\mathbf{q}$ is the nodes' prior reputation scores, the column vector $\mathbf{p}$ is the nodes' posterior reputation scores, and the matrix $\mathbf{W}$ is the edge weight matrix. Different methods have different prior reputation scores $\mathbf{q}$, assign different edge weight matrix $\mathbf{W}$, and use different function f. Next, we discuss the choices for these parameters in state-of-the-art methods. Since

LBP-based methods outperform RW-based methods Koutra et al. (2011); Jia et al. (2017b); Wang et al. (2017b,a), we will focus on LBP-based methods for simplicity. However, we stress that our framework is also applicable to RW-based methods (see more details in Section 5.5).

LBP-based methods on undirected graphs: These methods Pandit et al. (2007); Chau et al. (2011); Gong et al. (2014a); Tamersoy et al. (2014); Jia et al. (2017b); Wang et al. (2017b); Gao et al. (2018) often assign a positive prior reputation score to a labeled positive node and a negative prior reputation score to a labeled negative node in the training dataset. For example, the prior reputation score q_u of the node u can be assigned as follows:

$$q_u = \begin{cases} \theta & \text{if } u \in L_P \\ -\theta & \text{if } u \in L_N \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where $\theta > 0$ (e.g., $\theta = 1$ in our experiments), L_P is the set of labeled positive nodes, and L_N is the set of labeled negative nodes in the training dataset. The entry w_{uv} of the matrix $\mathbf{W}$ is 0 if u and v are not connected, otherwise w_{uv} and w_{vu} are the weight of the edge (u, v). The edge weight w_{uv} indicates the likelihood that u and v have the same label. Specifically, $w_{uv} > 0$ means that u and v are likely to have the same label, i.e., the edge (u, v) is *homogeneous*; $w_{uv} < 0$ means that u and v are likely to have different labels, i.e., the edge (u, v) is *heterogeneous*; $w_{uv} = 0$ means that u and v are not correlated. Moreover, the function f is as follows:

$$f(\mathbf{q}, \mathbf{W}, \mathbf{p}) = \mathbf{q} + \mathbf{W}\mathbf{p}, \tag{5.3}$$

where we consider the optimized LBP Gatterbauer et al. (2015); Wang et al. (2017b,a); Gatterbauer (2017) instead of the standard LBP Pearl (1988).

LBP-based methods on directed graphs: These methods Wang et al. (2017a) also assign the prior reputation scores in Equation 5.2. Moreover, each connected node pair (u, v) has two entries w_{uv} and w_{vu} in the weight matrix $\mathbf{W}$, where w_{uv} does not necessarily equal w_{vu}. However, they use a different function f as follows:

$$f(\mathbf{q}, \mathbf{W}, \mathbf{p}) = \mathbf{q} + (\mathbf{W} \circ \mathbf{A}_b)\mathbf{p} + (\mathbf{W} \circ \mathbf{A}_i)\mathcal{I}(\mathbf{p}) + (\mathbf{W} \circ \mathbf{A}_o)\mathcal{J}(\mathbf{p}), \tag{5.4}$$

where the operator $\circ$ represents element-wise product of two matrices; the matrix A_b is the adjacency matrix for bidirectional edges, i.e., if both the edge (u, v) and the edge (v, u) exist, then $A_{b,uv} = A_{b,vu} = 1$, otherwise $A_{b,uv} = A_{b,vu} = 0$; A_i is the adjacency matrix for unidirectional incoming edges, i.e., if (v, u) exists but (u, v) does not, then $A_{i,uv} = 1$, otherwise $A_{i,uv} = 0$; A_o is the adjacency matrix for unidirectional outgoing edges, i.e., if (u, v) exists but (v, u) does not, then $A_{o,uv} = 1$, otherwise $A_{o,uv} = 0$; $\mathcal{I}$ and $\mathcal{J}$ are functions that apply to every entry of $\mathbf{p}$, and they reset the positive and negative entries to be 0, respectively.

Suppose v is a neighbor of u. In LBP-based methods, a bidirectional neighbor v multiplies its posterior reputation score by the edge weight and shares it with the node u; an incoming neighbor v does so only if v's posterior reputation score is negative; and an outgoing neighbor v does so only if v's posterior reputation score is positive. The intuition is that an incoming neighbor v influences the node u's label only if v is predicted to be negative, while an outgoing neighbor v influences the node u's label only if v is predicted to be positive. For instance, in Sybil detection on Twitter, if v follows u but u does not follow back, then v is informative for determining u's label only if v is benign, because a Sybil can arbitrarily follow any users; if u follows v but v does not follow back, then v is informative for determining u's label only if v is Sybil, because any users can follow a benign user.

Iteratively solving the posterior reputation scores: The posterior reputation scores in Equation 5.1 are often solved iteratively. Specifically, the posterior reputation scores are initialized to the prior reputation scores. Then, the posterior reputation scores are iteratively updated until convergence as follows:

$$\mathbf{p}^{(t+1)} = f(\mathbf{q}, \mathbf{W}, \mathbf{p}^{(t)}), \tag{5.5}$$

where $\mathbf{p}^{(t)}$ is the posterior reputation scores in the tth iteration. Note that the edge weight matrix $\mathbf{W}$ is fixed during the iterative process. Finally, in LBP-based methods, a node is predicted to be positive if the node's posterior reputation score is positive, otherwise the node is predicted to be negative.

5.3.2 Learning Edge Weights as An Optimization Problem

Our key intuition is that the edge weights and the posterior reputation scores produced by a collective classification method should satisfy the following two goals.

- **Goal 1.** The posterior reputation scores of the labeled positive nodes and the labeled negative nodes in the training dataset should be large and small, respectively.
- **Goal 2.** The edge weights and the posterior reputation scores should be *consistent.* In particular, we use the posterior reputation scores to predict node labels; an edge is predicted to be homogeneous if the two corresponding nodes are predicted to have the same label, otherwise the edge is predicted to be heterogeneous. Consistency means that an edge that is predicted to be homogeneous should have a positive weight, while an edge that is predicted to be heterogeneous should have a negative weight.

Quantifying Goal 1: Given the training dataset L, we quantify the Goal 1 as finding an edge weight matrix $\mathbf{W}$ such that the difference between the posterior reputation scores of the labeled nodes and their labels is minimized. Formally, we aim to find an edge weight matrix $\mathbf{W}$ that *minimizes* the following function:

$$\mathrm{L}(\mathbf{W}) = \frac{1}{2}\sum_{l\in L}(p_l - y_l)^2, \tag{5.6}$$

where p_l is the posterior reputation score of node l and y_l is the label of node l in the training dataset, where $y_l = 1$ if u is positive and $y_l = -1$ otherwise. In machine learning, $\mathrm{L}(\mathbf{W})$ is known as a *loss function* over the training dataset.

Quantifying Goal 2: An edge weight w_{uv} is consistent with the posterior reputation scores of u and v when 1) u and v are predicted to have the same label (i.e., $p_u p_v > 0$) and w_{uv} is positive, or 2) u and v are predicted to have different labels (i.e., $p_u p_v < 0$) and w_{uv} is negative. Therefore, to capture Goal 2, we aim to learn a weight matrix $\mathbf{W}$ that *maximizes* the following function:

$$\mathrm{C}(\mathbf{W}) = \sum_{(u,v)\in E} p_u p_v w_{uv}, \tag{5.7}$$

where $\mathrm{C}(\mathbf{W})$ measures the consistency of a given weight matrix.

Optimization problem: Combining the two goals, we aim to learn a weight matrix $\mathbf{W}$ via solving the following optimization problem:

$$\min_{\mathbf{W}} \ \mathcal{L}(\mathbf{W}) = \mathrm{L}(\mathbf{W}) - \lambda \mathrm{C}(\mathbf{W}), \tag{5.8}$$

where $\lambda > 0$ is a hyperparameter to balance the two goals.

Challenge for solving the optimization problem: It is computationally challenging to solve the optimization problem in Equation 5.8. The reason is that every node's posterior reputation score depends on every edge weight. For instance, we can use the gradient descent method to solve the optimization problem. Specifically, we can iteratively update the edge weight w_{uv} as follows:

$$w_{uv} \leftarrow w_{uv} - \gamma \frac{\partial \mathcal{L}(\mathbf{W})}{\partial w_{uv}}, \tag{5.9}$$

where γ is called *learning rate*. However, the derivative $\frac{\partial \mathcal{L}(\mathbf{W})}{\partial w_{uv}}$ depends on $\frac{\partial \mathbf{p}}{\partial w_{uv}}$ (derivative of every node's posterior reputation score with respect to the edge weight), due to the consistency term $\mathrm{C}(\mathbf{W})$. Since $\mathbf{p}$ is a solution to Equation 5.1, $\frac{\partial \mathbf{p}}{\partial w_{uv}}$ is a solution to the following system of equations:

$$\frac{\partial \mathbf{p}}{\partial w_{uv}} = \frac{\partial f(\mathbf{q}, \mathbf{W}, \mathbf{p})}{\partial w_{uv}}. \tag{5.10}$$

Therefore, in each iteration of gradient descent at updating the weight matrix, we need to iteratively solve Equation 5.10 with $|V|$ variables for *each* edge, which is computationally infeasible for large graphs.

5.3.3 Joint Weight Learning and Propagation

The reason for the computational challenge is that we quantify the two goals using the final posterior reputation scores. We observe that the final posterior reputation scores are iteratively computed using Equation 5.5. Therefore, we propose to quantify the two goals using the current posterior reputation scores during the iterative process. Specifically, given the posterior reputation scores in the tth iteration, we aim to learn a weight matrix such that 1) the posterior reputation scores in the $(t+1)$th iteration of the labeled positive nodes and the labeled negative nodes in

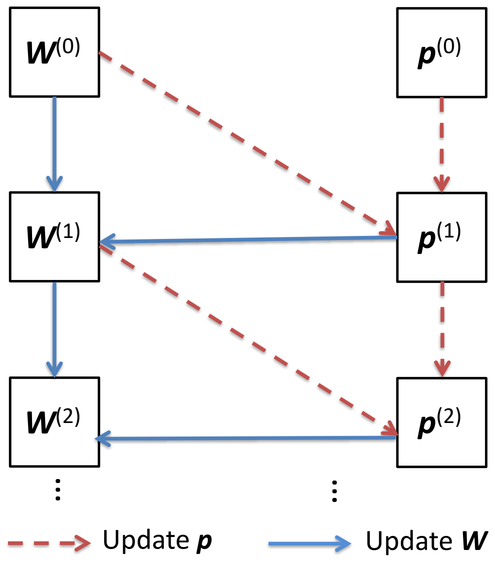


Figure 5.1 Jointly learning edge weights and updating posterior reputation scores.

the training dataset are large and small, respectively (Goal 1); and 2) the edge weights and the posterior reputation scores in the tth iteration are consistent (Goal 2). Then, we compute the posterior reputation scores in the $(t+1)$th iteration using the learnt weight matrix. In other words, we alternately learn edge weights and propagate reputation scores. Figure 5.1 illustrates our framework.

Propagating posterior reputation scores: Given the weight matrix $\mathbf{W}^{(t-1)}$ and the posterior reputation scores $\mathbf{p}^{(t-1)}$ in the $(t-1)$th iteration, we compute the posterior reputation score $\mathbf{p}^{(t)}$ in the tth iteration as follows:

$$\mathbf{p}^{(t)} = f(\mathbf{q}, \mathbf{W}^{(t-1)}, \mathbf{p}^{(t-1)}). \tag{5.11}$$

Learning weight matrix: Moreover, given the weight matrix $\mathbf{W}^{(t-1)}$ in the $(t-1)$th iteration and the posterior reputation score $\mathbf{p}^{(t)}$ in the tth iteration, we learn the weight matrix $\mathbf{W}^{(t)}$ in the tth iteration as a solution to the following optimization problem:

$$\min_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) = \frac{1}{2}\sum_{l \in L}(p_l^{(t+1)} - y_l)^2 - \lambda \sum_{(u,v)\in E} p_u^{(t)} p_v^{(t)} w_{uv}^{(t)}, \tag{5.12}$$

where the first term in the objective function $\mathcal{L}(\mathbf{W}^{(t)})$ quantifies the Goal 1, the second term quantifies the Goal 2, and $\mathbf{p}^{(t+1)} = f(\mathbf{q}, \mathbf{W}^{(t)}, \mathbf{p}^{(t)})$. From a machine learning perspective, the first term is known as a *loss function* over the training dataset, while the second term is a regularization term, which we call *consistency regularization*. We use gradient descent to solve the optimization problem. Specifically, we initialize $\mathbf{W}^{(t)}$ to be the weight matrix $\mathbf{W}^{(t-1)}$, and then we iteratively update the edge weight $w_{uv}^{(t)}$ for each edge (u, v) using Equation 5.9: $w_{uv}^{(t)} \leftarrow w_{uv}^{(t)} - \gamma \frac{\partial \mathcal{L}(\mathbf{W}^{(t)})}{\partial w_{uv}^{(t)}}$. However, different from the objective function $\mathcal{L}(\mathbf{W})$ in Equation 5.8, the derivative $\frac{\partial \mathcal{L}(\mathbf{W}^{(t)})}{\partial w_{uv}^{(t)}}$ can be efficiently computed because $\mathbf{p}^{(t)}$ is given. In particular, we have:

$$\frac{\partial \mathcal{L}(\mathbf{W}^{(t)})}{\partial w_{uv}^{(t)}} = \sum_{l \in L} (p_l^{(t+1)} - y_l) \frac{\partial p_l^{(t+1)}}{\partial w_{uv}^{(t)}} - \lambda p_u^{(t)} p_v^{(t)}. \tag{5.13}$$

For LBP-based methods on undirected graphs, the function f is described in Equation **??**. Therefore, we have:

LBP for undirected graphs:

$$\frac{\partial p_l^{(t+1)}}{\partial w_{uv}^{(t)}} = \begin{cases} p_v^{(t)} & \text{if } u = l \\ p_u^{(t)} & \text{if } v = l \\ 0 & \text{otherwise} \end{cases} \tag{5.14}$$

For LBP on directed graphs, the function f is described in Equation 5.4. Therefore, we have:

LBP for directed graphs:

$$\frac{\partial p_l^{(t+1)}}{\partial w_{uv}^{(t)}} = \begin{cases} A_{b,uv} p_v^{(t)} + A_{i,uv} \mathcal{I}(p_v^{(t)}) + A_{o,uv} \mathcal{J}(p_v^{(t)}) & \text{if } u = l \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

Note that, in gradient descent, we can repeat multiple iterations to compute the weight matrix $\mathbf{W}^{(t)}$. However, we find that our methods already work well using only one iteration. Moreover, one iteration makes our methods more efficient. Therefore, we will apply one iteration to compute $\mathbf{W}^{(t)}$.

We alternately propagate posterior reputation scores and learn edge weights until the posterior reputation scores in two consecutive alternations are small enough (e.g., $\|\mathbf{p}^{(t+1)} - \mathbf{p}^{(t)}\|_1 / \|\mathbf{p}^{(t+1)}\|_1 < 10^{-3}$) or we have reached the allowed maximum number of alternations (e.g., 15 in our experiments).

Computational complexity: In both propagating posterior reputation scores and learning edge weights, our framework traverses all edges. The time complexity of our framework is $O(|E| \cdot T)$, where T is the number of alternations. State-of-the-art LBP-based methods Pandit et al. (2007); Chau et al. (2011); Gong et al. (2014a); Wang et al. (2017b); Jia et al. (2017b); Wang et al. (2017a) have the same asymptotic time complexity. As we will demonstrate in our experiments, our framework will be 2-3 times slower than state-of-the-art LBP-based methods in practice. This is because our framework learns edge weights in each iteration. However, this time overhead is tolerable in practice, especially the targeted security and privacy applications are not time-critical. For instance, on a Twitter dataset with 1.5 billion edges, our method finishes within 3 hours on a server with 512GB memory and 32 cores. Moreover, our method can be easily parallelized and should be scalable to graphs with tens of billions of edges on a modern data center.

Table 5.1 Dataset statistics.

Dataset	**#Nodes**	**#Edges**	**Ave. degree**
Twitter	41,652,230	1,468,364,884	71
Sina Weibo	3,538,487	652,889,971	369
Yelp	520,230	718,144	3
Google+	5,735,175	30,644,909	11

5.4 Evaluation

We evaluate our framework on two security applications (Sybil detection, fake review detection) and one privacy application (attribute inference attacks in social networks). We compare our framework with state-of-the-art RW-based methods and LBP-based methods in terms of AUC, scalability, and robustness to label noise.

5.4.1 Experimental Setup

5.4.1.1 Dataset Description

We use a Twitter dataset and a Sina Weibo dataset for Sybil detection, a Yelp dataset for fake review detection, and a Google+ dataset for attribute inference attacks. Table 5.1 summarizes the basic dataset statistics.

Twitter: We obtained a Twitter dataset with real Sybils from Wang et al. Wang et al. (2017b). Specifically, the directed Twitter follower-followee graph was collected by Kwak et al. Kwak et al. (2010). A directed edge (u, v) means that u follows v. This graph contains around 42M users and 1.5B edges. A user that was suspended by Twitter is treated as a Sybil (positive), while an active user is treated as benign (negative). In total, 205,355 nodes are Sybils, 36,156,909 nodes are benign, and the remaining nodes are unlabeled.

Sina Weibo: We obtained a Sina Weibo dataset with around 3.5M users and 653M directed edges from Fu et al. Fu et al. (2017). Like 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 Sybil users, 1498 were benign users, and 20 were unknown.

Yelp: We obtained a Yelp dataset from Rayana and Akoglu (2015), which contains Yelp reviews for restaurants located in New York City. The dataset has 160,225 users, 923 products, and 359,052 reviews. By constructing an undirected User-Review-Product graph, we have 520,230 nodes and 718,144 edges. A review is treated as fake if the review is filtered or not recommended by Yelp, otherwise the review is treated as genuine. Around 10% of the reviews are fake in the dataset. In the User-Review-Product graph, we only concern the labels of the review nodes, where a fake review is treated as a positive node and a genuine review is treated as a negative node.

Google+: We obtained an undirected Google+ social network with user attributes from Gong et al. (2012); Jia et al. (2017b). The dataset has around 5.7M users and 31M edges. The user attribute is *cities lived* and the dataset has 50 most popular cities. 3.25% of users disclosed at least one of these cities as their cities lived. We treat each city as a binary attribute and perform attribute inference for each binary attribute separately. Specifically, given a city, a user is treated as positive if the user lives/lived in the city, while treated as negative if the user does not.

5.4.1.2 Training and testing

On Twitter, we sample 3,000 positive nodes and 3,000 negative nodes uniformly at random as the training set; and we treat the remaining positive and negative nodes as the testing set. On Sina Weibo, we split the 1980 labeled users into two halves; one is treated as the training set and the other as the testing set. On Yelp, we randomly sample 1,000 fake reviews and 1,000 genuine reviews as the training set; and we treat the remaining reviews as the testing set. On Google+, we select 75% of the users who have at least one city as training set and treat the remaining users who have at least one city as testing set. Recall that we have 50 cities in the Google+ dataset, where each city is treated as a binary attribute independently. The AUC of one method for the Google+ dataset is averaged over the 50 cities.

5.4.1.3 Compared methods

We compare our framework with state-of-the-art RW-based methods and LBP-based methods on undirected graphs and directed graphs.

RW-based methods for undirected graphs (RW-N-U, RW-P-U, RW-B-U, and RW-FLW-U): We consider the RW-based methods RW-N Cao et al. (2012), RW-P Wu et al. (2006), RW-B Jia et al. (2017a), and RW-FLW Boshmaf et al. (2015). Please refer to Section **??** for details about these methods. Note that the method developed by Wu et al. Wu et al. (2006) is originally for directed graphs. We apply it to undirected graphs via ignoring the edge directions. RW-FLW requires a classifier to predict the probability that a node is a victim. We consider the classifier is optimal, i.e., it correctly predicts all victims, which gives advantages to RW-FLW. More specifically, we set such probability of each victim to be 0.9 and such probability of each non-victim to be 0.1. We add a suffix "-U" to each method to indicate that they are for undirected graphs.

We note that Twitter and Sina Weibo are directed graphs. When applying these methods on the two datasets, we transform them to undirected graphs. Specifically, we keep an edge between two nodes only if they are connected by directed edges in both directions. Moreover, these methods only work for connected graphs, so we extract the largest connected component from the transformed undirected graph.

RW-based methods for directed graphs (RW-N-D and RW-P-D): We use TrustRank Gyöngyi et al. (2004) as RW-N-D and DistrustRank Wu et al. (2006) as RW-P-D. We add a suffix "-D" to indicate that these methods are for directed graphs.

LBP-based methods for undirected graphs (LBP-U, LBP-FLW-U): We consider the optimized LBP method Jia et al. (2017b) and LBP-FLW Gao et al. (2018) that learns edge weights. Like RW-based methods for undirected graphs, we transform the directed Twitter and Sina Weibo graphs into undirected ones. In LBP-FLW, we extract structure features for nodes and learn an SVM classifier to predict the prior reputation score for each unlabeled node. Then, following the setting in Gao et al. (2018), we assign a weight 0.4 to an edge if the two corresponding nodes are predicted to have the same label and a weight -0.4 otherwise. For the undirected Twitter and Sina Weibo graphs, the features are extracted from the original directed graph and include local clustering coefficient, incoming edges accepted ratio, and outgoing edges accepted ratio Gao et al. (2018). For the undirected Yelp and Google+ graphs, only the local clustering coefficient is used.

LBP-based methods for directed graphs (LBP-D): Only one LBP-based method (GANG) Wang et al. (2017a) for directed graphs was developed and we compare with it.

Our LBP-based methods (LBP-JWP-w/o, LBP-JWP-L1, LBP-JWP-L2, and LBP-JWP): LBP-JWP-w/o is the variant of our LBP-based method that does not use the consistency term in the optimization problem in Equation 5.12 when learning edge weights. From the machine learning perspective, the first term in the objective function of the optimization problem in Equation 5.12 is a *loss function* over the training dataset, and the second term is a regularization term about the edge weights, which we call *consistency regularization.* In machine learning, L_1 and

L_2 regularizations are widely used, which are $-\sum_{(u,v)\in E} |w_{uv}|_1$ and $-\sum_{(u,v)\in E} w_{uv}^2$, respectively. LBP-JWP-L1 and LBP-JWP-L2 respectively are the variants that use the conventional L_1 and L_2 regularizations instead of our consistency regularization when learning edge weights. LBP-JWP is our method with the consistency regularization. We add the suffix "-U" and "-D" to indicate the versions that are used for undirected and directed graphs, respectively.

Table 5.2 AUCs of undirected methods.

Methods		Twitter	Sina Weibo	Yelp	Google+
RW	**RW-N-U**	0.57	0.61	0.55	0.59
	RW-P-U	0.58	0.61	0.57	0.58
	RW-LFW-U	0.53	0.54	0.48	0.57
	RW-B-U	0.63	0.68	0.58	0.63
LBP	**LBP-U**	0.64	0.68	0.58	0.66
	LBP-FLW-U	0.62	0.66	0.58	0.66
Ours	**LBP-JWP-w/o-U**	0.69	0.74	0.60	0.69
	LBP-JWP-L1-U	0.65	0.70	0.59	0.66
	LBP-JWP-L2-U	0.68	0.72	0.60	0.68
	LBP-JWP-U	**0.73**	**0.77**	**0.62**	**0.72**

5.4.1.4 Implementation and parameter setting

We implement the RW-based methods and our methods in C++. The authors of the existing LBP-based methods Wang et al. (2017b,a) made their C++ source code publicly available, and we use them to evaluate existing LBP-based methods. We performed all our experiments on a Linux machine with 512GB memory and 32 cores. For our methods, we assign prior reputation scores 1, -1, and 0 to labeled positive nodes, labeled negative nodes, and unlabeled nodes, respectively. Following Wang et al. (2017b) and Wang et al. (2017a), we initialize all edge weights to be the inverse of the average node degree in a graph, i.e., $w_{uv}^{(0)} = \frac{1}{\text{Ave. degree}}$. Moreover, we normalize edge

Table 5.3 AUCs of directed methods.

Methods		Twitter	Sina Weibo
RW	**RW-N-D**	0.60	0.66
	RW-P-D	0.63	0.64
LBP	**LBP-D**	0.72	0.80
Ours	**LBP-JWP-w/o-D**	0.75	0.82
	LBP-JWP-L1-D	0.72	0.79
	LBP-JWP-L2-D	0.73	0.80
	LBP-JWP-D	**0.78**	**0.85**

weights to be in the range [-0.5,0.5] in each iteration. We set the learning rate γ in Equation 5.9 to be 1.0. Moreover, we set the regularization parameter $\lambda = 0.1$ on the Twitter and Sina Weibo datasets and $\lambda = 1.0$ on the Google+ and Yelp datasets considering their different average node degrees. For all other compared methods, we set the parameters according to their authors.

5.4.2 Accuracy (AUC)

Table 5.2 and Table 5.3 respectively show the AUCs of the compared undirected graph and directed graph based methods.

Our methods outperform existing ones: Our methods LBP-JWP-U and LBP-JWP-D achieve the best AUCs consistently on different datasets. For instance, on undirected graphs, LBP-JWP-U improves upon the best existing method by 0.04 to 0.09 on the four datasets. This means that our strategy of jointly learning edge weights and propagating reputation scores is effective. Moreover, LBP-JWP outperforms LBP-JWP-w/o, LBP-JWP-L_1, and LBP-JWP-L_2 on both undirected and directed graphs. This means that our consistency regularization does improve AUC and outperforms conventional L_1 and L_2 regularizations. In fact, L_1 and L_2 regularizations decrease AUCs, compared to LBP-JWP-w/o that does not use regularization.

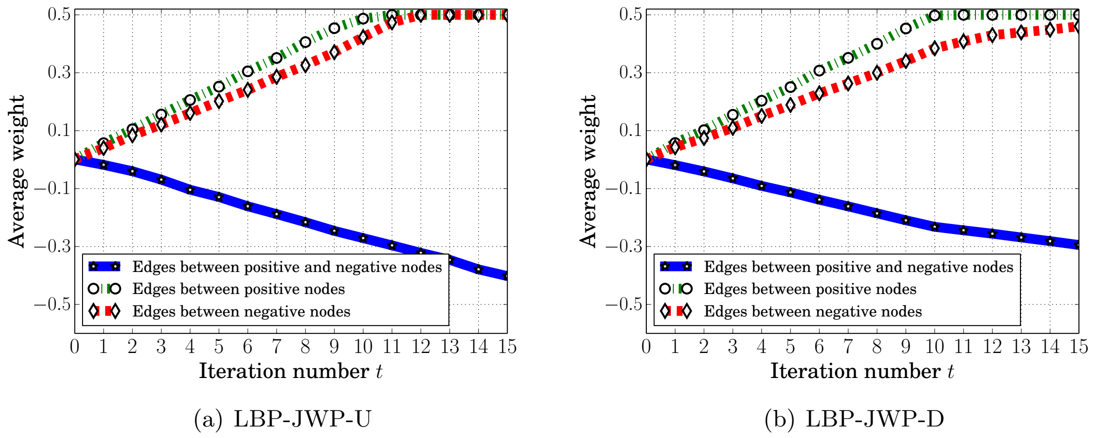
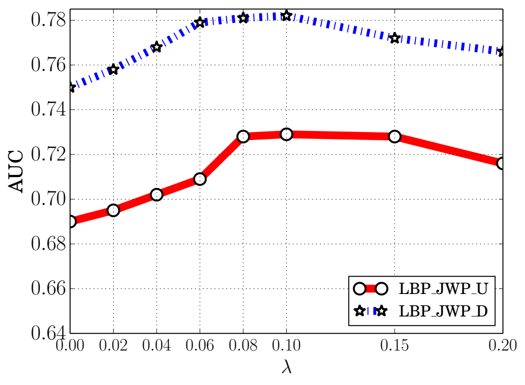


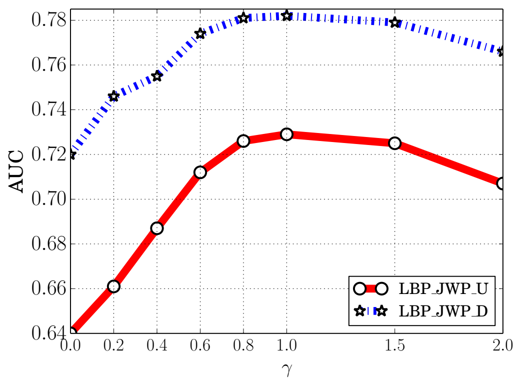
Figure 5.2 Average weight learnt by (a) LBP-JWP-U and (b) LBP-JWP-D of edges between positive nodes and negative nodes, edges between positive nodes, and edges between negative nodes on Twitter, as the number of iterations increases. We observe that on both methods, the average weights of edges between positive nodes and negative nodes decrease, while the average weights of edges between negative (or positive) nodes increase as t increases.

Note that all compared methods have relatively low AUCs on Yelp. This is because the Yelp graph is very sparse, i.e., Ave. degree=3, and thus its graph structure has less information that can be exploited by graph-based methods.

To further understand why our framework outperforms existing methods, we visualize the learnt edge weights. Specifically, we classify edges into three categories: 1) edges between positive nodes and negative nodes, 2) edges between positive nodes, and 3) edges between negative nodes. The edges in the first category are heterogeneous, while the edges in the second and third categories are homogeneous. Figure 5.2 shows the average weights of the three categories of edges as a function of the number of iterations t on the Twitter dataset. As we can see, for both LBP-JWP-U and LBP-JWP-D, the average weights of edges between positive nodes and negative nodes decrease, while the average weights of edges between negative (or positive) nodes increase as the number of iterations increases. This is because we enforce the consistency regularization when learning the edge weights.



(a) Impact of λ



(b) Impact of γ

Figure 5.3 Impact of (a) the regularization parameter λ and (b) the learning rate γ. As λ and γ increase, AUCs of our methods first increase, then stabilize, and finally decrease.

However, for most existing methods, the edge weights are constant, which ignore the differences between heterogeneous and homogeneous edges. RW-FLW learns edge weights. However, we find that the average weights of all three types of edges are close to -0.3 (after normalizing edge weights to be [-0.5, 0.5]). This is because a large number of nodes are victims. Moreover, for LBP-FLW, the average weights of all three types of edges are close to 0.2. This means that LBP-FLW can not learn small weights for heterogeneous edges. Note that all these methods fix the edge weights and propagate reputation scores using the fixed edge weights.

LBP-based methods outperform RW-based methods: We find that LBP-based methods achieve at least the same AUCs as RW-based methods, and LBP-based methods achieve higher AUCs in most cases. In particular, for methods on undirected graphs, the best existing LBP-based method achieves the same AUCs as the best existing RW-based method on Sina Weibo and Yelp, while the best existing LBP-based method achieves slightly higher AUCs on Twitter and Google+. For methods on directed graphs, existing LBP-based method significantly outperforms existing RW-based methods. For instance, on Sina Weibo, LBP-D outperforms RW-N-D by 0.14.

A possible reason is that LBP-D leverages both labeled positive nodes and labeled negative nodes in the training dataset, while RW-N-D only considers labeled negative nodes.

Different variants of RW-based methods: We observe that RW-B-U consistently outperforms other RW-based methods for undirected graphs. One reason could be that RW-B-U incorporates both labeled positive nodes and labeled negative nodes in the training dataset, while the other RW-based methods incorporate either of them. Performance of RW-N and RW-P is dataset-dependent, i.e., RW-N outperforms RW-P on some datasets while RW-P outperforms RW-N on other datasets.

Impact of the regularization parameter λ and the learning rate γ: Figure 5.3(a) shows AUCs of our methods vs. the regularization parameter λ on the Twitter dataset (we have similar tendencies on the other datasets and thus choose Twitter for simplicity). $\lambda = 0$ means that we do not use the consistency term. We observe that AUCs first increase, then stabilize, and finally decrease as λ increases. For instance, AUC of LBP-JWP-U increases until λ is around 0.08, then stabilizes until λ is around 0.15, and finally decreases after that. Figure 5.3(b) shows AUCs of our methods vs. the learning rate γ on the Twitter dataset. $\gamma = 0$ means that we do not learn edge weights, and our methods reduce to existing methods. Likewise, we observe that AUCs first increase, then stabilize, and finally decrease.

Impact of the number of iterations t: Figure 5.4 shows the AUCs of our methods on Twitter vs. iteration number t. We observe that AUCs first gradually increase as our methods run more iterations and then stabilize after around 10 iterations.

5.4.3 Scalability

We measure scalability of the compared methods with respect to the number of edges in the graph. Since we need graphs with different number of edges, we synthesize graphs using a network generation model. In particular, we use the popular Preferential Attachment (PA) model Barabási and Albert (1999) developed by the network science community to generate graphs. Specifically, we first use the PA model to synthesize an undirected graph G. Then, we treat each undirected

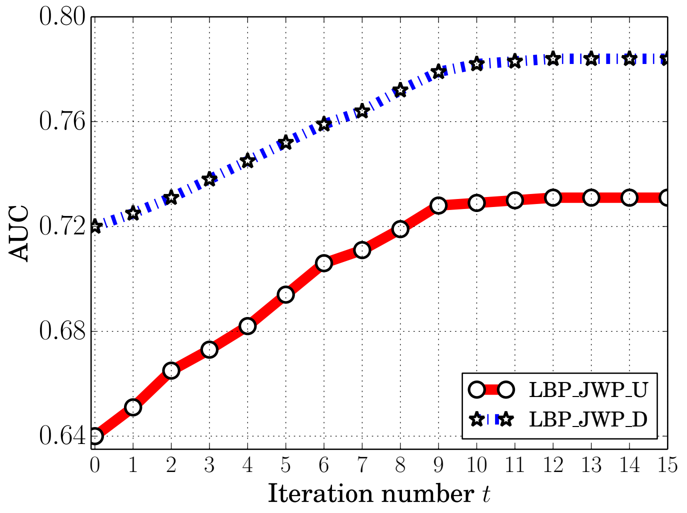
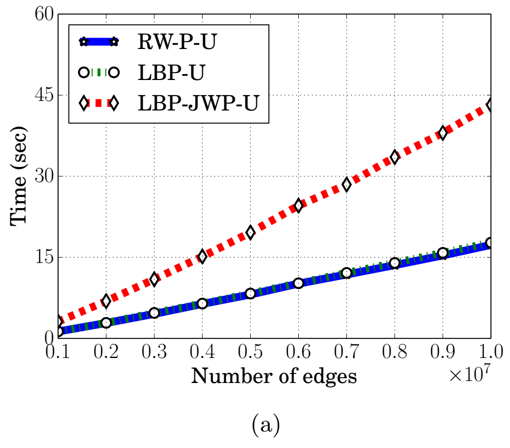


Figure 5.4 Impact of the number of iterations on the AUCs of our methods. We observe that AUCs first gradually increase as we perform more iterations and then stabilize.

edge (u, v) in G as two directional edges (u, v) and (v, u). Finally, we sample 50% of directed edges and remove them to construct a directed graph G'.

We run methods for undirected graphs on G and methods for directed graphs on G'. We note that all the compared methods iteratively solve the posterior reputation scores. For fair comparison, we run the iterative processes with the same number of iterations (e.g., 10 in our experiments). Figure 5.5(a) and Figure 5.5(b) respectively show the running time of these methods on undirected graphs and directed graphs as the number of (undirected and directed) edges increases. Note that RW-N-U, RW-P-U, and RW-FLW-U have the same running time and thus we only show RW-P-U for simplicity. Similarly, RW-N-D and RW-P-D have the same running time and thus we only show RW-P-D for simplicity.

Our methods have an acceptable computational overhead: We observe that our methods LBP-JWP-U and LBP-JWP-D are less efficient than existing methods. This is because our methods jointly learn edge weights and propagate reputation scores, while existing methods only



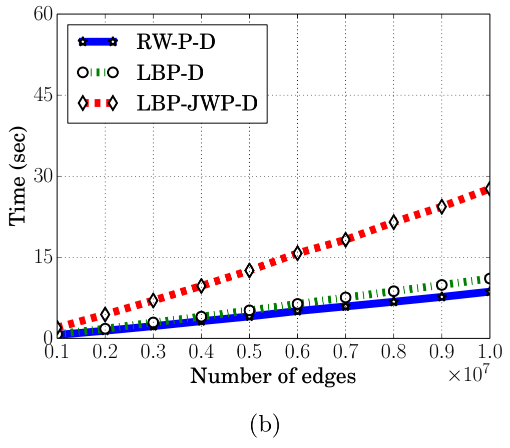
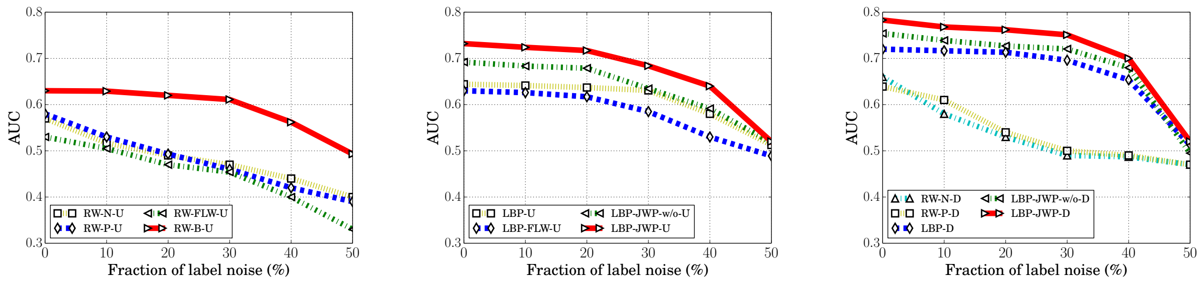


Figure 5.5 Running time on synthesized graphs with increasing number of edges. (a) Methods for undirected graphs. (b) Methods for directed graphs.

propagate reputation scores. However, the computational overhead of our methods is acceptable. In particular, our methods are 2-3 times slower than state-of-the-art methods.

Undirected graphs vs. directed graphs: When an undirected graph and a directed graph have the same number of edges, a method for the directed graph is more efficient than its variant for the undirected graph. For instance, RW-P-D is more efficient than RW-P-U, LBP-D is more efficient than LBP-U, LBP-JWP-D is more efficient than LBP-JWP-U. The reason is that RW-P-D traverses each directed edge once, while RW-P-U traverses each undirected edge twice in each iteration of propagating reputation scores; LBP-D (or LBP-JWP-D) only traverses an edge (u, v) twice when the directed edge (v, u) does not exist, while LBP-U (or LBP-JWP-U) traverses each undirected edge twice in each iteration.

LBP-based methods vs. RW-based methods: For undirected graphs, LBP-U and RW-P-U have the same running time. This is because both methods traverse each undirected edge twice in each iteration of propagating reputation scores. However, on directed graphs, LBP-D is slightly slower than RW-P-D. This is because RW-P-D traverses each directed edge once in each iteration, while LBP-D traverses an edge (u, v) twice when the directed edge (v, u) does not exist. These results are consistent with previous studies Wang et al. (2017b,a).



(a) (b) (c)

Figure 5.6 Impact of label noise on Twitter. (a) RW-based methods for undirected graphs; (b) LBP-based methods for undirected graphs; (c) Methods for directed graphs.

5.4.4 Robustness to Label Noise

We randomly sample $\alpha\%$ of labeled positive nodes and labeled negative nodes in the training dataset, and change their labels to be negative and positive, respectively. Therefore, the label noise level in the training dataset is $\alpha\%$. We repeat the sampling process 5 times. Figure 5.6(a), Figure 5.6(b), and Figure 5.6(c) respectively show the average AUCs on Twitter for undirected graph based methods using RW, undirected graph based methods using LBP, and directed graph based methods, as we increase $\alpha\%$ from 0% to 50%. We cut off at the 50% because no methods are effective for a label noise level that is higher than 50%.

First, LBP-JWP consistently performs the best and can tolerate label noise up to 20% on the undirected graph and 30% on the directed graph. For instance, LBP-JWP-D's AUC only slightly decreases when the label noise is 30%. Moreover, LBP-JWP has higher AUCs than LBP-JWP-w/o, which shows the effectiveness of our consistency regularization. Second, LBP-based methods and RW-B-U can tolerate relatively larger label noise than RW-N and RW-P. A possible reason is that LBP-based methods and RW-B-U leverage both labeled positive nodes and labeled negative nodes in the training dataset, while RW-N and RW-P only use labeled negative nodes or labeled positive nodes.

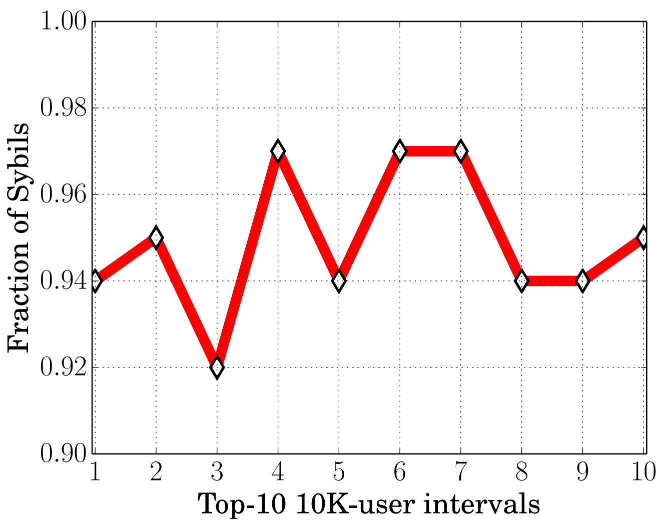


Figure 5.7 Fraction of Sybils in the sampled 100 users in each of the top-10 10K-user intervals for LBP-JWP-D on the Sina Weibo dataset.

5.4.5 Case Study on Sina Weibo

We apply our LBP-JWP-D to detect Sybils on Sina Weibo. Specifically, we use the 1,980 labeled users as a training dataset, leverage LBP-JWP-D to compute a posterior reputation score for every node, and rank all unlabeled nodes in a descending order with respect to their posterior reputation scores. Therefore, the nodes that are ranked higher could be more likely to be Sybils. In practice, social network providers often rely on human workers to manually inspect users and detect Sybils Cao et al. (2012). The ranking list produced by our method can be used as a priority list for the human workers, i.e., the human workers can inspect users according to the ranking list. Within the same amount of time, the human workers could find more Sybils when inspecting users according to the ranking list than inspecting users that are randomly picked.

We follow the same strategy of Wang et al. (2017a) to evaluate the top-ranked 100K users produced by our method. In particular, via manual inspection, Wang et al. Wang et al. (2017a) classified users on Sina Weibo into five categories: *Suspended users*, *Spammers*, *Compromised users*, *Normal users*, and *Unknown users*. The first three categories (i.e., suspended users, spammers, and compromised users) are treated as Sybils, and the normal users are treated as benign. We divide

Table 5.4 Composition of the manually labeled 1,000 randomly sampled users.

Category		**LBP-D**		**LBP-JWP-D**	
Sybil	Suspended Users	41.5%	92.0%	41.9%	94.9%
	Spammers	42.5%		44.3%	
	Compromised Users	8.0%		8.7%	
Benign		6.8%		4.3%	
Unknown		1.2%		0.8%	

the top-ranked 100K users into 10 10K-user intervals. We randomly sample 100 users from each interval, and thus we have 1,000 sampled users in total. We manually inspected the Sina Weibo pages of the 1,000 sampled users and labeled them to be one of the five categories following the same strategy in Wang et al. (2017a).

Table 5.4 shows the manual labeling results of the 1,000 sampled users. For comparison, we also include the results for LBP-D, which are from Wang et al. (2017a) instead of being labeled by us. We note that LBP-D produces a different ranking list from our method, and the sampled 1,000 users are also different for them. We observe that 94.9% of the nodes ranked by our method are Sybils, around 3% higher than those ranked by LBP-D Wang et al. (2017a). We note that Íntegro Boshmaf et al. (2015) has a similar accuracy at detecting Sybils in Tuenti. Moreover, our method detects a large number of Sybils that evaded Sina Weibo's detector. In particular, if a Sybil was detected by Sina Weibo, then the Sybil should have been suspended or deleted. However, 44.3% of our top-ranked 100K nodes are spammers and 8.7% of them are compromised users, all of which evaded Sina Weibo's detector. Figure 5.7 shows the fraction of Sybils in the sampled 100 users in each 10K-user interval. We observe that even if the training dataset is small (around 0.05% of all users in the dataset), above 90% of nodes are Sybils in each of the top-10 10K-user interval ranked by our method.

Table 5.5 AUCs of our framework for RW-based methods.

Methods	**Twitter**	**Sina Weibo**	**Yelp**	**Google+**
RW-B-U	0.63	0.68	0.58	0.63
RW-JWP-w/o-U	0.66	0.72	0.58	0.65
RW-JWP-U	0.69	0.74	0.60	0.68

5.5 Discussion and Limitation

Applying our framework to RW-based methods: For conciseness, we focus on applying our framework to LBP-based collective classification methods for graph-based security and privacy analytics. However, our framework is also applicable to RW-based methods. In particular, RW-based methods can also be viewed as iteratively solving system of equations, though the function $f(\mathbf{q}, \mathbf{W}, \mathbf{p})$ is different for RW-based methods. Therefore, our framework can also learn edge weights for RW-based methods. We implemented our framework to learn edge weights for RW-B-U Jia et al. (2017a), a RW-based method for undirected graphs that incorporates both labeled positive nodes and labeled negative nodes in the training dataset. The function f for RW-B-U is $f(\mathbf{q}, \mathbf{W}, \mathbf{p}) = \mathbf{W}\mathbf{p}$, where w_{uv} is the edge weight of (u, v) divided by the weighted degree of u. Table 5.5 shows the AUCs of RW-B-U, RW-JWP-w/o-U, and RW-JWP-U for the four datasets. RW-JWP-U uses our framework to learn edge weights for RW-B-U, while RW-JWP-w/o-U uses our framework without the consistency regularization to learn edge weights for RW-B-U. Our results show that our framework improves AUCs for RW-B-U and the consistency regularization helps improve AUCs.

Computational overhead: Theoretically, our method has the same asymptotic time complexity as state-of-the-art random walk based and belief propagation based methods: linear to the number of edges in the graph. Empirically, our method is 2-3 times slower. However, this time overhead is tolerable in practice, especially the targeted security and privacy applications are not time-critical. For instance, on the Twitter dataset with 1.5 billion edges, our method finishes within 3 hours on a server with 512GB memory and 32 cores. Our method can be easily parallelized and should be scalable to graphs with billions of edges on a modern data center.

No manual labels: Like existing methods, our framework relies on a manually labeled training dataset. When there are no such training dataset, our framework may also be applicable. Specifically, Wang et al. Wang et al. (2018b) recently proposed SybilBlind, which generalizes LBP-based collective classification methods to the scenarios where no training dataset is available. Their key idea is to randomly sample some nodes from the graph and treat them as labeled nodes (some of them would be labeled incorrectly, i.e., the training data has noise). Then, they apply a LBP-based method with the sampled training data. Since LBP-based methods are robust to some noise in training data, they repeat the sampling process multiple times and design some method to select the sampling trials that have small label noise. We believe such idea can also be applied to our framework when no manual labels are available, because our framework is also robust to some noise in training data.

Comparing with other graph-based classification methods: A graph-based classification problem can also be solved via *graph embedding* methods and Graph Convolutional Networks (GCN) Kipf and Welling (2017). In particular, a graph embedding method first learns feature vectors for nodes via unsupervised learning, and then learns a classifier (logistic regression classifier in our experiments) based on the feature vectors and the training dataset. We compare with three state-of-the-art graph embedding methods (DeepWalk Perozzi et al. (2014), LINE Tang et al. (2015), and node2vec Grover and Leskovec (2016)) as well as GCN for Sybil detection. We obtained the public source code of these methods from their authors. As DeepWalk, node2vec, and GCN are not scalable to the large graphs used in our experiments, we perform experiments on an undirected Facebook graph with synthetic Sybils. The Facebook graph contains 4,039 negative nodes and 88,234 edges. Following prior work Wang et al. (2018a), we replicate the nodes and their edges as positive nodes and edges. Moreover, we add 10k edges between the positive nodes and negative nodes uniformly at random. We randomly select 100 positive nodes and 100 negative nodes as the training dataset. A graph embedding method learns a 128-dimension feature vector for each node. We set the number of hidden units of GCN to be 64 and set the dropout rate as 0.5. We observe that our method is more accurate than the compared methods. Specifically, DeepWalk, LINE, node2vec, and GCN achieve AUCs of 0.62, 0.94, 0.74, and 0.88, respectively, while our LBP-JWP-U achieves an AUC of 1. We speculate the reason is that graph embedding aims to learn general feature representations that could be used for various purposes (classification is just one of the purposes) and GCN is not able to learn discriminative hidden features for classification.

5.6 Summary

The chapter has proposed a novel framework to learn edge weights for graph-based security and privacy analytics. The framework can be applied to various security and privacy problems including, but not limited to, Sybil detection in social networks, malware detection, fake review detection, and attribute inference attacks.

Moreover, the framework can be incorporated into a state-of-the-art collective classification method (both RW-based methods and LBP-based methods) to further enhance its classification accuracy with an acceptable computational overhead. Experimental results demonstrate that jointly learning edge weights and propagating reputation scores is effective for graph-based security and privacy analytics.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This dissertation has a comprehensive study of collective classification methods for graph-based security and privacy problems. In Chapter 1, we first illustrated the setting and key steps of collective classification, demonstrated that various security and privacy problems can be modeled as collective classfication, and surveyed representative graph-based security and privacy methodss. Then, we summarized the advantages and disadvantages of existing RW-based and LBP-based collective classification methods. Our purpose is to design novel collective classification methods to address the disadvantages and maintain the advantages of existing methods.

In Chapter 2, we unified existing RW-based and LBP-based methods into a local rule-based framework. Under our framework, designing better collective classification methods reduces to designing better local rules. In Chapter 3, we designed a novel local rule for undirected graph-based Sybil detection in social networks. Then, we developed a novel collective classification method that integrates advantages of both RW-based and LBP-based methods and overcomes their limitations. In Chapter 4, we further studied directed graph-based Sybil detection in social networks. Sybil detection, as well as other security and privacy problems, in directed graphs has its unique characteristics. By capturing the unique characteristics, we developed a customized pairwise Markov Random Field model on directed graphs, and inferred the model via an optimized LBP.

In Chapter 5, we targeted the long-standing edge weight learning problem in graph-based security and privacy analytics, and developed an edge weight learning framework. The key idea of our framework is to formulate edge weight learning as an optimization problem and then design approximation algorithms to efficiently solve the relaxed optimization problem. Our framework is general, as it can be applied to various security and privacy problems and incorporated into both RW-based and LBP-based collective classification methods.

6.2 Future Work

6.2.1 Collective Classification under Adversarial Settings

In this dissertation, we have evaluated collective classification methods with respect to accuracy (AUC, top-ranking performance), scalability, convergence, and robustness to *random* label noise. However, we need to further study the security of collective classification methods under adversarial settings (i.e., in the presence of attackers). For instance, how can an attacker inject carefully crafted label noise to reduce accuracy?

In future, we will explore the complete attack surface of existing collective classification methods. Specifically, we will characterize a threat model by considering attacker's goal (i.e., what does an attacker aim to achieve?), attacker's knowledge (i.e., what does an attacker know about the setting of a collective classification method), and attacker's capability (i.e., how can an attacker manipulate a collective classification). Under the threat model, we will design attack strategies as well as defenses.

6.2.2 Incorporating Problem-Specific Characteristics

In this dissertation, we only used the graph structure to study security and privacy problems. For a specific security or privacy problem, nodes and edges in the graph could have problem-specific rich features. In future, we will extend our methods to incorporate such problem-specific features to further enhance accuracy. For instance, for Sybil detection in social networks, each node could have rich content and behavior features. Our methods can use these node features to learn the prior reputation scores of nodes. Moreover, an edge could have features such as interaction frequency between the two users. We can model the weight of an edge as a function of the edge's features, e.g., a logistic function. Then, we can design algorithms to learn the parameters in the function. For instance, we can design an objective function with respect to such parameters and then use gradient descent to optimize the parameters.

6.2.3 Generalizing Our Methods to Other Application Domains

In this dissertation, we have studied collective classification methods for Sybil detection in social networks (e.g., Twitter and Sina Weibo), attribute inference attacks in social networks (e.g., Google+), and fake review detection in Yelp.

Recently, we have obtained the URL redirection graph and the domain redirection graph from collaborators; the user-store ordering graph from Jingdong. Therefore, in future, we will also consider generalizing our collective classification methods to detect malicious domains and malicious users in e-commerce in these datasets.

BIBLIOGRAPHY

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

Alvisi, L., Clement, A., Epasto, A., Lattanzi, S., and Panconesi, A. (2013). Sok: The evolution of sybil defense via social networks. In *IEEE S & P*.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286.

Bilge, L., Strufe, T., Balzarotti, D., and Kirda, E. (2009). All your contacts are belong to us: Automated identity theft attacks on social networks. In *WWW*.

Boshmaf, Y., Logothetis, D., Siganos, G., Leria, J., Lorenzo, J., Ripeanu, M., and Beznosov, K. (2015). Integro: Leveraging victim prediction for robust fake account detection in osns. In *NDSS*.

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

Chaabane, A., Acs, G., and Kaafar, M. A. (2012). You are what you like! information leakage through users' interests. In *NDSS*.

Chakrabarti, D., Funiak, S., Chang, J., and Macskassy, S. A. (2017). Joint label inference in networks. *JMLR*.

Chau, D. H. P., Nachenberg, C., Wilhelm, J., Wright, A., and Faloutsos, C. (2011). Polonium: Tera-scale graph mining and inference for malware detection. In *SIAM SDM*.

Danezis, G., Diaz, C., Troncoso, C., and Laurie, B. (2010). Drac: An architecture for anonymous low-volume communications. In *PETS*.

Danezis, G. and Mittal, P. (2009). SybilInfer: Detecting Sybil nodes using social networks. In *NDSS*.

Derzko, N. and Pfeffer, A. (1965). Bounds for the spectral radius of a matrix. *Mathematics of Computation*, 19(89).

Erdos, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60.

Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R. (2013). Exploiting burstiness in reviews for review spammer detection. In *ICWSM*.

Feng, Q., Zhou, R., Xu, C., Cheng, Y., Testa, B., and Yin, H. (2016). Scalable graph-based bug search for firmware images. In *ACM CCS*.

Freeman, D. M. (2017). Can you spot the fakes? on the limitations of user feedback in online social networks. In *WWW*.

Fu, H., Xie, X., Rui, Y., Gong, N. Z., Sun, G., and Chen, E. (2017). Robust spammer detection in microblogs: Leveraging user carefulness. *ACM TIST*.

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

Gao, P., Wang, B., Gong, N. Z., Kulkarni, S. R., Thomas, K., and Mittal, P. (2018). Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *IEEE CNS*.

Gatterbauer, W. (2017). The linearization of belief propagation on pairwise markov random fields. In *AAAI*.

Gatterbauer, W., Günnemann, S., Koutra, D., and Faloutsos, C. (2015). Linearized and single-pass belief propagation. *PVLDB*, 8(5).

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., Ganguly, N., and Gummadi, K. P. (2012). Understanding and combating link farming in the twitter social network. In *WWW*.

Gilbert, E. and Karahalios, K. (2009). Predicting tie strength with social media. In *CHI*.

Gong, N. Z., Frank, M., and Mittal, P. (2014a). Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE TIFS*, 9(6):976–987.

Gong, N. Z. and Liu, B. (2016). You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security Symposium*.

Gong, N. Z. and Liu, B. (2018). Attribute inference attacks in online social networks. *ACM TOPS*.

Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., Shi, E. R., and Song, D. (2014b). Joint link prediction and attribute inference using a social-attribute network. *ACM TIST*.

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

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *ACM KDD*.

Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. (2004). Combating web spam with trustrank. In *VLDB*.

Hacking Election. (2016).

Hacking Financial Market. (2016).

Jia, J., Wang, B., and Gong, N. Z. (2017a). Random walk based fake account detection in online social networks. In *DSN*, pages 273–284. IEEE.

Jia, J., Wang, B., Zhang, L., and Gong, N. Z. (2017b). Attriinfer: Inferring user attributes in online social networks using markov random fields. In *WWW*.

Jiang, M., Cui, P., Beutel, A., Faloutsos, C., and Yang, S. (2014). Catchsync: catching synchronized behavior in large directed graphs. In *KDD*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *ICLR*.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5).

Koutra, D., Ke, T.-Y., Kang, U., Chau, D. H. P., Pao, H.-K. K., and Faloutsos, C. (2011). Unifying guilt-by-association approaches: Theorems and fast algorithms. In *ECML/PKDD*.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM.

Kwon, B. J., Mondal, J., Jang, J., Bilge, L., and Dumitras, T. (2015). The dropper effect: Insights into malware distribution with downloader graph analytics. In *ACM CCS*.

Levin, D. A., Peres, Y., and Wilmer, E. L. (2009). *Markov chains and mixing times*. American Mathematical Soc.

Li, H., Chen, Z., Liu, B., Wei, X., and Shao, J. (2014). Spotting fake reviews via collective positive-unlabeled learning. In *IEEE ICDM*.

Li, Z., Alrwais, S., Xie, Y., Yu, F., and Wang, X. (2013). Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *IEEE S & P*.

Liu, Y., Ji, S., and Mittal, P. (2016). Smartwalk: Enhancing social network security via adaptive random walks. In *CCS*.

Mohaisen, A., Hopper, N., and Kim, Y. (2011). Keep your friends close: Incorporating trust into social network-based sybil defenses. In *IEEE INFOCOM*.

Olteanu, A.-M., Huguenin, K., Shokri, R., Humbert, M., and Hubaux, J.-P. (2017). Quantifying interdependent privacy risks with location data. *IEEE TMC*.

Oprea, A., Li, Z., Yen, T.-F., Chin, S. H., and Alrwais, S. (2015). Detection of early-stage enterprise infection by mining large-scale log data. In *DSN*.

Pandit, S., Chau, D. H., Wang, S., and Faloutsos, C. (2007). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference.*

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *ACM KDD*.

Rajab, M. A., Ballard, L., Lutz, N., Mavrommatis, P., and Provos, N. (2013). Camp: Content-agnostic malware protection. In *NDSS*.

Rayana, S. and Akoglu, L. (2015). Collective opinion spam detection: Bridging review networks and metadata. In *ACM KDD*.

Saad, Y. (2003). *Iterative methods for sparse linear systems.* Siam.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*.

Stringhini, G., Shen, Y., Han, Y., and Zhang, X. (2017). Marmite: spreading malicious file reputation through download graphs. In *ACSAC*.

Tamersoy, A., Roundy, K., and Chau, D. H. (2014). Guilt by association: large scale malware detection by mining file-relation graphs. In *ACM KDD*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *WWW*, pages 1067–1077.

Thomas, K., Grier, C., Ma, J., Paxson, V., and Song, D. (2011). Design and evaluation of a real-time url spam filtering service. In *IEEE S & P*.

Viswanath, B., Post, A., Gummadi, K. P., and Mislove, A. (2010). An analysis of social network-based Sybil defenses. In *ACM SIGCOMM*.

Wang, B., Gong, N. Z., and Fu, H. (2017a). GANG: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *IEEE ICDM*.

Wang, B., Jia, J., Zhang, L., and Gong, N. Z. (2018a). Structure-based sybil detection in social networks via local rule-based propagation. *IEEE TNSE*.

Wang, B., Zhang, L., and Gong, N. Z. (2017b). Sybilscar: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM*.

Wang, B., Zhang, L., and Gong, N. Z. (2018b). Sybilblind: Detecting fake users in online social networks without manual labels. In *RAID*.

Wang, G., Mohanlal, M., Wilson, C., Wang, X., Metzger, M., Zheng, H., and Zhao, B. Y. (2013). Social turing tests: Crowdsourcing Sybil detection. In *NDSS*.

Wang, Y. and Nakao, A. (2010). Poisonedwater: An improved approach for accurate reputation ranking in p2p networks. *FGCS*, 26(8):1317–1326.

Wei, W., Xu, F., Tan, C., and Li, Q. (2012). SybilDefender: Defend against Sybil attacks in large social networks. In *IEEE INFOCOM*.

Wilson, C., Boe, B., Sala, A., Puttaswamy, K. P., and Zhao, B. Y. (2009). User interactions in social networks and their implications. In *Eurosys*.

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

Xu, X., Liu, C., Feng, Q., Yin, H., Song, L., and Song, D. (2017). Neural network-based graph embedding for cross-platform binary code similarity detection. In *ACM CCS*.

Yang, C., Harkreader, R., Zhang, J., Shin, S., and Gu, G. (2012). Analyzing spammer's social networks for fun and profit. In *WWW*.

Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., and Dai, Y. (2011). Uncovering social network Sybils in the wild. In *ACM IMC*.

Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., and Dai, Y. (2014). Uncovering social network sybils in the wild. *ACM TKDD*.

Ye, Y., Li, T., Zhu, S., Zhuang, W., Tas, E., Gupta, U., and Abdulhayoglu, M. (2011). Combining file content and file relations for cloud based malware detection. In *ACM KDD*.

Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. (2008). SybilLimit: A near-optimal social network defense against Sybil attacks. In *IEEE S & P*.

Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. (2006). Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM*. ACM.

Zhang, J., Zhang, R., Sun, J., Zhang, Y., and Zhang, C. (2016). Truetop: A sybil-resilient system for user influence measurement on twitter. *IEEE/ACM ToN*.

Zheleva, E. and Getoor, L. (2009). To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW*.

Zheng, H., Xue, M., Lu, H., Hao, S., Zhu, H., Liang, X., and Ross, K. (2018). Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks. In *NDSS*.

Zhu, X., Ghahramani, Z., Lafferty, J., et al. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

APPENDIX. THEOREM PROOFS

Proof of Theorem 3 in Section 3.3.1

We denote $\mathcal{Z}_u = q_u \prod_{v\in\Gamma(u)} f_{vu} + (1-q_u)\prod_{v\in\Gamma(u)}(1-f_{vu})$. Rewriting $p_u = \frac{1}{\mathcal{Z}_u} q_u \prod_{v\in\Gamma(u)} f_{vu}$ with the corresponding residual variables yields

$$\begin{aligned}
0.5+\hat{p}_u &= \frac{1}{\mathcal{Z}_u}(0.5+\hat{q}_u)\prod_{v\in\Gamma(u)}(0.5+\hat{f}_{vu})\\
\Longrightarrow \ln(1+2\hat{p}_u) &= -\ln\mathcal{Z}_u + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\left(0.5+\hat{f}_{vu}\right)\\
&= -\ln\mathcal{Z}_u + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\left(0.5\right) + \sum_{v\in\Gamma(u)} \ln(1+2\hat{f}_{vu})
\end{aligned}$$

Using approximation $\ln(1+x)\approx x$ when x is small, we have:

$$2\hat{p}_u = -\ln\mathcal{Z}_u + 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) + \sum_{v\in\Gamma(u)} 2\hat{f}_{vu}. \tag{.1}$$

Similarly, via rewriting $1-p_u = \frac{1}{\mathcal{Z}_u}(1-q_u)\prod_{v\in\Gamma(u)}(1-f_{vu})$ with the corresponding residual variables and using approximation $\ln(1-x)\approx -x$ when x is small, we have:

$$-2\hat{p}_u = -\ln\mathcal{Z}_u - 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) - \sum_{v\in\Gamma(u)} 2\hat{f}_{vu}. \tag{.2}$$

Adding Equation .1 with Equation .2 yields $\ln\mathcal{Z}_u = |\Gamma(u)|\cdot\ln(0.5)$. Via substituting this relation into Equation .1 or Equation .2, we have:

$$\hat{p}_u = \hat{q}_u + \sum_{v\in\Gamma(u)} \hat{f}_{vu}. \tag{.3}$$

Proof of Theorem 7 in Section 3.4.2

Overview: We leverage the classic analysis methods proposed by the authors of SybilRank. Specifically, SybilRank proposed the following three classic steps: 1) modeling the exchange of trust scores between benign region and Sybil region in each iteration, 2) modeling the trust score dynamics in the benign and Sybil regions, and 3) assuming the increased trust scores in the Sybil region all focus on a small group of Sybils. We follow these three steps to analyze residual posterior probabilities in the simplified version of SybilSCAR-D. However, one key difference is that SybilSCAR-D uses a different local rule with SybilRank, and thus the mathematical details in all the three steps are different.

Notations: We denote by $\mathcal{B}$ and $\mathcal{S}$ the set of benign nodes and Sybils, respectively. We denote by $d(\mathcal{B})$ and $d(\mathcal{S})$ the average degree of benign nodes and Sybil nodes, respectively. We denote by $|\mathcal{B}|$ and $|\mathcal{S}|$ the number of benign nodes and Sybils, respectively. For a node set $\mathcal{N}$, we denote its *volume* as the sum of degrees of nodes in $\mathcal{N}$, i.e., $Vol(\mathcal{N}) = \sum_{u \in \mathcal{N}} d_u$. Moreover, we have

$$C_{\mathcal{B}} = \frac{g}{Vol(\mathcal{B})}, \quad C_{\mathcal{S}} = \frac{g}{Vol(\mathcal{S})}, \tag{.4}$$

which were introduced by SybilRank. We denote by $\hat{P}_{\mathcal{B}}^{(t)}$ and $\hat{P}_{\mathcal{S}}^{(t)}$ the average residual posterior probability of benign nodes and Sybils in the tth iteration, respectively. Initially, $\hat{P}_{\mathcal{S}}^{(0)} = 0$ (since we do not consider labeled Sybils in the training dataset) and $\hat{P}_{\mathcal{B}}^{(0)} < 0$.

Exchange of residual posterior probabilities between benign region and Sybil region: In the $(t+1)$th iteration, the average residual posterior probability of Sybils and the average residual posterior probability of benign nodes can be approximated as follows:

$$\hat{P}_{\mathcal{S}}^{(t+1)} = C_{\mathcal{S}}\hat{P}_{\mathcal{B}}^{(t)} + (1 - C_{\mathcal{S}})\hat{P}_{\mathcal{S}}^{(t)}, \tag{.5}$$

$$\hat{P}_{\mathcal{B}}^{(t+1)} = C_{\mathcal{B}}\hat{P}_{\mathcal{S}}^{(t)} + (1 - C_{\mathcal{B}})\hat{P}_{\mathcal{B}}^{(t)}. \tag{.6}$$

We take Equation .5 as an example to illustrate how we derive the equations. In the considered version of SybilSCAR-D, in each iteration, a node's residual posterior probability is the average of its neighbors' residual posterior probabilities. Since we assume the attack edges are randomly established between benign nodes and Sybils, the total residual posterior probability propagated from the benign nodes to Sybils is $g\hat{P}_{\mathcal{B}}^{(t)}$. Moreover, the total residual posterior probability propagated within the Sybils is $(Vol(\mathcal{S}) - g)\hat{P}_{\mathcal{S}}^{(t)}$, because each edge between Sybils contributes one copy of $\hat{P}_{\mathcal{S}}^{(t)}$ on average. Since each node takes the average of its neighbors' residual posterior probabilities, each Sybil has an average residual posterior probability $\hat{P}_{\mathcal{S}}^{(t+1)}$ as $\frac{1}{|\mathcal{S}|d(\mathcal{S})}(g\hat{P}_{\mathcal{B}}^{(t)} + (Vol(\mathcal{S}) - g)\hat{P}_{\mathcal{S}}^{(t)})$, which gives us Equation .5.

Note that the derivation of Equations .5 and .6 is inspired by SybilRank Cao et al. (2012). However, since SybilSCAR-D and SybilRank use different local rules (though both are linear local rules), their exchange dynamics between benign region and Sybil region are different. This difference leads to different dynamics within benign/Sybil region and eventually leads to different security guarantees. Given Equations .5 and .6, we have:

$$\hat{P}_{\mathcal{B}}^{(t+1)} - \hat{P}_{\mathcal{S}}^{(t+1)} = (1 - C_{\mathcal{B}} - C_{\mathcal{S}})(\hat{P}_{\mathcal{B}}^{(t)} - \hat{P}_{\mathcal{S}}^{(t)}). \tag{.7}$$

Dynamics in the Sybil and benign regions: The decrease of the average residual posterior probabilities of Sybils is as follows:

$$\hat{P}_{\mathcal{S}}^{(t+1)} - \hat{P}_{\mathcal{S}}^{(t)} = C_{\mathcal{S}}(\hat{P}_{\mathcal{B}}^{(t)} - \hat{P}_{\mathcal{S}}^{(t)}) = C_{\mathcal{S}}(1 - C_{\mathcal{B}} - C_{\mathcal{S}})(\hat{P}_{\mathcal{B}}^{(t)} - \hat{P}_{\mathcal{S}}^{(t)}) = (1 - C_{\mathcal{B}} - C_{\mathcal{S}})^t C_{\mathcal{S}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}), \tag{.8}$$

where the above equation is negative (so we call it a decrease) because $(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)})$ is negative. Therefore, we have:

$$\hat{P}_{\mathcal{S}}^{(t)} - \hat{P}_{\mathcal{S}}^{(0)} = \sum_{i=0}^{t-1}(1 - C_{\mathcal{B}} - C_{\mathcal{S}})^t \times C_{\mathcal{S}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}). \tag{.9}$$

Similarly, the increase of the average residual posterior probabilities of benign nodes is as follows:

$$\hat{P}_{\mathcal{B}}^{(t+1)} - \hat{P}_{\mathcal{B}}^{(t)} = -C_{\mathcal{B}}(\hat{P}_{\mathcal{B}}^{(t)} - \hat{P}_{\mathcal{S}}^{(t)}) = -(1 - C_{\mathcal{B}} - C_{\mathcal{S}})^t C_{\mathcal{B}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}) = -(1 - C_{\mathcal{B}} - C_{\mathcal{S}})^t \times C_{\mathcal{B}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}), \tag{.10}$$

where the above equation is positive (so we call it an increase) because $(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)})$ is negative. Furthermore, we have:

$$\hat{P}_{\mathcal{B}}^{(t)} - \hat{P}_{\mathcal{B}}^{(0)} = -\sum_{i=0}^{t-1}(1 - C_{\mathcal{B}} - C_{\mathcal{S}})^t \times C_{\mathcal{B}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}). \tag{.11}$$

Security guarantee: We assume after $\Omega = O(\log |V|)$ iterations, benign nodes have similar residual posterior probabilities, which are the average residual posterior probability of benign nodes. We note that SybilRank relies on a similar assumption, i.e., after $O(\log |V|)$ iterations, benign nodes have similar degree-normalized trust scores. We assume the decrease of residual posterior probabilities of Sybils all focus on $n_{\mathcal{S}}$ Sybils, which gives an upper bound of Sybils whose residual posterior probabilities are smaller than benign nodes. If we want these Sybils to have residual posterior probabilities that are smaller than benign nodes, then we have:

$$0 - \frac{(\hat{P}_{\mathcal{S}}^{(0)} - \hat{P}_{\mathcal{S}}^{(\Omega)})|\mathcal{S}|}{n_{\mathcal{S}}} < \hat{P}_{\mathcal{B}}^{(\Omega)} \iff n_{\mathcal{S}} < \frac{(\hat{P}_{\mathcal{S}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)})|\mathcal{S}|}{\hat{P}_{\mathcal{B}}^{(\Omega)} - 0} \iff n_{\mathcal{S}} < \frac{(\hat{P}_{\mathcal{S}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)})|\mathcal{S}|}{\hat{P}_{\mathcal{B}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)}}, \tag{.12}$$

where the last step holds because $\hat{P}_{\mathcal{S}}^{(0)} = 0$. Moreover, we have:

$$\begin{aligned}
\frac{(\hat{P}_{\mathcal{S}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)})|\mathcal{S}|}{\hat{P}_{\mathcal{B}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)}} &= \frac{(\hat{P}_{\mathcal{S}}^{(\Omega)} - \hat{P}_{\mathcal{S}}^{(0)})|\mathcal{S}|}{\hat{P}_{\mathcal{B}}^{(\Omega)} - \hat{P}_{\mathcal{B}}^{(0)} + \hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)}} && (.13)\\
&= \frac{\sum_{0\le t\le(\Omega-1)}(1-C_{\mathcal{S}}-C_{\mathcal{B}})^t C_{\mathcal{S}}(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)})|\mathcal{S}|}{(1-\sum_{0\le t\le(\Omega-1)}(1-C_{\mathcal{S}}-C_{\mathcal{B}})^t C_{\mathcal{B}})(\hat{P}_{\mathcal{B}}^{(0)} - \hat{P}_{\mathcal{S}}^{(0)})} && (.14)\\
&< \frac{\sum_{0\le t\le(\Omega-1)}(1-C_{\mathcal{S}})^t C_{\mathcal{S}}|\mathcal{S}|}{(1-\sum_{0\le t\le(\Omega-1)}(1-C_{\mathcal{S}}-C_{\mathcal{B}})^t C_{\mathcal{B}})} && (.15)\\
&= \frac{(1-(1-C_{\mathcal{S}})^{\Omega})|\mathcal{S}|}{1-\frac{1-(1-C_{\mathcal{S}}-C_{\mathcal{B}})^{\Omega}}{C_{\mathcal{S}}+C_{\mathcal{B}}}C_{\mathcal{B}}} && (.16)\\
&\approx \frac{C_{\mathcal{S}}\Omega|\mathcal{S}|}{1-\frac{1-(1-(C_{\mathcal{S}}+C_{\mathcal{B}})\Omega+\frac{\Omega(\Omega-1)}{2}(C_{\mathcal{S}}+C_{\mathcal{B}})^2)}{C_{\mathcal{S}}+C_{\mathcal{B}}}C_{\mathcal{B}}} && (.17)\\
&\approx \frac{g\Omega}{d(\mathcal{S})(1-(\Omega-\frac{\Omega^2}{2}(C_{\mathcal{S}}+C_{\mathcal{B}}))C_{\mathcal{B}})} && (.18)\\
&< \frac{g\Omega}{d(\mathcal{S})(1-(\Omega-\frac{\Omega^2}{2}(C_{\mathcal{S}}+C_{\mathcal{B}}))(C_{\mathcal{S}}+C_{\mathcal{B}}))} && (.19)\\
&= \frac{g\Omega}{d(\mathcal{S})(\frac{1}{2}+\frac{1}{2}(\Omega(C_{\mathcal{S}}+C_{\mathcal{B}})-1)^2)} && (.20)\\
&\le \frac{2g\Omega}{d(\mathcal{S})}, && (.21)
\end{aligned}$$

where in Equation .17, we use the first-order and second-order Taylor expansion on nominator and denominator, respectively; In Equation .18, we use $C_{\mathcal{S}} = \frac{g}{Vol(\mathcal{S})}$, $Vol(\mathcal{S}) = d(\mathcal{S})|\mathcal{S}|$, and $\Omega(\Omega-1)(C_{\mathcal{S}}+C_{\mathcal{B}}) \approx \Omega^2(C_{\mathcal{S}}+C_{\mathcal{B}})$. By setting $\Omega = O(\log|V|)$, we have:

$$n_{\mathcal{S}} = O\Big(\frac{g\log|V|}{d(\mathcal{S})}\Big). \tag{.22}$$

Proof of Theorem 10 in Section 4.4.2

Our idea is to first linearize Equation 4.12 and then combine the linearized version with Equation 4.16. Optimizing LBP for a standard pMRF on undirected graphs in Jia et al. Jia et al. (2017b) also involves linearizing Equation 4.12. In particular, they linearized Equation 4.12 as follows:

$$\hat{p}_u^{(t)} = \hat{q}_u + \sum_{v\in\Gamma(u)} \hat{m}_{vu}^{(t)}. \tag{.23}$$

Next, we combine Equation .23 with Equation 4.16 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 .23 as follows:

$$\hat{p}_u^{(t)} = \hat{q}_u + \sum_{v \in \Gamma_b(u)} \hat{m}_{vu}^{(t)} + \sum_{v \in \Gamma_i(u)} \hat{m}_{vu}^{(t)} + \sum_{v \in \Gamma_o(u)} \hat{m}_{vu}^{(t)}. \tag{.24}$$

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 9. 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 4.17 via combining Equation .24 with Equation 4.16.

Proof of Theorem 11 in Section 4.5

$\mathbf{A}_i'^{(t)}$ 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:

$$\begin{aligned}
&\rho(2\hat{w}(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)})) < 1 \\
&\Longleftarrow \|2\hat{w}(\mathbf{A}_b + \mathbf{A}_i'^{(t)} + \mathbf{A}_o'^{(t)})\|_1 < 1 \\
&\Longleftarrow 2\hat{w}\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1 < 1 \\
&\Longleftarrow \hat{w} < \frac{1}{2\|\mathbf{A}_b + \mathbf{A}_i + \mathbf{A}_o\|_1}.
\end{aligned}$$

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