More Efficient Sybil Detection Mechanisms Leveraging Resistance of Users to Attack Requests*

Ali Safarpoor Dehkordi
Australian National University
Canberra, Australia
ali.safarpoordehkordi@anu.edu.au

Ahad N. Zehmakan
Australian National University
Canberra, Australia
ahadn.zehmakan@anu.edu.au

ABSTRACT

We investigate the problem of sybil (fake account) detection in social networks from a graph algorithms perspective, where graph structural information is used to classify users as sybil and benign. We introduce the novel notion of user resistance to attack requests (friendship requests from sybil accounts). Building on this notion, we propose a synthetic graph data generation framework that supports various attack strategies. We then study the optimization problem where we are allowed to reveal the resistance of a subset of users with the aim to maximize the number of users which are discovered to be benign and the number of potential attack edges (connections from a sybil to a benign user). Furthermore, we devise efficient algorithms for this problem and investigate their theoretical guarantees. Finally, through a large set of experiments, we demonstrate that our proposed algorithms improve detection performance notably when applied as a preprocessing step for different sybil detection algorithms. The code and data used in this work are publicly available on GitHub.[1]

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Human-centered computing → Social networks**.

KEYWORDS

Sybil Detection, Social Networks, Algorithmic Graph Data Mining

ACM Reference Format:
Ali Safarpoor Dehkordi and Ahad N. Zehmakan. 2025. More Efficient Sybil Detection Mechanisms Leveraging Resistance of Users to Attack Requests. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Motivation. Online social networks (OSNs) like Facebook, Instagram, and Twitter (now X) have become essential parts of our lives. They are places where we connect with friends and family, get the latest news, plan events, and even conduct business. Thus, they have revolutionized many fundamental aspects of our lives.

However, the rapid growth of OSNs has raised privacy and security concerns as malicious entities exploit them for identity theft and spreading harmful content. A key element is sybils (also referred to as "fake", "malicious", and "fraudulent" accounts), used for spamming, malware, and phishing, causing financial and reputational damage [1, 32]. Thus, detecting sybils is crucial for a safer online environment.

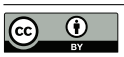
Early Detection Mechanisms. Detecting sybils can be conducted in different stages of the formation. The first stage is during registration, by using data like IP, Captcha, and location [26, 45], The next stage to detect sybils that pass through the registration is based on their first actions after joining the network, such as the ratio of accepted friend requests, the randomness of the requests sent, or the total number of requests, cf. [8, 9].

Late Detection Mechanisms. While many sybils can be caught using early detection mechanisms, more sophisticated attackers can bypass these by mimicking benign users. Numerous studies have focused on identifying these hard-to-find accounts. Some research analyzes the content provided by users to detect fake or malicious activities, cf. [33]. Other studies [13, 24] examine user profiles, considering factors such as profile pictures, the number of followers and followings, or the volume of activities like the number of posts. While these methods are more in-depth than the early detection mechanisms, they may still fail to identify sybils that sophisticatedly mimic benign users' behavior.

Graph Based Detection Mechanisms. While sybils have a lot of power over manipulating information such as their IP, username, number of posts, and profile pictures, which are used by the aforementioned methods, they have very little control over their position with respect to the overall network structure. Thus, a natural question arises: can network structural information be used to detect sophisticated sybil attacks? More precisely, suppose we are given a graph G, corresponding to a social network, where some nodes (users) are labeled sybil or benign. The goal is to label the remaining nodes as accurately as possible. Prior work has leveraged various methods such as random walk [21], belief propagation [39], and graph neural networks (GNN) [41].

Shortcomings of Exiting Methods. The existing algorithms [28, 38, 44] are usually designed under the *homophily* assumption and very limited number of attack edges, that is, most edges (links) in the graph are between nodes of the same type (sybil-to-sybil or benign-to-benign) with significantly fewer edges in between (sybil-to-benign or benign-to-sybil). Due to the lack of real-world data, these algorithms are usually tested on synthetic graph models, which are tailored to possess such homophily properties. However, based on current research [40], we know that the homophily property is not satisfied in many real-world examples.

*The full version of the paper arxived in [12].
[1]GitHub repository: https://github.com/aSafarpoor/AAMAS2025-Paper/tree/main



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Missing Piece of Puzzle. One key aspect that steers the formation of connection in a network is how benigns react to friendship requests from sybils. We introduce the notion of user resistance, where a resistant user rejects such requests while a non-resistant accepts them. (For simplicity, we suppose a user is either resistant or non-resistant, but our results can be extended to the setup where nodes have various degrees of resistance.) The notion of resistance permits us to design a more dynamic graph generation framework where, unlike prior static models [16, 43], the outcome graph is a function of the attack strategy and resistance of users rather than a simplified pre-assumption such as homophily. We then devise effective and efficient mechanisms to leverage resistance information for finding potential attack edges and benigns, which serve as helpful preprocessing steps for sybil detection algorithms.

Attack Models. It is commonly assumed, cf. [28], that an attacker connects its created sybil accounts together in a fashion that it mimics benigns, for example, by copying the connections between a set of benigns with the same size.

The vital question is how the connections between sybils and the rest of the network are formed. Prior models [39] implant these connections following a presumed structural property such as homophily. However, in reality, these connections are a function of two variables: (i) the attacker's strategy to send connection requests and (ii) whether a request that has been sent is accepted by the corresponding benign, which is determined by their resistance. Following this natural observation, we introduce a novel generic modeling framework and three potential attack strategies. This creates a powerful test bed for sybil detection mechanisms.

Potential Attack Edges. As mentioned, most sybil detection algorithms function well when the homophily property holds. Still, as observed in [16], this is often not the case, especially when the attacker has a powerful strategy and users are non-resistant. Thus, it would be beneficial to determine potential attack edges (sybil-to-benign connections). Then, they can be removed or receive less weights to enhance the homophily and consequently the performance of sybil detection algorithms.

Incoming edges for non-resistant benigns are *potential* attack edges. The issue is that, in reality, we don't know which nodes are resistant/non-resistant. However, we might have an estimate of whether a node is resistant (e.g., based on their number of connections, interaction behaviors, and so on). Furthermore, we can determine whether a user is resistant by sending a sequence of requests from a dummy sybil account. However, we don't want to bombard the whole network with such dummy sybil requests. Thus, we suppose we have a fixed budget k of the number of users whose resistance can be revealed. We formulate this as an optimization problem and provide an optimal linear time algorithm.

Discovering Benigns. If we know that a node v is benign and resistant, we can conclude that a node u with an edge *to* v is also benign. Now, if u is known to be resistant, we can conclude that a node w, which has an edge to u, is benign too, and so on. How many new benigns can be discovered if we are allowed to reveal the resistance of k nodes? We prove that this problem is computationally "hard". However, we propose a greedy-based and traversing algorithm, which turns out to be very performant based on experiments on real-world graph data. In addition to potential attack edge discovery from above, expanding the benign set can serve as an important preprocessing step for sybil detection algorithms.

Enhancing Detection Algorithms by Preprocessing. Based on the contributions mentioned above, we investigate the state-of-the-art detection methods, including SybilSCAR [39], SybilWalk [21], and SybilMetric [2], with and without preprocessing steps conducted by our resistance-based mechanisms. We first examine the performance of these methods on several real-world social networks incorporated into our synthetic framework under three different attack strategies. We then gauge the performance when our potential attack edge and benigns discovery mechanisms are applied as a preprocessing step. They prove to be very impactful in enhancing the accuracy performance.

Outline. In the rest of this section, we provide some basic definitions and an overview of some prior work. Our data generation framework, accompanied by attack models, is provided in Section 2. The maximizing benigns and potential attack edges discovery mechanisms using resistance information are given in Section 3.1 and 3.2. Finally, the experimental results of our proposed mechanisms are discussed in Section 4.

1.1 Preliminaries

Graph Definition. A social network is represented by a directed graph $G = (V, E)$ with node set V and edge set $E \subseteq V \times V$. We have $|V| = n$ and $|E| = m$. Nodes correspond to users; edges represent connections, such as following (directed) or friendship (bidirectional).

We define $\Gamma_{\text{in}}(v) := \{u : (u, v) \in E\}$ and $\Gamma_{\text{out}}(v) := \{u : (v, u) \in E\}$ to be respectively the set of incoming and outgoing neighbors of a node v. Then, we define $d_{\text{in}}(v) := |\Gamma_{\text{in}}(v)|$, $d_{\text{out}}(v) := |\Gamma_{\text{out}}(v)|$ and $d(v) := d_{\text{in}}(v) + d_{\text{out}}(v)$. Let $\Delta_{\text{in}} = \max\{d_{\text{in}}(v) \mid v \in V\}$ denote the maximum incoming degree and $\Delta_{\text{out}} = \max\{d_{\text{out}}(v) \mid v \in V\}$. We show an edge from u to v by e_{uv}, and if the edge is undirected, we show that by $\bar{e}_{uv}$. In addition, for two subsets $V_1, V_2 \subseteq V$ we define $\partial(V_1, V_2) := \{e_{vu} \in E : v \in V_1 \wedge u \in V_2\}$. A path in a graph $G = (V, E)$ is a sequence of nodes $v_1, v_2, \ldots, v_k$ where each adjacent pair is connected by an edge, i.e., $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k-1$.

Benign and Sybil Classification (BSC) Problem
Input: Given a graph $G = (V, E)$ where nodes are labeled as *Benign*, *Sybil*, or *Unknown*, partitioned into subsets B, S, and U respectively ($B \cup S \cup U = V$ and $(B \cap S) \cup (B \cap U) \cup (S \cap U) = \emptyset$).
Goal: Maximize the number of correctly labeled nodes in U.

User Resistance. In the real world, each user can choose to accept or reject sybil requests. Some users are more cautious and reject such requests, while others might accept them. For simplicity, we assume each user, which is represented as a node in a graph, either accepts all sybil requests or rejects all of them. To model this, we define a binary value $r : v \to \{0, 1\}$ to represent a user's resistance to sybil requests, where 1 means resistant (rejecting) and 0 means non-resistant (accepting). Since, in reality, the value of $r(v)$ is not known to us, we suppose we know a probability function $p_r(v)$ whose value is more likely to be closer to 1 if v is resistant and 0 otherwise. Such estimates can be achieved in practice by studying certain user features, such as the number of incoming/outgoing connections, the number of known sybil/benign

neighbors, and content posted. In our paper, we suppose such a probability distribution is given to us as input.

1.2 Related Work

Some mechanisms aim to detect sybils at the early stages. This could be as early as registration time using techniques like captchas or analyzing data such as registration time and IP addresses, cf. [26, 45]. The sybils that pass through these filters might be caught based on their first activities, such as friendship requests, cf. [9].

While such early detection mechanisms are useful in identifying a considerable fraction of sybils, more sophisticated attacks can deceive them by mimicking benign users. Thus, there has been a growing interest in leveraging network structure information to detect more sophisticated attacks, cf. [21, 42]. This is because, unlike information such as username or IP, sybil users have very limited power to control their network structural properties. Below, we give an overview of various existing methods.

Graph Metrics. Graph metrics can help distinguish sybils from benign nodes. Asghari et al.[2] analyzed metrics like degree, betweenness, eigenvector centrality, clustering coefficient, and shortest path length. Yoon[42] introduced graph accessibility for sybil detection. Jethava and Rao [20] combined user behavior with graph metrics like the Jaccard index and betweenness centrality.

Random Walk. Random walk-based detection mechanisms have particularly become popular, cf. [7, 46]. They usually rely on the premise that a random walk starting from a sybil user is more likely to encounter sybil users. SybilGuard [44] was one of the first methods to leverage random walks for sybil detection. SybilLimit [43] improved on this by enhancing scalability and providing a tighter bound on the number of accepted sybils. SybilInfer [11] additionally used Bayesian inference and Monte Carlo simulations to provide a more robust mechanism. SybilRank [10] proposed using a ranking algorithm that prioritizes nodes based on the degree-normalized probability of a short random walk from a non-sybil landing on them. SybilWalk [21] improved random walk-based models by incorporating known benigns and sybils simultaneously. This model defines two extra nodes as label nodes and then connects known nodes to their respective label node.

Belief Propagation. Another popular approach, cf. [31], is to assign some initial probability of being sybil to each node (where known sybils get higher and known benigns get lower probabilities) and then use some updating mechanism, called belief propagation, to improve these probabilities. The idea is that a node's probability (belief) is updated as a function of the probabilities of its neighbors. SybilBelief [16] is a semi-supervised learning algorithm that uses belief propagation to detect sybils. SybilSCAR [39] unified random walk and belief propagation methods, applying local rules iteratively to identify sybils. GANG [38] improved previous models by using directed graphs. SybilFuse [15] employed collective classification by training local classifiers to calculate trust scores first, then propagating these scores using weighted random walk and belief propagation to improve detection accuracy.

Machine Learning. Machine Learning (ML) methods for sybil detection usually involve feature extraction, model training, and classification. Some basic ML methods include SVM, Logistic Regression, and Random Forest, cf.[23, 24, 35]. Furthermore, deep learning methods can process raw data and capture complex structures, making them ideal for sybil detection. Goyal et al. [17] used graph convolutional network (GCN) to learn structural features, while Borkar et al. [6] employed recurrent networks for text content processing, followed by clustering for sybil detection. Recent advances leverage graph neural networks (GNNs) and attention mechanisms. Yang and Zheng [41] used attention-based GNNs, Khan et al.[22] developed a GNN-based framework to analyze user profiles and connections, and Liu et al.[27] integrated diverse attributes using heterogeneous GNNs.

2 ATTACK MODELS

To effectively evaluate detection algorithms, it is essential to provide high-quality datasets. Many existing datasets suffer from a lack of proper labeling, as they are often labeled manually and are quite limited in size. To tackle this issue, some synthesized datasets have been provided [16, 28] but they still fall short in various aspects. For example, [16] uses a synthesized model to generate the sybil area, which is less favorable than using real-world data. Furthermore, the existing models [28] make special assumptions on the strategy employed by the attacker and the structure of the edges between sybil and benign parts, for example, uniform random edges. They then leverage these structural properties to devise algorithms that are tailored for this setting. However, the proposed algorithms should ideally be agnostic of the attack strategy deployed by the attacker since the attacker can adopt various attack strategies. Moreover, the availability of diverse datasets generated by various attack strategies enhances the reliability of evaluation outcomes. This encourages us to synthesize datasets with different characteristics. We will see that the notion of user resistance, defined in this paper, forms an excellent foundation for synthesizing realistic datasets.

To synthesize a labeled network, one needs to define the nodes and their labels and the edge set, especially *attack edges*, formed from sybils to benigns. An existing dataset of online social networks is usually used as the benign set, treating all individuals as benign [39]. For the sybil set, a common approach is that the attacker may replicate a subgraph from the real graph to secure an initial real-world structure. If attackers avoid using this strategy, their presence becomes readily detectable through the analysis of simple graph features.

Following the above description, the sybil set S will be created based on the chosen subset $B' \subset B$, where B is the benign set. Then, the edges between nodes in S are formed to copy the corresponding structure in B'. Each node $v \in B'$ has a corresponding copy in S and vice versa; these nodes are referred to as *dual* of each other.

Following a specific strategy, the attacker sends requests to the benigns. If a benign has resistance 1, the request is rejected; otherwise, it is accepted. Recall that we show the resistance of each node v with $r(v)$, which is set to 0 (non-resistant) or 1 (resistant). For example, if the resistance of all nodes is set to 1, all the sent requests are rejected, and no edge is formed from S to B. However, in reality, not all nodes are resistant.

As stated, our proposed framework is not limited to a particular attack strategy and facilitates the study of various strategies. Below, we provide three natural approaches which may be used by the

attacker. Before that, we explain some concepts which are shared among them.

Assume the number of **A**ttack **E**dges from node $s_i \in S$ to B is $AE(i)$. As discussed above, we suppose that the attacker "copies" the subgraph among a subset of benigns $B' \subset B$ as S. To mimic a benign behavior further, one natural strategy is to ensure that each node in S has as many edges to $B \setminus B'$ as its dual. More precisely, one aims to have $AE(i) = |\partial(\{dual(s_i)\}, B \setminus B')|$.

Furthermore, it is plausible to consider adding edges from the benign region (B) to the sybil region (S). Given the low likelihood that real users spontaneously connect to sybils, we posit that only nodes that have already accepted an edge from a sybil user might reciprocally connect back to it. Formally, consider each edge $(u, v) \in E$ where $u \in S$ and $v \in B$. We add the reverse edge (v, u) with a certain probability, say 1/2, for each such edge.

Below, we describe our proposed attack strategies. Please refer to the full version [12] for a detailed pseudocode.

Random Attack Strategy. In this strategy, the attacker simply sends random requests to benigns. The number of requests from each node $i \in S$ is $c \cdot AE(i)$. For an appropriate choice of constant c, as a function of the average resistance in B, the expected number of attack edges from i will be our desired value $AE(i) = |\partial(\{dual(s_i)\}, B \setminus B')|$ (the same applied to the strategies below).

Preferential Attachment Attack Strategy. The attacker might prefer to target nodes with the least resistance, especially those that have accepted more attack requests in the past. This also aligns with the *power-law* behavior observed in real-world social networks. Thus, we leverage the preferential attachment model of network growth. We use a modified version of the preferential attachment model based on the Barabási-Albert (BA) graph [3], named *Modified BA*. This takes into account both the original degree of each node and the number of accepted attack requests. This modification has been applied to take into account that the attacker is more likely to send requests to nodes that have already proven vulnerable (by accepting previous sybil requests).

BFS Attack Strategy. To enhance the difficulty of detecting sybil behavior, another natural approach is that each sybil attempts connections with the neighbors of its dual in the benign region. The acceptance of these connection requests, however, depends on the targeted node's resistance. To achieve the desired number of attack edges for each sybil node s_i (that is, the aforementioned value of $AE(i)$), we process nodes using a BFS algorithm. For each node s_i, the BFS starts from $dual(s_i)$. If the number of nodes that BFS can reach is not enough, a preferential attachment attack strategy is utilized to complete the process.

3 PREPROCESSING ALGORITHMS

In this part of the paper, we will study the problems of Maximizing Benigns (in Section 3.1) and Discovering Potential Attack Edges (in Section 3.2) leveraging the resistance probability information. We propose algorithms for solving these problems. As will be discussed in Section 4, the outcome of these algorithms serves as a very valuable preprocessing step for detection algorithms.

3.1 Maximizing Benigns

Maximizing Benigns (MB) Problem

Input: Given $G = (V, E)$, sets of benigns and sybils B and S such that $B \cap S = \emptyset$, budget k, and resistant probability distribution $p_r : V \to [0, 1]$.

Goal: Maximize the number of newly discovered benigns by revealing the resistance of k nodes.

An algorithm is permitted to reveal the resistance of nodes in a node set A of size k as a reveal set. Each selected node u will be revealed to be resistant ($r(u) = 1$) or not, with probability $p_r(u)$, independently. (In real life, we send a set of friend requests from some dummy sybils to a node u and determine whether it is resistant, but our budget has a bound to avoid bombarding all users with such requests.) Then, the goal is to maximize the number of newly discovered benigns.

Observation 3.1. *A node u in $V \setminus (B \cup S)$ can be newly discovered benign if and only if it has a path to a node in B and each node v on that path (except, potentially, u itself) has revealed to have $r(v) = 1$.*

Definition 3.1. *$f(A)$ is the expected number of discovered benigns upon revealing the nodes in the reveal set A.*

3.1.1 Hardness Results. We prove that our problem is computationally hard by a reduction from the Maximum Coverage Problem.

Definition 3.2 (Maximum Coverage (MC) Problem). *Given set $W = \{w_1, w_2, \cdots, w_h\}$, collection of subsets $Q = \{q_1, q_2, \cdots, q_l\}$, $\forall q_i \in Q : q_i \subseteq W$, and budget k, what is the maximum number of elements which can be covered by a set $A \subset Q$ of size k? We say a set q_i covers an element w_j if $w_j \in q_i$.*

Theorem 3.1 ([14]). *There is no $(1-\frac{1}{e})$-approximation polynomial time algorithm for the MC problem unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).*

Assume $I = \langle W, Q, k\rangle$ is an instance of MC. We define a transformation to convert I into $I' = \langle G, B, S, k, p_r(\cdot)\rangle$, which is an instance of MB. Without loss of generality, we assume that each $w_i \in W$ appears in at least one of the subsets $q_j \in Q$ (otherwise, they could be simply ignored because they cannot be covered at all). Now, we present the transformer in the following.

Transformer. To define the transformer to convert I to I', we first need to construct the MC problem in graph space as $G = (V, E)$. We define $V = V_Q \cup V_W$, where $V_Q = \{v_{q_i} : 1 \le i \le l\}$ and $V_W = \{v_{w_i} : 1 \le i \le h\}$. Note that $V_Q \cap V_W = \emptyset$. We also define $E = \{(v_{w_i}, v_{q_j}) : w_i \in q_j\}$. In addition, k is the same, $S = \emptyset$, $B = V_Q$, and $p_r(v) = 1$ for all nodes (which means we know all nodes are resistant). An example is provided in the full version [12].

Connection Between Optimal Solutions. For any arbitrary instance of the MC problem, let OPT_{MC} denote the optimal solution, and similarly OPT_{MB} for the MB problem. We claim that

$$OPT_{\text{MC}} = OPT_{\text{MB}} \tag{1}$$

First, let A be an optimal solution for the MB problem. Note that $A \cap V_W = \emptyset$ because revealing a node in V_W does not result in discovering any new benign. Consider that set A_q, which includes a set q_i if and only if $v_{q_i} \in A$. If a node v_{w_j} is newly discovered benign by revealing A, then there is a node v_{q_i} such that $(v_{w_j}, v_{q_i}) \in E$ and $v_{q_i} \in A$. This implies that $w_j \in q_i$ and $q_i \in A_q$. Therefore, w_j is covered by A_q. This implies that $OPT_{\text{MC}} \ge OPT_{\text{MB}}$.

It remains to prove that $OPT_{\text{MC}} \leq OPT_{\text{MB}}$. Let $A_q \subset Q$ generate an optimal solution for the MC problem. Define A to include a node v_{q_i} if and only if $q_i \in A_q$. Consider an element w_j, which is covered by A_q, and let q_i be a set which covers it. Then, $v_{q_i} \in A$ and $(v_{w_j}, v_{q_i}) \in E$. Thus, w_j will be a newly discovered benign since $w_j \in V \setminus B$, and it has an edge to a node with resistance 1. This implies that $OPT_{\text{MC}} \leq OPT_{\text{MB}}$.

Inapproximability. Assume there exists a polynomial-time $(1-\frac{1}{e})$-approximation algorithm Alg_{MB}, that solves the MB problem. For an instance of the MC problem, we can use the previously mentioned transformer to construct an instance of our problem in polynomial time. Then, we apply Alg_{MB} to solve the constructed instance, yielding a solution Sol_{MB}. Using the same argument as previously described for the relationship between optimal solutions, we can show that $Sol_{\text{MC}} \geq Sol_{\text{MB}}$. Thus, $Sol_{\text{MC}} \geq Sol_{\text{MB}} \geq (1 - \frac{1}{e})OPT_{\text{MB}}$. With the help of Equation (1) we can conclude $Sol_{\text{MC}} \geq (1 - \frac{1}{e})OPT_{\text{MC}}$

Therefore, we get an $(1 - \frac{1}{e})$-approximation algorithm for MC, which operates in polynomial time. However, according to Theorem 3.1, this is impossible unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$. Thus, we have the theorem below.

Theorem 3.2. *There is no $(1 - \frac{1}{e})$-approximation polynomial time algorithm for the MB problem unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$.*

3.1.2 Greedy Approach. For the MB problem, we can use the classical greedy algorithm, which iteratively reveals the node that maximizes the expected number of newly discovered benigns (an exact description is given in the full version [12]). According to Nemhauser et al. [30], if $f(\cdot)$ is non-negative, monotone and submodular, then this algorithm is $(1 - \frac{1}{e})$-approximation (which matches the bound from Theorem 3.2). However, we prove that the objective function $f(\cdot)$, in fact, is *not* submodular (please see the full version [12] for a proof). Thus, we cannot conclude $(1 - \frac{1}{e})$-approximation guarantee. On the positive side, the greedy algorithm has proven to be performant, cf. [4, 18], even if the submodularity property doesn't hold. Thus, we study this further. However, it turns out that computing $f(\cdot)$ is #P-hard, implying that the greedy algorithm is not polynomial time. To tackle this issue, we show that we can estimate $f(\cdot)$ with an arbitrarily small error parameter using the Monte Carlo method.

#P-hardness. We rely on a reduction from *a-b* Connectedness for Induced Subgraphs Problem, which is known to be #P-hard [34].
a-b Connectedness for Induced Subgraphs (CIS):
Input: Given a directed graph $G = (V, E)$ and $a, b \in V$.
Goal: What is the number of induced subgraphs of G such that there is a path from a to b.

Theorem 3.3. *Computing $f(\cdot)$ is #P-hard.*

Proof. Suppose $G = (V, E)$ and $a, b \in V$ are given as the input of CIS problem. We consider two instances of our problem to compute $f(\cdot)$. First consider graph $G = (V, E)$, $B = \{b\}$, $S = \emptyset$, $\begin{cases} r(v) = \frac{1}{2} & v \in V \setminus \{a, b\} \\ r(v) = 1 & v \in \{a, b\} \end{cases}$, and reveal set $A = V$. The second case is identical to the first case, except we add a node a' and add an edge from a' to a. Let's call this graph $G' = (V', E')$, where $V' = V \cup \{a'\}$. Furthermore, again $B = \{b\}$, $S = \emptyset$, the values of $r(v)$ are the same for all nodes, and $r(a')$ is any arbitrary value (its choice doesn't impact our argument), and $A = \{V\}$. See Figure 1 for a visualization.

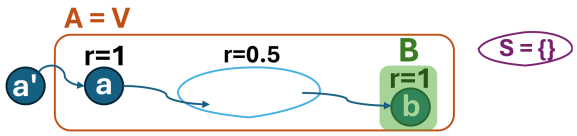


Figure 1: The construction used in the proof of #P-hardness.

Let $f_{-a'}(A)$ and $f_{+a'}(A)$ denote the value of $f(A)$, the expected number of discovered benigns, in the first and second case, respectively. Firstly, we observe that a' is discovered to be benign if and only if a is discovered to be benign. This implies that $f_{+a'}(A) - f_{-a'}(A)$ is equal to the probability of the node a being discovered to be benign. Secondly, note that each of the nodes in $V \setminus \{a, b\}$ is revealed to be resistant or not with probability $1/2$, independently. Thus, based on Observation 3.1, the probability that a is revealed to be benign is the same as having a path from a to b once we remove each node in $V \setminus \{a, b\}$ with probability $1/2$. Combining the aforementioned two points, we can conclude that $f_{+a'}(A) - f_{-a'}(A)$ is equal to the fraction of all 2^{n-2} possible induced subgraphs obtained from removing nodes in $V \setminus \{a, b\}$ in graph G, where there is a path from a to b.

So far we concluded that the solution to CIS problem is equal to $(f_{+a'}(A) - f_{-a'}(A)) \times 2^{n-2}$. Since the provided transformer is polynomial time and CIS is #P-hard, we conclude that the problem of computing $f(\cdot)$ is #P-hard as well. □

Monte Carlo Greedy Algorithm. Algorithm 1 modifies the classical greedy approach by using the Monte Carlo method to estimate $f(\cdot)$ since, as we proved, its computation is #P-hard. The function $\texttt{EST}(\cdot)$ runs the revealing process R times and computes the number of discovered benigns each time. Then, the average values found are returned as an estimation of the original expectation.

As part of the function $\texttt{EST}(\cdot)$, to compute the number of discovered benigns based on $r(\cdot)$ and the reveal set A, we apply a BFS starting from known benigns in A. During the BFS, only nodes with resistance 1 (starting from A) are added to the queue, marking their neighbors as discovered. This ensures every discovered node has a path of nodes with resistance 1 to an initial benign, meeting the criterion in Observation 3.1. The pseudocode of this algorithm is provided in the full version [12].

Furthermore, using Hoeffding's inequality [19], one can achieve arbitrary error margin ϵ and gain confidence α by setting the number of iterations $R \geq \frac{k^2 \Delta_{\text{in}}^2 \ln(\frac{1}{1-\alpha})}{2\epsilon^2}$. The time complexity of the algorithm is $O(k \cdot R \cdot (n^2 + nm))$. One can check that, for constant ϵ and α, the run time is polynomial. Please refer to the full version [12] for more details on these two points.

3.1.3 Proposed Traversing Algorithm. We propose an algorithm that iteratively selects the node expected to reveal the maximum number of newly discovered benigns. More precisely, we keep a set of nodes N such that if a node in N is revealed to be resistant, then we can conclude all its incoming neighbors are benign. Based on Observation 3.1, these nodes already have a path to a node in B, and all nodes on the path are revealed to be benign. Thus, initially, N is

Algorithm 1 Monte Carlo Greedy Algorithm

Input $G = (V, E)$, B, S, budget k, $p_r(\cdot)$, ϵ, α
Output *reveal_set* A

Initialize $A \leftarrow \emptyset$
for $i = 1$ to k **do**
  $v \leftarrow \arg\max_{w \in V \setminus A} \textsc{Est}(A \cup \{w\}, p_r(\cdot), B, S, \epsilon, \alpha)$
  $A \leftarrow A \cup \{v\}$
return A

function $\textsc{Est}(A, p_r(\cdot), B, S, \epsilon, \alpha)$
  Initialize $count \leftarrow 0$
  $R \leftarrow \left\lceil \frac{k^2 \Delta_{\text{in}}^2 \ln(\frac{1}{1-\alpha})}{2\epsilon^2} \right\rceil$
  for R iterations **do**
    Assign resistance $r(v)$ for each $v \in A$ based on $p_r(v)$
    $new\_benign \leftarrow$ number of discovered benigns based on B, S, and assigned resistances.
    $count \leftarrow count + new\_benign$
  return $count/R$

Algorithm 2 Traversing Algorithm

Input $G = (V, E)$, B, S, budget k, $p_r(\cdot)$
Output *reveal_set* A

1 $A \leftarrow \emptyset$
2 $N \leftarrow B$
3 **for** $v \in V$ **do**
4 $\hat{\Gamma}_{\text{in}}(v) \leftarrow \Gamma_{\text{in}}(v) \setminus (B \cup S)$
5 $\gamma_{\text{in}}(v) = |\hat{\Gamma}_{\text{in}}(v)|$
6 **while** $|A| < k$ **do**
7 pick v with highest $p_r(v) \cdot \gamma_{\text{in}}(v)$ between nodes in N
8 $A \leftarrow A \cup \{v\}$
9 $N \leftarrow N \setminus \{v\}$
10 **if** $r(v) = 1$ **then**
11 $N \leftarrow N \cup \hat{\Gamma}_{\text{in}}(v)$
12 **for** $u \in \hat{\Gamma}_{\text{in}}(v)$ **do**
13 **for** $w \in \Gamma_{\text{out}}(u)$ **do**
14 $\hat{\Gamma}_{\text{in}}(w) \leftarrow \hat{\Gamma}_{\text{in}}(w) \setminus \{u\}$
15 $\gamma_{\text{in}}(w) \leftarrow \gamma_{\text{in}}(w) - 1$
16 **return** A

simply B, but it is updated as the algorithm reveals more resistant nodes. However, among all the incoming neighbors of a node v, we are only interested in the ones that will be "newly" discovered benign. Thus, we define $\hat{\Gamma}(v)$, which excludes unrelated nodes such as the already discovered ones (that is, $B \cup S$) or already discovered benigns. We continuously update $\hat{\Gamma}(v)$ as more nodes are revealed and added to A. More precisely, for each node v, if $r(v)$ is revealed to be 1, then we must consider every in-neighbor u of v, and for each out-neighbor w of u, we then remove u from the set of in-neighbors of w. This is because u has already been discovered to be benign. The pseudocode of this algorithm is provided in Algorithm 2.

The `For` loop runs in $O(n + m)$. The `While` loop is computed k times, and the lines 7-9 can clearly be executed in $O(n)$. The complexity of the `If` statement part is $O(\Delta_{\text{in}} \cdot \Delta_{\text{out}})$. This is achieved by using the appropriate data structure (please refer to the full version [12] for more details). Thus, the time complexity of this algorithm is $O((n+m) + k \cdot (n + \Delta_{\text{in}} \cdot \Delta_{\text{out}}))$, which can be bounded by $O(kn^2)$.

Algorithm 3 Proposed Algorithm: Select Top k Nodes

Input Input $G = (V, E)$, B, S, budget k, $p_r(\cdot)$
Output List of k nodes

$node\_values \leftarrow \emptyset$
for each node v in B **do**
  $value \leftarrow (1 - p_r(v)) \cdot |\Gamma_{\text{in}}(v) \setminus (B \cap S)|$
  $node\_values \leftarrow node\_values \cup \{(v, value)\}$
$top\_k\_nodes \leftarrow$ top k nodes of $node\_values$ based on $value$ using median of medians algorithm
return top_k_nodes

3.2 Discovering Potential Attack Edges

Definition 3.3. *An edge $e = (u, v)$ is a **P**otential **A**ttack **E**dge (PAE) if $u \in V \setminus (B \cup S)$, $v \in B$, and $r(v) = 0$. In other words, it is an incoming edge to a benign v with $r(v) = 0$.*

Discovering Potential Attack Edges Problem
Input: Given $G = (V, E)$, sets of benigns and sybils B and S such that $B, S \subseteq V$, $B \cap S = \emptyset$, budget k, and probability of resistant $p_r : V \rightarrow [0, 1]$.
Goal: Maximize expected number of discovered potential attack edges by revealing $r(v)$ of k nodes from B.

For each node v, we know $d_{\text{in}}(v)$ and $p_r(v)$, and $(1 - p_r(v)) \cdot |\Gamma_{\text{in}}(v) \setminus (B \cap S)|$ is the expected number of edges which will be discovered by revealing the node v. Since there is no overlap between edges discovered by each revealed node, choosing k nodes with the highest value of $(1 - p_r(v)) \cdot |\Gamma_{\text{in}}(v) \setminus (B \cap S)|$ is the optimal solution. This is outlined in Algorithm 3. For choosing the top k nodes based on the computed values, we can use the median of medians algorithm [5], which runs in $O(n)$. Thus, the overall time complexity is linear. As we will discuss, we can reduce the weight of discovered PAEs as a preprocessing step to assist sybil detection algorithms.

4 EXPERIMENTS

After establishing our experiments setting in Sections 4.1 and 4.2, we test our proposed algorithms for maximizing benigns problem and potential attack edges problem in Section 4.3. Then, in Section 4.4, we study the impact of applying our algorithms as a preprocessing step for various sybil/benign classification methods.

4.1 Experimental Setup

We use real-world graph data as the benign set. Then, our attack strategies are used to generate sybils and their connections to sybil/benigns. For the graph, the data used include Facebook, LastFM, and Twitter from SNAP [25]. In addition, we use, Pokec network [2] (we use only the subgraph induced by benigns). Some statistics of these networks are presented in the full version [12].

[2] https://github.com/binghuiwang/sybildetection/blob/master/Directed_Pokec.rar (accessed July 20, 2024)

For undirected networks, we assume the edges in both directions exist. After generating sybils and attack edges in each dataset, as explained in the next section, we randomly selected 2% of benigns and an equal number of sybils for the training set. The remaining sybils were included in the test set, along with an equal number of benigns. This means the size of known benigns for Facebook, Pokec, LastFM, and Twitter datasets are 80, 200, 150, and 200, respectively.

4.2 Attack Strategies

To choose the parameters in our attack models, we rely on some real-world statistics. Facebook reports indicate that the proportion of sybils is around 16% [29]. A 2013 study also found that 10% of Twitter users were fake [37]. Thus, we choose the size of the sybil set to be around 10% of the network in our attack strategies. Furthermore, Vishwanath [36] conducted a study involving the creation of fake Facebook profiles and sending friend requests to students in a large university. It was observed that only 30% of students declined friend requests from fake Facebook profiles, 52% were undecided after two weeks, and 18% accepted immediately. Based on these results, we chose 25% of nodes to be non-resistant.

To compute the probability of resistance $p_r(v)$ for a node v based on resistance $r(v)$, we generate a random number x between 0 and 1 and then we use $p_r(v) = (1 - r(v))x^3 + r(v)(1 - x^3)$. We choose this simple formula to introduce randomness while giving higher probabilities when $r(v) = 1$ and lower probabilities when $r(v) = 0$. For example, for $r(v) = 1$, the probability of $p_r(v) \geq 0.5$ is 0.79, and the average probability value is 0.75. In addition, in all attack strategies, we set $c = 4$ (please refer to Section 2 for the parameters of the attack models) since we aim for a non-resistance ratio of 25%. This way, the number of accepted attacks for a sybil s_i in expectation will be $\frac{25}{100} \cdot c \cdot AE(i) = AE(i)$, which is our desired value. Some statistics on the outcome of the attack strategies are provided in the full version [12].

4.3 Preprocessing

In this section, we evaluate our proposed algorithms for maximizing benigns and potential attack edges problems.

4.3.1 Maximizing Benigns. We compare our proposed algorithms, Monte Carlo Greedy (Algorithm 1) and Traversing (Algorithm 2), against the following baseline methods.

1. Random. Randomly select k nodes from benign set B to reveal.

2. Highest-Resistance. Pick k nodes with highest $p_r(\cdot)$ from B.

3. Highest-Resistance-and-Degree. Pick k nodes v with highest $p_r(v).|\Gamma_{\text{in}}(v) \setminus (B \cup S)|$ from B.

Between our two proposed algorithms, Traversing reveals the resistance of each selected node and uses that outcome information when selecting the next nodes, while the Monte Carlo Greedy does not. To make this a fairer comparison, we also consider a Monte Carlo Greedy variant, which reveals each selected node's resistance. This is called Resistance Aware Monte Carlo Greedy (please refer to the full version [12] for an exact description of this algorithm).

Figure 2 illustrates the performance of different algorithms on the Facebook dataset. (The high variance in the results of the Monte Carlo Greedy algorithms in this experiment is attributed to the fact that the run was performed only once.) We observe that our proposed algorithms, Traversing and two variants of Monte Carlo, significantly outperform other algorithms. Among our proposed algorithms, the Traversing algorithm performs better than the Monte Carlo Greedy algorithms. Furthermore, based on our experiments, the Traversing algorithm is substantially faster than the two Monte Carlo Greedy algorithms. For example, for the Facebook dataset and Preferential Attachment attack strategy, with a budget of $k = 30$, the Traversing algorithm completes in 119 milliseconds, while the Monte Carlo Greedy algorithm takes 52 minutes. Thus, the Traversing algorithm is not only more *accurate*, but also *significantly faster*. For other datasets, similar results are provided in the full version [12].

4.3.2 Potential Attack Edges. We now evaluate the performance of our Proposed Algorithm 3 for the potential attack edges (PAE) problem against the Random algorithm, which makes k random choices. Figure 3 (top row) illustrates the number of PAEs found by each algorithm for a range of budgets. Our algorithm outperforms the Random algorithm. This is unsurprising since our algorithm is optimal, as we proved in Section 3.2.

One natural question is what fraction of PAEs are, in fact, attack edges (edges from sybil to benign). We report this information in Figure 3 (second row). As one can observe, for our algorithm, around 20% of found PAEs are attack edges for various budget choices. To get a better understanding of how good this performance is, we consider the Full-Knowledge algorithm. We suppose this algorithm knows all the values of resistance (note that this information is not available to our algorithm) and aims to greedily pick nodes which maximize the ratio of attack edges over PAEs. While for small budgets, the gap is large, as the budget grows, the ratio of attack edges over PAEs for our algorithm almost matches the Full-Knowledge algorithm, which is impressive considering that our algorithm doesn't know the exact values of resistance $r(\cdot)$. For other datasets, similar results are provided in the full version [12].

4.4 Classification Algorithms

Our algorithms for discovering benigns and potential attack edges can act as a preprocessing step for various sybil detection algorithms. We analyze the performance of three state-of-the-art detection algorithms, with and without such preprocessing step.

We consider the SybilWalk [21] and SybilSCAR [39] algorithms. We also analyze a node classifier algorithm based on logistic regression that we call SybilMetric [2]. We first executed each of these algorithms without any preprocessing. We then applied the Traversing algorithm to maximize the number of benigns (our best algorithm for maximizing benigns based on our experiments in the previous section) before running the detection strategies. In the final setup, we also used our potential attack edges discovering algorithm on both the initial known benigns and the newly identified benigns from the first preprocessing step. Since detection algorithms usually rely on homophily property, removing (or reducing the weight of) attack edges could benefit them. Therefore, in the last setup, before running the detection algorithms, the weights for the discovered PAEs were adjusted to reduce their importance.

Furthermore, in all experiments, for both preprocessing phases, the budget is set to 1% of benigns. We used the same setup as the prior work for the studied detection algorithms.

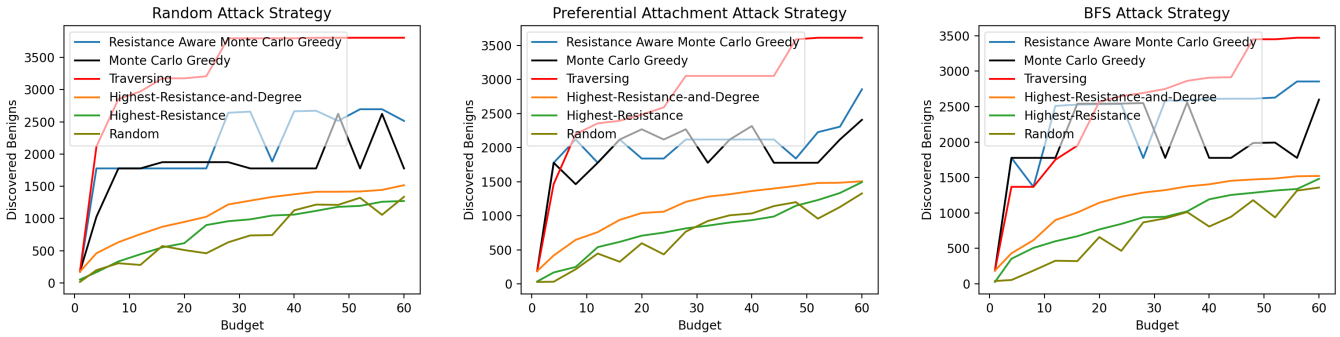


Figure 2: The number of discovered benigns by each algorithm when the budget ranges from 1 to k in the maximizing benigns problem on the *Facebook* dataset and for different attack strategies.

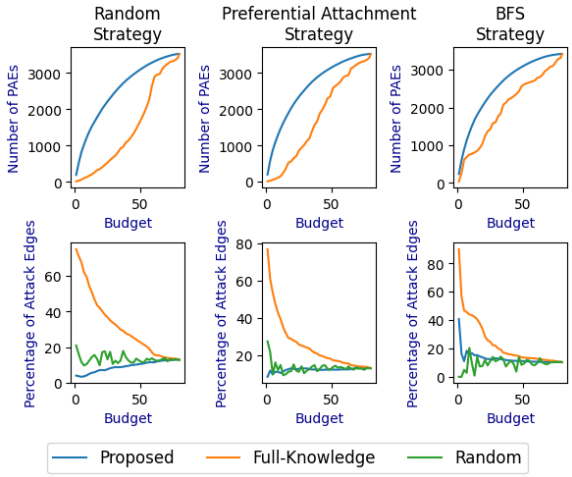


Figure 3: The number of discovered PAEs for different budgets (first row) and the percentage of attack edges relative to the number of discovered PAEs (second row) on the *Facebook* dataset. Each column represents a different attack strategy.

Our comparison results for the Facebook dataset are shown in Table 1. To measure the performance of the detection algorithms, we use AUC (Area Under the Curve), which is commonly used by the prior work since it is less affected by biases compared to accuracy. Based on these results, the performance of all three algorithms is improved after adding maximizing benigns outputs. Using discovered PAEs as a preprocessing step also enhances the AUC in some cases, but not always. This flags that one may need to incorporate the information regarding the PAEs directly into these algorithms instead of just reducing weights as we did to get a stronger boost in the performance. (This would be an interesting potential avenue for future research.) Thus, overall, our preprocessing steps, especially discovering benigns, can significantly boost the performance of detection strategies. For other datasets, similar results are provided in the full version [12]. It is also worth emphasizing that our preprocessing steps are not computationally demanding. More precisely,

Table 1: Performance of SybilSCAR, SybilWalk, and SybilMetric with and without preprocessing on the *Facebook* dataset. Init represents the setup without preprocessing. MB incorporates discovered benigns by the Traversing algorithm. MB+PAE incorporates both discovered benigns and PAEs.

Attack Strategy	Step	SybilSCAR AUC	SybilWalk AUC	SybilMetric AUC
Random	Init	0.924	0.966	1.00
	MB	0.988	0.998	1.00
	MB+PAE	0.988	0.998	0.99
BA	Init	0.876	0.929	1.00
	MB	0.954	0.972	1.00
	MB+PAE	0.944	0.972	1.00
BFS	Init	0.986	0.985	0.97
	MB	0.995	0.996	0.99
	MB+PAE	0.991	0.997	1.00

in most cases, the computational overhead added by the preprocessing step is negligible in comparison to the time required by the detection algorithm itself.

5 CONCLUSION

In this paper, we introduced novel attack strategies for synthesizing more realistic and generic datasets by introducing the concept of user resistance. We then considered two optimization problems where we leveraged resistance information to discover benigns and potential attack edges. We introduced several algorithms and theoretically analyzed their runtime and solution accuracy. We then showed the outcomes of these algorithms contain valuable information that can be used as a preprocessing step for detection algorithms. We conducted a large set of experiments that confirmed the positive impact of our preprocessing algorithms

Future research could explore dynamic attack strategies to enhance the adaptability of sybil detection algorithms. Developing bias-robust algorithms is also essential, as current methods are vulnerable to dataset biases, especially when preprocessing reveals new benigns and skews the dataset. In addition, developing algorithms for maximizing benigns problem with theoretical guarantees could be a potential avenue for future work.

REFERENCES

[1] Kayode Sakariyah Adewole, Nor Badrul Anuar, Amirrudin Kamsin, Kasturi Dewi Varathan, and Syed Abdul Razak. 2017. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications* 79 (2017), 41–67.

[2] Sara Asghari, Mostafa Haghir Chehreghani, and Morteza Haghir Chehreghani. 2022. On using node indices and their correlations for fake account detection. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 5656–5661.

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[4] Andrew An Bian, Joachim M. Buhmann, Andreas Krause, and Sebastian Tschiatschek. 2017. Guarantees for Greedy Maximization of Non-submodular Functions with Applications. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 498–507.

[5] Manuel Blum, Robert W Floyd, Vaughan R Pratt, Ronald L Rivest, and Robert E Tarjan. 1973. Time bounds for selection. *J. Comput. System Sci.* 7, 4 (1973), 448–461. https://doi.org/10.1016/S0022-0000(73)80033-9

[6] Bharat S Borkar, Dipak R Patil, Ashok V Markad, and Manish Sharma. 2022. Real or fake identity deception of social media accounts using recurrent neural network. In *2022 International Conference on Fourth Industrial Revolution Based Technology and Practices (ICFIRTP)*. IEEE, 80–84.

[7] Yazan Boshmaf, Dionysios Logothetis, Georgos Siganos, Jorge Lería, Jose Lorenzo, Matei Ripeanu, Konstantin Beznosov, and Hassan Halawa. 2016. Íntegro: Leveraging victim prediction for robust fake account detection in large scale OSNs. *Computers & Security* 61 (2016), 142–168.

[8] Adam Breuer, Ran Eilat, and Udi Weinsberg. 2020. Friend or Faux: Graph-Based Early Detection of Fake Accounts on Social Networks. In *Proceedings of The Web Conference 2020*. Association for Computing Machinery, 1287–1297.

[9] Adam Breuer, Nazanin Khosravani, Michael Tingley, and Bradford Cottel. 2023. Preemptive Detection of Fake Accounts on Social Networks via Multi-Class Preferential Attachment Classifiers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Long Beach, CA, USA) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 105–116.

[10] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *9th USENIX symposium on networked systems design and implementation (NSDI 12)*. USENIX Association, 197–210.

[11] George Danezis and Prateek Mittal. 2009. Sybilinfer: Detecting sybil nodes using social networks.. In *Ndss*, Vol. 9. San Diego, CA, 1–15.

[12] Ali Safarpoor Dehkordi and Ahad N. Zehmakan. 2025. More Efficient Sybil Detection Mechanisms Leveraging Resistance of Users to Attack Requests. arXiv:2501.16624

[13] Buket Erşahin, Özlem Aktaş, Deniz Kılınç, and Ceyhun Akyol. 2017. Twitter fake account detection. In *2017 international conference on computer science and engineering (UBMK)*. IEEE, 388–392.

[14] Uriel Feige. 1998. A threshold of ln n for approximating set cover. *Journal of ACM* 45, 4 (1998), 634–652.

[15] Peng Gao, Binghui Wang, Neil Zhenqiang Gong, Sanjeev R Kulkarni, Kurt Thomas, and Prateek Mittal. 2018. Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *2018 IEEE conference on communications and network security (CNS)*. IEEE, 1–9.

[16] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. 2014. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE transactions on information forensics and security* 9, 6 (2014), 976–987.

[17] Bharti Goyal, Nasib Singh Gill, Preeti Gulia, Om Prakash, Ishaani Priyadarshini, Rohit Sharma, Ahmed J Obaid, and Kusum Yadav. 2023. Detection of fake accounts on social media using multimodal data with deep learning. *IEEE Transactions on Computational Social Systems* (2023).

[18] Jianxiong Guo, Yi Li, and Weili Wu. 2019. Targeted protection maximization in social networks. *IEEE Transactions on Network Science and Engineering* 7, 3 (2019), 1645–1655.

[19] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding* (1994), 409–426.

[20] Gordhan Jethava and Udai Pratap Rao. 2022. User behavior-based and graph-based hybrid approach for detection of sybil attack in online social networks. *Computers and Electrical Engineering* 99 (2022), 107753.

[21] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2017. Random walk based fake account detection in online social networks. In *2017 47th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 273–284.

[22] Zafran Khan, Zeeshan Khan, Byung-Geun Lee, Hong Kook Kim, and Moongu Jeon. 2024. Graph neural networks based framework to analyze social media platforms for malicious user detection. *Applied Soft Computing* 155 (2024), 111416.

[23] Priyanka Kondeti, Lakshmi Pranathi Yerramreddy, Anita Pradhan, and Gandharba Swain. 2021. Fake account detection using machine learning. In *Evolutionary Computing and Mobile Sustainable Networks: Proceedings of ICECMSN 2020*. Springer, 791–802.

[24] Ngoc C Le, Manh-Tuan Dao, Hoang-Linh Nguyen, Tuyet-Nhi Nguyen, and Hue Vu. 2020. An application of random walk on fake account detection problem: A hybrid approach. In *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 1–6.

[25] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved July 20, 2024 from http://snap.stanford.edu/data

[26] Xiao Liang, Zheng Yang, Binghui Wang, Shaofeng Hu, Zijie Yang, Dong Yuan, Neil Zhenqiang Gong, Qi Li, and Fang He. 2021. Unveiling fake accounts at the time of registration: An unsupervised approach. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. Association for Computing Machinery, 3240–3250.

[27] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM international conference on information and knowledge management*. Association for Computing Machinery, 2077–2085.

[28] Haoyu Lu, Daofu Gong, Zhenyu Li, Feng Liu, and Fenlin Liu. 2023. Sybilhp: Sybil detection in directed social networks with adaptive homophily prediction. *Applied Sciences* 13, 9 (2023), 5341.

[29] Martin Moore. 2023. Fake accounts on social media, epistemic uncertainty and the need for an independent auditing of accounts. *Internet Policy Review* 12, 1 (2023).

[30] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14 (1978), 265–294.

[31] Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[32] Devakunchari Ramalingam and Valliyammai Chinnaiah. 2018. Fake profile detection techniques in large-scale online social networks: A comprehensive review. *Computers & Electrical Engineering* 65 (2018), 165–177.

[33] Santosh Kumar Uppada, K Manasa, B Vidhathri, R Harini, and B Sivaselvan. 2022. Novel approaches to fake news and fake account detection in OSNs: user social engagement and visual content centric model. *Social Network Analysis and Mining* 12, 1 (2022), 52.

[34] Leslie G Valiant. 1979. The complexity of enumeration and reliability problems. *siam Journal on Computing* 8, 3 (1979), 410–421.

[35] Estée Van Der Walt and Jan Eloff. 2018. Using machine learning to detect fake identities: bots vs humans. *IEEE access* 6 (2018), 6540–6549.

[36] Arun Vishwanath. 2018. Why Do So Many People Fall for Fake Profiles Online? Retrieved April 18, 2024 from https://theconversation.com/why-do-so-many-people-fall-for-fake-profiles-online-102754

[37] Keith Wagstaff. 2013. 1 in 10 Twitter Accounts Fake, Say Researchers. Retrieved April 18, 2024 from https://www.nbcnews.com/technology/1-10-twitter-accounts-fake-say-researchers-2d11655362

[38] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. 2017. GANG: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 465–474.

[39] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. SybilSCAR: Sybil detection in online social networks via local rule based propagation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[40] Fan Xu, Nan Wang, Hao Wu, Xuezhi Wen, Xibin Zhao, and Hai Wan. 2024. Revisiting graph-based fraud detection in sight of heterophily and spectrum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 9214–9222.

[41] Peipei Yang and Zhuoyuan Zheng. 2020. Fake account detection with attention-based graph convolution networks. In *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*. IEEE, 106–110.

[42] Minji Yoon. 2021. Graph fraud detection based on accessibility score distributions. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer, 483–498.

[43] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. 2008. Sybillimit: A near-optimal social network defense against sybil attacks. In *2008 IEEE Symposium on Security and Privacy*. IEEE, 3–17.

[44] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. 2006. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. Association for Computing Machinery, 267–278.

[45] Dong Yuan, Yuanli Miao, Neil Zhenqiang Gong, Zheng Yang, Qi Li, Dawn Song, Qian Wang, and Xiao Liang. 2019. Detecting fake accounts in online social networks at the time of registrations. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. Association for Computing Machinery, 1423–1438.

[46] Xiaoying Zhang, Hong Xie, Pei Yi, and John CS Lui. 2022. Enhancing Sybil detection via social-activity networks: A random walk approach. *IEEE Transactions on Dependable and Secure Computing* 20, 2 (2022), 1213–1227.