Removing Malicious Nodes from Networks

Sixie Yu
Computer Science and Engineering
Washington University in St. Louis
sixie.yu@wustl.edu

Yevgeniy Vorobeychik
Computer Science and Engineering
Washington University in St. Louis
yvorobeychik@wustl.edu

ABSTRACT

A fundamental challenge in networked systems is detection and removal of suspected malicious nodes. In reality, detection is always imperfect, and the decision about which potentially malicious nodes to remove must trade off false positives (erroneously removing benign nodes) and false negatives (mistakenly failing to remove malicious nodes). However, in network settings this conventional tradeoff must now account for node connectivity. In particular, malicious nodes may exert malicious influence, so that mistakenly leaving some of these in the network may cause damage to spread. On the other hand, removing benign nodes causes direct harm to these, and indirect harm to their benign neighbors who would wish to communicate with them. We formalize the problem of removing potentially malicious nodes from a network under uncertainty through an objective that takes connectivity into account. We show that optimally solving the resulting problem is NP-Hard. We then propose a tractable solution approach based on a convex relaxation of the objective. Finally, we experimentally demonstrate that our approach significantly outperforms both a simple baseline that ignores network structure, as well as a state-of-the-art approach for a related problem, on both synthetic and real-world datasets.

ACM Reference Format:
Sixie Yu and Yevgeniy Vorobeychik. 2019. Removing Malicious Nodes from Networks. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 10 pages.

1 INTRODUCTION

The problem of removing malicious nodes from networks has long been of considerable importance, and it has attracted a great deal of recent attention. In social networks, accounts occupied by malicious parties spread toxic information (e.g., hate speech, fake news, and spam), stirring up controversy and manipulating political views among social network users [1, 6]. Major social media entities, such as Facebook, have devoted considerable effort on identifying and removing fake or malicious accounts [16, 17]. Despite these efforts, there is evidence that the problem is as prevalent as ever [2, 14]. A similar challenge obtains in cyber-physical systems (e.g., smart grid infrastructure), where computing nodes compromised by malware can cause catastrophic losses [13], but removing non-malicious nodes may cause power failure [22].

A common thread in these scenarios is the tradeoff faced in deciding which nodes to remove: removing a benign node (false positive) causes damage to this node, which may be inconvenience or loss of productivity, and potentially also results in indirect losses to its neighbors; on the other hand, failing to remove a malicious node (false negative) can have deliterious effects as malicious influence spreads to its neighbors. The key observation is that the loss associated with a decision whether to remove a node depends both on the node's likelihood of being malicious and its *local network structure*. Consequently, the typical approach in which we simply classify nodes as malicious or benign using a threshold on the associated maliciousness probability [8] is inadequate, as it fails to account for network consequences of such decisions. Rather, the problem is fundamentally about choosing which subset of nodes to remove, as decisions about removing individual nodes are no longer independent.

We consider the problem of choosing which subset of nodes to remove from a network given an associated probability distribution over joint realizations of all nodes as either malicious or benign (that is, we allow probability that node i is malicious to depend on whether its neighbors are malicious, as in collective classification and relational learning [12, 19]). We then model the problem as minimizing expected loss with respect to this distribution, where the loss function is composed of three parts: the *direct* loss ($\mathcal{L}_1$) stemming from removed benign nodes, the *indirect* loss associated with cutting links between removed and remaining benign nodes ($\mathcal{L}_2$), and the loss associated with malicious nodes that remain, quantified in terms of *links* these have to benign nodes ($\mathcal{L}_3$).

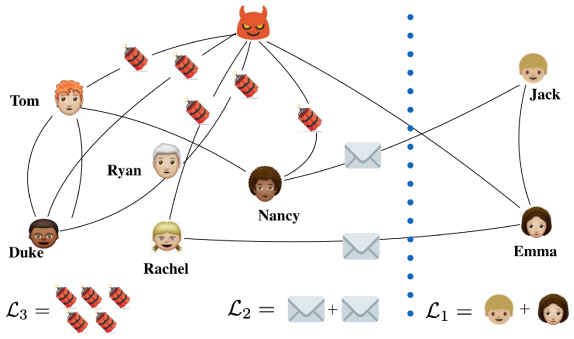

Figure 1: An illustration of a decision to remove two nodes, Jack and Emma, from the network, on our loss function.

To illustrate, consider Figure 1. In this example, we have decided to remove, Jack and Emma, the two *benign* nodes on the right of the vertical dotted line. On the other hand, we chose not to remove the malicious node in red. Suppose that we pay a penalty of α_1 for each benign node we remove, a penalty of α_2 for each link we cut between two benign nodes, and α_3 for each link between *remaining* malicious and benign nodes. Since we removed 2 benign nodes ($\mathcal{L}_1 = 2$), cut 2 links between benign nodes (one between Jack and

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

Nancy, and another between Emma and Rachel; $\mathcal{L}_2 = 2$), and the malicious node is still connected to 5 benign nodes (Tom, Duke, Ryna, Rachel, and Nancy; $\mathcal{L}_3 = 5$), our total loss is $2\alpha_1 + 2\alpha_2 + 5\alpha_3$. If we instead only removed the malicious node, our total loss would have been 0, while removing the malicious node instead of Emma (but together with Jack) would result in the loss of $\alpha_1 + 2\alpha_2$.

As minimizing our loss function is intractable, we resort to its convex relaxation. We solve the convex relaxation for a globally optimal solution, and then convert it to an approximate solution to the original problem by Euclidean projection. Extensive experiments demonstrate that our approach is better than the baseline which treats nodes as independent, and both better and significantly more scalable than a state of the art approach for a related problem.

In summary, our contributions are:

(1) a model that captures both direct and indirect effects of mistakes in removing benign and malicious nodes from the network,
(2) an algorithm based on convex relaxation for computing an approximately optimal solution to our problem, and
(3) extensive experimental evaluation of our approach on both synthetic and real-world data.

Related Work. There are several prior efforts dealing with a related problem of *graph scan statistics* and hypothesis testing [3, 15, 18]. These approaches focus on the following scenario. We are given a graph G where each node in the graph is associated with a random variable. The null hypothesis is that these random variables are sampled from the standard Gaussian distribution $\mathcal{N}(0, 1)$, while the alternative hypothesis is that there is a fraction of nodes (malicious nodes) where the random variables associated with them are sampled from a Gaussian distribution $\mathcal{N}(\mu, 1)$ with μ other than 0. A scan statistics T is defined, which can be thought as a function defined over random variables associated with a subset of nodes. Then the hypothesis test is equivalent to maximizing T over subset of nodes, and the null hypothesis is rejected if strong evidence exists (i.e. large value of T).

Arias-Castro et al. [3] proposed a scan statistic for special graph models. Priebe et al. [15] proposed a scan statistic defined over clusters with special geometric structures. These methods do not easily generalize to arbitrary graph models or arbitrary scan statistics. Sharpnack et al. [18] employed the generalized log-likelihood ratio as the scan statistic. By assuming that the set of malicious nodes has sparse connections with others, the hypothesis test can be converted to solving a graph cut problem, which is further relaxed into a convex optimization by leveraging the Lovász extension of a graph cut.

Our problem can be formulated as a hypothesis testing problem. A random variable associated with each node indicates whether it's malicious or not, with associated maliciousness probability. Computing a set of malicious nodes is then equivalent to searching for a subset of nodes that maximizes the graph scan statistic T, which provides the strongest evidence to reject the null hypothesis. However, there are several problems with this formulation. First, in our setting we are not solely concerned about *direct* loss (wrongly removing benign nodes or wrongly keeping malicious nodes), but also the *indirect* loss, for example, the number of edges that have been cut between benign nodes, which is diffcult to capture using a single graph scan statistic (i.e. generalized log-likelihood ratio). Second, hypothesis testing with graph scan statistics usually requires one to solve a combinatorial optimization problem that has an exponentially large search space. Consequently, it is typically necessary to assume special structure about the problem (i.e. Sharpnack et al. [18] assumed small cut-size). In contrast, our proposed approach considers direct and indirect loss associated with mistakes, and makes no assumptions about graph structure.

2 MODEL

We consider a social network that is represented by a graph $G = (V, E)$, where V is the set of nodes ($|V| = N$) and E the set of edges connecting them. Each node $i \in V$ represents a user and each edge (i, j) represents an edge (e.g., friendship) between users i and j. For simplicity, we focus our discussion on undirected graphs, although this choice is not consequential for our results. We denote the adjacency matrix of G by $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. The elements of $\boldsymbol{A}$ are either 1/0 if the graph is unweighted, or some non-negative real numbers if the graph is weighted. Again, we simplify exposition by focusing on unweighted graphs; generalization is direct.

We consider the problem of removing malicious nodes from the network G. We explain the problem by first considering complete information about the identity of malicious and benign nodes, and subsequently describe our actual model in which this information is unknown (as this in fact is the crux of our problem). Specifically, let $\pi \in \{0, 1\}^N$ be a *configuration* of the network, with $\pi_i = 1$ indicating that a node i is malicious, with $\pi_i = 0$ when i is benign. For convenience, we also define $\bar{\pi}_i = 1 - \pi_i$ that indicates whether i is benign. Consequently, π (and $\bar{\pi}$) assigns a malicious or benign label to every node. Let the malicious and benign nodes be denoted by V^+ and V^-, respectively. Our goal is to identify a subset of nodes S to remove in order to minimize the impact of the remaining malicious nodes on the network, while at the same time minimizing disruptions caused to the benign subnetwork.

To formalize this intuition, we define a loss function associated with the set S of nodes to remove. This loss function has three components, each corresponding to a key consideration in the problem. The first part of the loss function, $\mathcal{L}_1 = |V^- \cap S|$, is the *direct* loss associated with removing benign nodes; this simply penalizes every false positive, as one would naturally expect, but ignores the broken *relationships* among benign nodes that result from our decision. That is captured by the second component, $\mathcal{L}_2 = |\{(i, j) | i \in (V^- \cap (V \setminus S)), j \in (V^- \cap S), \forall i, j \in V\}|$, which imposes a penalty for cutting connections between benign nodes that are removed and benign nodes that remain. In other words, the second loss component captures the *indirect* consequence of removing benign nodes *on the structure of the benign subnetwork*. This aspect is critical to capture in network settings, as relationships and connectivity are what networks are about. The third component of the loss function, $\mathcal{L}_3 = \left|\{(i, j) | i \in \left(V^+ \cap (V \setminus S)\right), j \in (V^- \cap (V \setminus S))\}\right|$, measures the consequence of *failing to remove malicious nodes* in terms of connections from these to benign nodes. At the high level, this part of the loss naturally captures the influence that unremoved malicious nodes can exert on the benign part of the network.

The total loss combines these three components as a weighted sum, $\mathcal{L} = \alpha_1 \mathcal{L}_1 + \alpha_2 \mathcal{L}_2 + \alpha_3 \mathcal{L}_3$, with $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Other than

this constraint, we allow α_is to be arbitrary relative weights of the different components, specified depending on the domain. For example, if we are concerned about false positives, but not very much about network structure, we would set $\alpha_1 \gg \alpha_2$. Alternatively, we can set these coefficients to normalize the relative magnitudes of the loss terms (for example, setting $\alpha_1 = \frac{1}{N}$ and $\alpha_2 = \alpha_3 = \frac{N-1}{2N}$).

We now rewrite the loss function in a way that will prove more mathematically convenient. Let $\boldsymbol{s} \in \{0,1\}^N$, where $\boldsymbol{s}_i = 1$ if and only if node i is removed ($i \in S$), and, for convenience, let $\bar{\boldsymbol{s}} = \mathbf{1} - \boldsymbol{s}$, where $\bar{s}_i = 1$ if node i remains in the network ($i \in V \setminus S$). Then, the loss associated with $(\boldsymbol{s}, \bar{\boldsymbol{s}})$ is

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{s}, \bar{\boldsymbol{s}}) := \underbrace{\alpha_1 \sum_{i=1}^{N} \boldsymbol{s}_i \bar{\boldsymbol{\pi}}_i}_{\mathcal{L}_1} + \underbrace{\alpha_2 \sum_{i,j} \boldsymbol{A}_{i,j} \boldsymbol{s}_i \bar{\boldsymbol{s}}_j \bar{\boldsymbol{\pi}}_i \bar{\boldsymbol{\pi}}_j}_{\mathcal{L}_2} + \underbrace{\alpha_3 \sum_{i,j}^{N} \bar{\boldsymbol{s}}_i \bar{\boldsymbol{s}}_j \boldsymbol{A}_{i,j} \boldsymbol{\pi}_i \bar{\boldsymbol{\pi}}_j}_{\mathcal{L}_3}. \quad (1)$$

With complete information, it is immediate that the loss is minimized if S contains all, and only, the malicious nodes. Our main challenge is to solve this problem when the identity of malicious and benign nodes is uncertain, and instead we have a probability distribution over these. This probability distribution may capture any prior knowledge, or may be obtained by learning probability that a node is malicious given its features from past data. To formalize, let $\pi \sim \mathcal{P}$, where $\mathcal{P}$ captures the joint probability distribution over node configurations (malicious or benign). For our purposes, we make no assumptions on the nature of this distribution; a special case would be when maliciousness probabilities for nodes are independent (conditional on a node's observed features), but our model also captures natural settings in which configurations of network neighbors are correlated (e.g., when malicious nodes tend to have many benign neighbors). Our expected loss that we aim to minimize then becomes

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}}) := \\
&\mathbb{E}_{\pi\sim\mathcal{P}}\left[\alpha_1 \sum_{i=1}^{N} \boldsymbol{s}_i \bar{\boldsymbol{\pi}}_i + \alpha_2 \sum_{i,j}^{N} \boldsymbol{A}_{i,j} \boldsymbol{s}_i \bar{\boldsymbol{s}}_j \bar{\boldsymbol{\pi}}_i \bar{\boldsymbol{\pi}}_j + \alpha_3 \sum_{i,j}^{N} \bar{\boldsymbol{s}}_i \bar{\boldsymbol{s}}_j \boldsymbol{A}_{i,j} \boldsymbol{\pi}_i \bar{\boldsymbol{\pi}}_j\right] \\
&= \alpha_1 \sum_{i=1}^{N} \boldsymbol{s}_i \mathbb{E}_{\pi\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}_i] + \alpha_2 \sum_{i,j}^{N} \boldsymbol{A}_{i,j} \boldsymbol{s}_i \bar{\boldsymbol{s}}_j \mathbb{E}_{\pi\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}_i \bar{\boldsymbol{\pi}}_j] \\
&\qquad + \alpha_3 \sum_{i,j}^{N} \bar{\boldsymbol{s}}_i \bar{\boldsymbol{s}}_j \boldsymbol{A}_{i,j} \mathbb{E}_{\pi\sim\mathcal{P}}[\boldsymbol{\pi}_i \bar{\boldsymbol{\pi}}_j].
\end{aligned} \quad (2)$$

While we will assume that we know $\mathcal{P}$ in the remaining technical discussion, we relax this assumption in our experimental evaluation, where we also demonstrate that our approach is quite robust to errors in our estimation of $\mathcal{P}$.

In order to have a concise representation of our objective, we convert Eq. (2) to a matrix-vector form. Note that the configuration $\boldsymbol{\pi}$ of network is a random variable distributed according to $\mathcal{P}$. We let $\boldsymbol{\mu} \in \mathbb{R}^{N\times 1}$ and $\Sigma \in \mathbb{R}^{N\times N}$ denote its mean and covariance, respectively. For convenience we let $\boldsymbol{J}(n,m) \in \mathbb{R}^{n\times m}$ denote a matrix with all elements equal to one with dimensions determined by the arguments n and m. We define a diagonal matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$, where the diagonal entries are equal to $\mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}] = \mathbf{1} - \boldsymbol{\mu}$. Note that $\mathbf{1} \in \mathbb{R}^{N\times 1}$ is a vector with all elements equal to one. We define another matrix $\boldsymbol{P} := \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}\bar{\boldsymbol{\pi}}^T]$, where the operator $\odot$ denotes Hadamard product. By replacing $\bar{\boldsymbol{\pi}}$ with $\mathbf{1} - \boldsymbol{\pi}$ and leveraging the linearity of expectation we have:

$$\begin{aligned}
\boldsymbol{P} :=& \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}\bar{\boldsymbol{\pi}}^T] \\
=& \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[(\mathbf{1} - \boldsymbol{\pi})(\mathbf{1} - \boldsymbol{\pi})^T] \\
=& \boldsymbol{A} \odot \left(\mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\mathbf{1}\mathbf{1}^T] - \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\mathbf{1}\boldsymbol{\pi}^T] - \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}\mathbf{1}^T] + \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}\boldsymbol{\pi}^T]\right) \\
=& \boldsymbol{A} \odot \left(\boldsymbol{J}(N,N) - \boldsymbol{J}(N,1) \times \boldsymbol{\mu}^T - \boldsymbol{\mu} \times \boldsymbol{J}(1,N) + \Sigma + \boldsymbol{\mu} \times \boldsymbol{\mu}^T\right).
\end{aligned} \quad (3)$$

Similarly we define $\boldsymbol{M} := \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}\bar{\boldsymbol{\pi}}^T]$. Then we have

$$\begin{aligned}
\boldsymbol{M} :&= \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}\bar{\boldsymbol{\pi}}^T] = \boldsymbol{A} \odot \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}(\mathbf{1} - \boldsymbol{\pi})^T] \\
&= \boldsymbol{A} \odot \left(\boldsymbol{\mu} \times \boldsymbol{J}(1,N) - \Sigma - \boldsymbol{\mu} \times \boldsymbol{\mu}^T\right).
\end{aligned} \quad (4)$$

We can now rewrite Eq. (2) in a matrix-vector form:

$$\mathcal{L}(\boldsymbol{s}, \bar{\boldsymbol{s}}) := \alpha_1 \mathbf{1}^T \boldsymbol{B} \boldsymbol{s} + \alpha_2 \left(\boldsymbol{s}^T \boldsymbol{P} \mathbf{1} - \boldsymbol{s}^T \boldsymbol{P} \boldsymbol{s}\right) + \alpha_3 \bar{\boldsymbol{s}}^T \boldsymbol{M} \bar{\boldsymbol{s}}. \quad (5)$$

3 SOLUTION APPROACH

The problem represented by Eq. (5) is a non-convex quadratic integer optimization problem, which is difficult to solve directly. Indeed, we show that our problem is NP-Hard. To begin, we re-arrange the terms in Eq. (5), which results in:

$$\begin{aligned}
\min_{\boldsymbol{s}} \quad & \boldsymbol{s}^T \boldsymbol{A}_1 \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{b}_1 + c_1 \\
s.t. \quad & \boldsymbol{s} \in \{0,1\}^N
\end{aligned} \quad (6)$$

where $\boldsymbol{A}_1$, $\boldsymbol{b}$ and c_1 are:

$$\begin{aligned}
\boldsymbol{A}_1 =& \alpha_3 \boldsymbol{M} - \alpha_2 \boldsymbol{P} \\
\boldsymbol{b}_1 =& \alpha_1 \boldsymbol{B}^T \mathbf{1} + \alpha_2 \boldsymbol{P} \mathbf{1} - \alpha_3 \boldsymbol{M} \mathbf{1} - \alpha_3 \boldsymbol{M}^T \mathbf{1} \\
c_1 =& \alpha_3 \mathbf{1}^T \boldsymbol{M} \mathbf{1}.
\end{aligned} \quad (7)$$

Since Eq. (6) is equivalent to Eq. (5), we prove the NP-hardness of minimizing Eq. (6).

Theorem 3.1. *Solving Problem* (6) *is NP-Hard.*

Proof. We construct the equivalence between a special case of the model defined in Eq. (6) and the *Maximum Independent Set* (MIS) problem . Given a graph $G = (V, E)$, the MIS problem is to find an independent set in G of maximum cardinality, which is NP-hard to solve. We specify the special case by considering a specific form of the loss function defined in Eq. (2) where:

(1) $\alpha_2 = 0$,
(2) $\mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}_i] = \mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\bar{\boldsymbol{\pi}}_i] = \frac{1}{2}, \forall i = 1, \ldots, N$,
(3) $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_j$ are independent random variables for any $i \neq j$, which means $\mathbb{E}_{\boldsymbol{\pi}\sim\mathcal{P}}[\boldsymbol{\pi}_i \bar{\boldsymbol{\pi}}_j] = \frac{1}{4}, \forall i \neq j$.
(4) $\alpha_3 > 2\alpha_1 M$, where M is a large positive number.

which leads to the follwing loss:

$$\mathcal{L}^\dagger = \underbrace{\frac{\alpha_1}{2} \sum_{i=1}^{N} \boldsymbol{s}_i}_{\mathcal{L}_1^\dagger} + \underbrace{\frac{\alpha_3}{4} \sum_{i,j}^{N} \boldsymbol{A}_{ij} \bar{\boldsymbol{s}}_i \bar{\boldsymbol{s}}_j}_{\mathcal{L}_2^\dagger}.$$

Denote the nodes in the maximum independent set of G as $\mathcal{K}$. We first show that keeping only the nodes in $\mathcal{K}$ is the optimal solution. Note that removing any node from $\mathcal{K}$ increases the loss, by incurring $\frac{\alpha_1}{2}$ losses added to $\mathcal{L}_1^\dagger$. Next we denote $V' = V \setminus \mathcal{K}$, which is the set of nodes removed from the graph. we show that putting any set of nodes in V' back to $\mathcal{K}$ increases the loss. Suppose we put a set of nodes $\mathcal{B} \subseteq V'$ back to $\mathcal{K}$. This must introduce additional edges to $\mathcal{K}$, otherwise $\mathcal{K}$ is not the maximum independent set. Let the number of additionally introduced edges be C. Putting $\mathcal{B}$ back to $\mathcal{K}$ decreases $\mathcal{L}_1^\dagger$. however, it increases $\mathcal{L}_2^\dagger$. The net change of $\mathcal{L}^\dagger$ is:

$$-\frac{\alpha_1}{2}|\mathcal{B}| + \frac{\alpha_3}{2}C,$$

which is always positive because $\alpha_3 > 2\alpha_1 M$. Since we cannot remove or add any set of nodes to $\mathcal{K}$ without increasing $\mathcal{L}^\dagger$, keeping only the nodes in $\mathcal{K}$ is the optimal solution.

For the other direction, we show that if keeping the nodes in a set $\mathcal{K}$ minimizes the loss, then $\mathcal{K}$ is the maximum independent set of G. First, suppose $\mathcal{K}$ is not an independent set, which means there is at least one edge in $\mathcal{K}$. Then removing one or both of the endpoints always decrease the loss because $\alpha_3 > 2\alpha_2 M$. Intuitively, the loss of removing a benign node from G is way less than the loss of leaving a malicious edge in G. So $\mathcal{K}$ must be an independent set. Next, we show $\mathcal{K}$ is the maximum independent set. Suppose another set $\mathcal{K}'$ is the maximum independent set and $|\mathcal{K}'| > |\mathcal{K}|$. Then keeping the nodes in $\mathcal{K}'$ can further decrease $\mathcal{L}^\dagger$ by decreasing $\mathcal{L}_1^\dagger$, which contradicts the fact that keeping the nodes in $\mathcal{K}$ minimizes the loss. Therefore we conclude $\mathcal{K}$ is the maximum independent set. □

Our approach to solving Eq. (6) is by means of a convex relaxation, as we now describe. Note that the matrix $\boldsymbol{A}_1$ in Eq. (6) is not symmetric. We substitute $\boldsymbol{A}_1$ with $\boldsymbol{Q} := \frac{\boldsymbol{A}_1 + \boldsymbol{A}_1^T}{2}$ and $\boldsymbol{b} := \frac{1}{2}\boldsymbol{b}_1$, which results in an equivalent problem:

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{s}^T\boldsymbol{b} + c_1 \\ s.t. \quad & \boldsymbol{s} \in \{0,1\}^N \end{aligned} \tag{8}$$

where $\boldsymbol{Q} \in \mathbb{S}^{N\times N}$ is a real symmetric matrix. Directly minimizing Eq. (8) is still intractable, and we instead derive its convex relaxation into a *Semidefinite Program (SDP)*. We solve the convex relaxation for a global optimum. The objective value associated with the global optimum gives a lower bound to the objective value of Eq. (8). Next, we convert the global optimum to a feasible solution of Eq. (8). In what follows, we first derive an intermediate problem, which is a relaxation (not necessarily convex) of Eq. (8). This intermediate problem plays the role of a bridge between Eq. (8) and its convex relaxation due to several of its nice properties, which we will describe shortly. Based on the properties of the intermediate problem we derive its convex relaxation, which is also a convex relaxation of Eq. (8).

To derive the intermediate problem, we first relax Eq. (8) by expanding its feasible region. The original feasible region of Eq. (8) is the set of vertices of a hypercube. We expand the original feasible region to the entire hypercube, which is defined by $C = \{\boldsymbol{s} | \boldsymbol{0} \le \boldsymbol{s} \le \boldsymbol{1}, \boldsymbol{s} \in \mathbb{R}^N\}$. We further expand C to the circumscribed sphere of the hypercube, which results in $\tilde{C} = \{\boldsymbol{s} | (\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1}) \le \frac{N}{4}, \boldsymbol{s} \in \mathbb{R}^N\}$. After the successive expansion we have the following *Quadratically Constrained Quadratic Programming* (QCQP), which was previously dubbed as the "intermediate problem":

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{s}^T\boldsymbol{b} + c_1 \\ s.t. \quad & (\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1}) \le \frac{N}{4}. \end{aligned} \tag{9}$$

The problem Eq. (9) is still non-convex, since in our problem setting the matrix $\boldsymbol{Q}$ is usually not positive (semi-)definite. However, Eq. (9) offers several benefits. First, it is a QCQP with only one inequality constraint, which indicates that it has a convex dual problem and under mild conditions (Slater's condition) strong duality holds [5]. This suggests that we can find the global optimum of a non-convex problem (when Slater's conditions hold) by solving its dual problem. Second, applying duality theory twice on Eq. (9) results in its own convex relaxation, which is therefore the convex relaxation of Eq. (8). In what follows we thereby derive the convex relaxation of Eq. (9).

We first obtain the Lagrangian $l(\boldsymbol{s}, \lambda)$ of Eq. (9) as follows, where $\lambda \ge 0$ is a Lagrangian multiplier:

$$\begin{aligned} l(\boldsymbol{s}, \lambda) :=& \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{b}^T\boldsymbol{s} + c_1 + \lambda\big[(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1}) - \frac{N}{4}\big] \\ =& \boldsymbol{s}^T(\boldsymbol{Q} + \lambda\boldsymbol{I})\boldsymbol{s} + (2\boldsymbol{b} - \lambda\boldsymbol{1})^T\boldsymbol{s} + c_1. \end{aligned} \tag{10}$$

The dual function $g(\lambda)$ is then

$$\begin{aligned} g(\lambda) =& \inf_{\boldsymbol{s}} l(\boldsymbol{s}, \lambda) \\ =& \begin{cases} c_1 - (\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1})^T(\boldsymbol{Q} + \lambda\boldsymbol{I})^\dagger(\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1}), & cond_1 \\ -\infty, & \text{o.w.} \end{cases} \end{aligned} \tag{11}$$

where $(\boldsymbol{Q} + \lambda\boldsymbol{I})^\dagger$ is the *Pseudo-Inverse* of $(\boldsymbol{Q} + \lambda\boldsymbol{I})$. Note that $cond_1$ consists of two conditions: first, that $\boldsymbol{Q} + \lambda\boldsymbol{I}$ is positive semi-definite and second, that $\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1}$ lies in the column space of $\boldsymbol{Q} + \lambda\boldsymbol{I}$. If the conditions in $cond_1$ are satisfied, maximizing $g(\lambda)$ is feasible and the primal problem is bounded. Otherwise, $g(\lambda)$ is unbounded below ($-\infty$), and we have a certificate that the primal problem in Eq. (9) is also unbounded. With $cond_1$ satisfied, we introduce a variable γ as the lower bound of $g(\lambda)$, which indicates $c_1 - (\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1})^T(\boldsymbol{Q} + \lambda\boldsymbol{I})^\dagger(\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1}) \ge \gamma$. Then maximizing $g(\lambda)$ is equivalent to maximizing γ. Further, by *Schur Complement* (and remember $(\boldsymbol{Q} + \lambda\boldsymbol{I}) \succeq 0$), the inequality $c_1 - (\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1})^T(\boldsymbol{Q} + \lambda\boldsymbol{I})^\dagger(\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1}) \ge \gamma$ is equivalently represented by a linear matrix inequality

$$\begin{bmatrix} \boldsymbol{Q} + \lambda\boldsymbol{I} & \boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1} \\ (\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1})^T & c_1 - \gamma \end{bmatrix} \succeq 0,$$

which enables us to represent the dual problem of Eq. (9) as a *Semidefinite Program* (SDP) with two variables, γ and λ:

$$\begin{aligned} \max_{\gamma,\lambda} \quad & \gamma \\ s.t. \quad & \lambda \ge 0 \\ & \begin{bmatrix} \boldsymbol{Q} + \lambda\boldsymbol{I} & \boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1} \\ (\boldsymbol{b} - \frac{\lambda}{2}\boldsymbol{1})^T & c_1 - \gamma \end{bmatrix} \succeq 0, \end{aligned} \tag{12}$$

As discussed above, applying duality theory twice to Eq. (9) results in its own convex relaxation. Consequently, we continue to derive the dual of Eq. (12). The Lagrangian $l(\gamma, \lambda, \boldsymbol{S}, \boldsymbol{s}, \alpha)$ of Eq.(12)

is calculated as follows, where $\boldsymbol{S} \in \mathbb{S}^N, \boldsymbol{s} \in \mathbb{R}^N, \begin{bmatrix} \boldsymbol{S} & \boldsymbol{s} \\ \boldsymbol{s}^T & 1 \end{bmatrix} \geq 0$ and $\alpha \geq 0$ are Lagrangian multipliers:

$$\begin{aligned} & l(\gamma, \lambda, \boldsymbol{S}, \boldsymbol{s}, \alpha) = \\ & -\gamma - \lambda\alpha - tr\left(\begin{bmatrix} \boldsymbol{Q} + \lambda \boldsymbol{I} & \boldsymbol{b} - \frac{\lambda}{2}\mathbf{1} \\ (\boldsymbol{b} - \frac{\lambda}{2}\mathbf{1})^T & c_1 - \gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{S} & \boldsymbol{s} \\ \boldsymbol{s}^T & 1 \end{bmatrix} \right) \\ & = -\gamma - \lambda\alpha - \\ & \underbrace{tr\left(\begin{bmatrix} (\boldsymbol{Q} + \lambda \boldsymbol{I})\boldsymbol{S} + (\boldsymbol{b} - \frac{\lambda}{2}\mathbf{1})\boldsymbol{s}^T & \cdots \\ \cdots & (\boldsymbol{b} - \frac{\lambda}{2}\mathbf{1})^T \boldsymbol{s} + c_1 - \gamma \end{bmatrix} \right)}_{\text{We only need to keep these block matrices on the diagonal}} \\ & = \lambda\big[-\alpha - tr(\boldsymbol{S}) + \mathbf{1}^T \boldsymbol{s} \big] - \big[tr(\boldsymbol{Q}\boldsymbol{S}) + 2\boldsymbol{b}^T \boldsymbol{s} + c_1 \big], \end{aligned} \tag{13}$$

where $tr(\cdot)$ is trace operator. Notice that $\lambda\big[-\alpha - tr(\boldsymbol{S}) + \mathbf{1}^T \boldsymbol{s} \big]$ is a linear function of λ, so $[-\alpha - tr(\boldsymbol{S}) + \mathbf{1}^T \boldsymbol{s}]$ must be zero, as otherwise the linear function can be minimized without bound. In addition, the Lagrangian multiplier α is greater than or equal to zero, so from $-\alpha - tr(\boldsymbol{S}) + \mathbf{1}^T \boldsymbol{s} = 0$ we have $tr(\boldsymbol{S}) - \mathbf{1}^T \boldsymbol{s} \leq 0$, which is denoted by $cond_2$. The dual function $g(\boldsymbol{S}, \boldsymbol{s})$ is then:

$$\begin{aligned} g(\boldsymbol{S}, \boldsymbol{s}) &= \inf_{\gamma, \lambda, \alpha} l(\gamma, \lambda, \boldsymbol{S}, \boldsymbol{s}, \alpha) \\ &= \begin{cases} -tr(\boldsymbol{Q}\boldsymbol{S}) - 2\boldsymbol{b}^T \boldsymbol{s} - c_1, & cond_2 \\ -\infty, & \text{o.w.} \end{cases} \end{aligned} \tag{14}$$

The dual problem of Eq. (12) is the minimization of $-g(\boldsymbol{S}, \boldsymbol{s})$, which can be represented as a SDP as follows:

$$\begin{aligned} \min_{\boldsymbol{S} \in \mathbb{S}^N, \boldsymbol{s} \in \mathbb{R}^N} \quad & tr(\boldsymbol{Q}\boldsymbol{S}) + 2\boldsymbol{b}^T \boldsymbol{s} + c_1 \\ s.t. \quad & tr(\boldsymbol{S}) - \mathbf{1}^T \boldsymbol{s} \leq 0 \\ & \begin{bmatrix} \boldsymbol{S} & \boldsymbol{s} \\ \boldsymbol{s}^T & 1 \end{bmatrix} \geq 0, \end{aligned} \tag{15}$$

In order to see the connections between Eq. (15) and Eq. (9), we first note that by *Schur Complement* the linear matrix inequality

$$\begin{bmatrix} \boldsymbol{S} & \boldsymbol{s} \\ \boldsymbol{s}^T & 1 \end{bmatrix} \geq 0$$

is equivalent to $\boldsymbol{S} \geq \boldsymbol{s}\boldsymbol{s}^T$. Therefore if we reduce the feasible region of Eq. (15) by enforcing the equality constraint $\boldsymbol{S} = \boldsymbol{s}\boldsymbol{s}^T$, and then utilize that

$$tr(\boldsymbol{Q}\boldsymbol{S}) = tr(\boldsymbol{Q}\boldsymbol{s}\boldsymbol{s}^T) = \boldsymbol{s}^T \boldsymbol{Q}\boldsymbol{s}$$

and

$$tr(\boldsymbol{S}) - \mathbf{1}^T \boldsymbol{s} \leq 0 \equiv (\boldsymbol{s} - \frac{1}{2}\mathbf{1})^T (\boldsymbol{s} - \frac{1}{2}\mathbf{1}) \leq \frac{N}{4},$$

we have an equivalent problem to Eq. (9). This shows that Eq. (15) is a convex relaxation of Eq. (9) and, therefore, a convex relaxation of Eq. (6).

We solve Eq.(15) for a global optimal solution, which is denoted by $(\boldsymbol{S}^*, \boldsymbol{s}^*)$. Then we apply Euclidean projection to convert $\boldsymbol{s}^*$ to a feasible solution of Eq. (6), which is denoted by $\bar{\boldsymbol{s}}^*$. We remove all nodes from the network with $\bar{\boldsymbol{s}}_i^* > 0.5$. We call our full algorithm MINT (Malicious In NeTwork), which is detailed in Algorithm 1:

Next we show with appropriate choice of the trade-off parameters the optimal value of Eq. (8) is upper- and lower-bounded by the optimal value of Eq. (15), which provides performance guarantee

Algorithm 1 MINT

1: **Input**: $\boldsymbol{Q}$, $\boldsymbol{b}$, c_1
2: Compute the global optimal solution $\boldsymbol{s}^*$ of Eq. (15)
3: Solve $\bar{\boldsymbol{s}}^* = \arg\min_{\hat{\boldsymbol{s}} \in C} ||\hat{\boldsymbol{s}} - \boldsymbol{s}^*||_2$
4: Remove all nodes with $\bar{\boldsymbol{s}}_i^* \geq 0.5$

for the SDP relaxation. We denote the optimal objective value of the originally intractable optimizatioin by V^* and the optimal objective value of the SDP relaxation by P^*_{SDP}. Then we have the following theorem:

Theorem 3.2. *When the (i, j)-th element of the matrix $\boldsymbol{Q}$ in Eq. (8) satisfying $q_{ij} \geq 0, \forall i \neq j$, the optimal objective value V^* is upper- and lower-bounded by the optimal objective value P^*_{SDP} up to a constant β:*

$$P^*_{SDP} \leq V^* \leq P^*_{SDP} + \beta$$

Proof. The proof is deferred to the Appendix. □

To understand the relation between the condition $q_{ij} \geq 0, \forall i \neq j$ and the choice of the trade-off parameters, we first note that $\forall i \neq j$:

$$q_{ij} = (\alpha_2 + \alpha_3)\big(\frac{\mu_i + \mu_j}{2} - \mathbb{E}[\mu_i \mu_j]\big) - \alpha_2,$$

where μ_i is the maliciousness probability of the i-th node. Then $q_{ij} \geq 0$ is equivalent to:

$$\frac{\mu_i + \mu_j}{2} \geq \mathbb{E}[\mu_i \mu_j] + \frac{\alpha_2}{\alpha_2 + \alpha_3}, \forall i \neq j. \tag{16}$$

The left-hand side of Eq. (16) consists of the maliciousness probabilities estimated from data, which can be thought as constants when we analyze the the behavior of the inequality. When $\mathbb{E}[\mu_i \mu_j]$ is large, the edge (i, j) is more likely to be a connection between a malicious node and a benign node, which means we would like a small α_2 that encourages cutting connections. Notice that a small α_2 is exactly what we need to make the inequality in Eq. (16) hold. Therefore the condition $q_{ij} \geq 0, \forall i \neq j$ indicates that the choice of the trade-off paramters is important to guarantee the performance of the SDP relaxation.

4 EXPERIMENTS

In this section we present experiments to show the effectiveness of our approach. We considered both synthetic and real-world network structures, but in all cases derived distribution over maliciousness of nodes $\mathcal{P}$ using real data. For synthetic network, we considered two types of network structures: Barabasi-Albert (BA) [4] and Watts-Strogatz networks (Small-World) [21]. BA is characterized by its power-law degree distribution, where the probability that a randomly selected node has k neighbors is proportional to k^{-r}. For both networks we generated instances with $N = 128$ nodes. For real-world networks, we used a network extracted from Facebook data [10] which consisted of 4039 nodes and 88234 edges. We experimented with randomly sampled sub-networks with $N = 500$ nodes.

In our experiments, we consider a simplified case where the maliciousness probabilities for nodes are independent. In addition, we assume that a single estimator (e.g., logistic regression) was trained to estimate the probability that a node is malicious based on

features from past data. Note that these assumptions are reasonable for the purpose of validating the effectiveness of our model, since the focus of our model is not how to estimate maliciousness probabilities. For more complex cases, for example, when maliciousness probabilities for nodes are correlated, more advanced techniques, such as Markov Random Fields, can be applied to estimate the maliciousness probabilities, but our general approach would not change.

In all of our experiments, we derived $\mathcal{P}$ from data as follows. We start with a dataset $\boldsymbol{D}$ which includes malicious and benign instances (the meaning of these designations is domain specific), and split it into three subsets: $\boldsymbol{D}_{train}$ (the training set), $\boldsymbol{D}_1$, and $\boldsymbol{D}_2$, with the ratio of $0.3 : 0.6 : 0.1$. Our first step is to learn a probabilistic predictor of maliciousness as a function of a feature vector x, $\hat{p}(x)$, on $\boldsymbol{D}_{train}$. Next, we randomly assign malicious and benign feature vectors from $\boldsymbol{D}_2$ to the nodes on the network, assigning 10% of nodes with malicious and 90% with benign feature vectors. For each node, we use its assigned feature vector x to obtain our *estimated* probability of this node being malicious, $\hat{p}(x)$; this gives us the estimated maliciousness probability distribution $\hat{\mathcal{P}}$. This is the distribution we use in MINT and the baseline approaches. However, to ensure that our evaluation is fair and reasonably represents realistic limitations of the knowledge of the true maliciousness distribution, we train another probabilistic predictor, $p(x)$, now using $\boldsymbol{D}_{train} \cup \boldsymbol{D}_1$. Applying this new predictor to the nodes and their assigned feature vectors, we now obtain a distribution $\mathcal{P}^*$ which we use to evaluate performance.

We conducted two sets of experiments. In the first set of experiments we used synthetic networks and used data from the Spam [11] dataset to learn the probabilistic maliciousness model $p(x)$, and thereby derive $\mathcal{P}$. The Spam dataset $\boldsymbol{D}$ consists of spam and non-spam instances along with their corresponding labels.

In the second set of experiments we used real-world networks from Facebook and used Hate Speech data [7] collected from Twitter to obtain $\mathcal{P}$ as discussed above. The Hate Speech dataset is a crowd-sourced dataset that contains three types of tweets: 1. hate speech tweets that express hatred against a targeted group of people; 2. offensive language tweets that appear to be rude, but do not explicitly promote hatred; and 3. normal tweets that neither promote hatret nor are offensive. We categorized this dataset into two classes in terms of whether a tweet represents Hate Speech, with the offensive language tweets categorized as non-Hate Speech. After categorization, the total number of tweets is 24783, of which 1430 are Hate Speech. We applied the same feature extraction techniques as Davidson et al. [7] to process the data.

Note that our second set of experiments makes use of *real data* for both the network and the node maliciousness distribution $\mathcal{P}$. Moreover, as noted by Waseem and Hovy [20], hate speech is widespread among Facebook users, and our second set of experiments can be viewed as studying the problem of identifying and potentially removing nodes from a social network who egregiously spew hate.

Baselines. We compared our algorithm (MINT) with LESS, a state-of-the-art approach for graph hypothesis testing, and a simple baseline which removes a node i if its maliciousness probability $p_i > \theta^*$, where θ^* is a specified threshold.

The algorithm LESS was proposed in Sharpnack et al. [18], and considers a related hypothesis testing problem. The null hypothesis is that each node in the graph is associated with a random variable sampled from the standard Gaussian $\mathcal{N}(0, 1)$, while the alternative hypothesis is that there is a fraction of nodes where the random variables associated with them are sampeld from $\mathcal{N}(\mu, 1)$ with μ other that 0 (in our interpretation, these are the malicious nodes). The algorithm LESS employs the generalized log-likelihood over a subset of nodes as a test statistic, and the hypothesis test is to find the subset that has the strongest evidence aginst the null hypothesis. We remove the subset of nodes found by LESS.

The simple baseline has a trade-off parameter α between false-positive rate (FPR) and false-negative rate (FNR) (in our experiments $\alpha = 0.5$). We select an optimal threshold θ^* that minimizes $\alpha FPR + (1 - \alpha)FNR$ on training data.

Experiment Results. The averaged losses for our first set of experiments where $\mathcal{P}$ was simulated from Spam data are shown in Table 1. The top table contains the results on BA networks and the bottom table contains the results on Small-World networks. Each row corresponds to a combination of trade-off parameters $(\alpha_1, \alpha_2, \alpha_3)$; for example, $(0.1, 0.2, 0.7)$ corresponds to $(\alpha_1 = 0.1, \alpha_2 = 0.2, \alpha_3 = 0.7)$. We experimented with four combinations of these: $(0.1, 0.2, 0.7)$, $(0.2, 0.7, 0.1)$, $(0.7, 0.2, 0.1)$, and $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Each number was obtained by averaging over 50 randomly generated network topologies. Table 1 shows that MINT has the lowest loss across all settings except $(0.1, 0.2, 0.7)$.

To delve into the results more, we present the box plots for the experimental results on BA networks in Figure 2. Note that as Table 1 indicates that LESS performs considerably worse than both MINT and, remarkably, even the simple baseline across all combinations of the trade-off parameters, and we omit its box plots. Just as we observed in the table, three of the four box plots show a substantial improvement of MINT over the baseline in three out of the four cases, with the lone exception being when the trade-off parameters are $(0.1, 0.2, 0.7)$, that is, when the importance of preserving links among benign nodes is relatively low. In this case, it is reasonable to expect that the value of considering the network topology is dominated by the first-order considerations of removing malicious nodes and keeping benign, already largely captured by our simple baseline. Thus, our machinery is unnecessary in such a case, as its primary value is when overall connectivity of the benign subnetwork is also a first-order consideration, as we expect it to be in social network settings. This value is borne out by the results in the three remaining plots in Figure 2, where the baseline clearly underperforms MINT. An interesting observation is that in the upper right and lower left cases the average losses of MINT are close to 0, which is actually the best value that the loss function in Eq.(6) can achieve. Considering that minimizing Eq.(6) is a NP-hard problem, our convex relaxation gives a high quality approximation in polynomial time. [1]

The box plots for the experimental results on Small-World networks are shown in Figure 3, where we now include LESS as it is more competitive in this case. The overall trend is similar to Figure 2. Moreover, the box plots reveal that, while MINT is better than the simple baseline that ignores network structure in the three of

[1] Solving the SDP relaxation Eq. (15) is in polynomial-time with interior-point method

BA			
	Baseline	LESS	MINT
(0.1,0.2,0.7)	**7.8403**	28.6337	16.9782
(0.2,0.7,0.1)	14.6207	82.0922	**1.8650**
(0.7,0.2,0.1)	6.7699	32.2678	**1.5342**
(1/3,1/3,1/3)	5.8533	44.1410	**4.3730**
Small-World			
	Baseline	LESS	MINT
(0.1,0.2,0.7)	**8.7965**	12.5336	24.7706
(0.2,0.7,0.1)	20.0915	4.0273	**2.9719**
(0.7,0.2,0.1)	8.2982	4.3518	**1.8324**
(1/3,1/3,1/3)	7.4418	7.4027	**4.8369**

Table 1: Experiments where $\hat{\mathcal{P}}$ and $\mathcal{P}^*$ were simulated from Spam data.

the four cases where network structure matters the most, its peformance appears comparable to LESS on average, but exhibits much less variance than LESS. This may be attributed to the fact that both MINT and LESS are approximately solving hard optimization problems, and the MINT algorithm consistently arrives at a good approximation of the optimal solution, while the approximation quality of LESS is more variable. In any case, this is particularly noteworthy given the fact that MINT dramatically outperforms LESS in terms of scalability, as we show below.

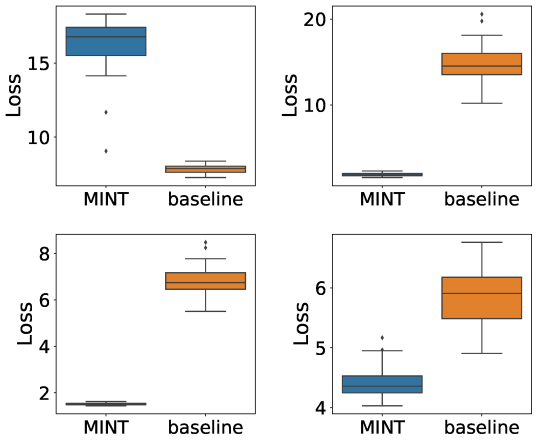

Figure 2: Experimental results on BA networks, where $\hat{\mathcal{P}}$ and $\mathcal{P}^*$ were simulated from Spam data. The averaged losses are reported in Table 1. Upper Left: $(0.1, 0.2, 0.7)$**; Upper Right:** $(0.2, 0.7, 0.1)$**; Lower Left:** $(0.7, 0.2, 0.1)$**; Lower Right:** $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$**. Each plot was averaged over** 50 **runs.**

Next, we evaluate the performance of MINT in our second set of experiments which use real data for both the network topology and to derive the maliciousness distribution (the latter using the Hate Speech dataset). In this case, LESS does not scale to the problem sizes we consider, and we only compare MINT to the simple baseline. The average losses are shown in Table 2, where each number was averaged over 50 randomly sampled sub-networks.

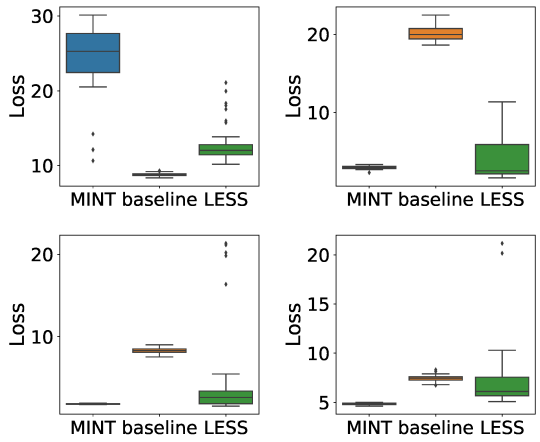

Figure 3: Experimental results on Small-World networks, where $\hat{\mathcal{P}}$ and $\mathcal{P}^*$ were simulated from Spam data. The averaged losses are reported in Table 1. Upper Left: $(0.1, 0.2, 0.7)$**; Upper Right:** $(0.2, 0.7, 0.1)$**; Lower Left:** $(0.7, 0.2, 0.1)$**; Lower Right:** $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$**. Each plot was averaged over** 50 **runs.**

The results demonstrate that MINT again significantly outperforms the baseline in all but one case in which the importance of cutting malicious links greatly outweighs other considerations. Note that when keeping benign nodes connected becomes more important than removing malicious nodes (e.g., when the trade-off parameters are $(0.2, 0.7, 0.1)$ and $(0.7, 0.2, 0.1)$), MINT surpasses the baseline by nearly an order of magnitude, which confirms that a simple baseline that trades off between false-positive rate and false-negative rate is not enough to take indirect harm into account.

Again, we present the comparison in greater depth using box plots in Figure 4. The overall trend is similar to other two boxplots. There is, however, one distinctive obervation that the dispersion of losses on Facebook networks is larger than the dispersion on BA networks. This observation likely results from the fact that the SDP relaxation Eq. (15) for Facebook networks is substantially looser than that for BA networks in the sense that it has more variables and constraints, which makes locating the exact optimal solution of Eq. (15) harder. Indeed, we were using interior-point method to solve Eq. (15) and there were a few cases where the maximum number of iterations was reached while the optimal solution had not been found. In any case, we still consistently observe performance improvement compared to the baseline even as we take this variance into account.

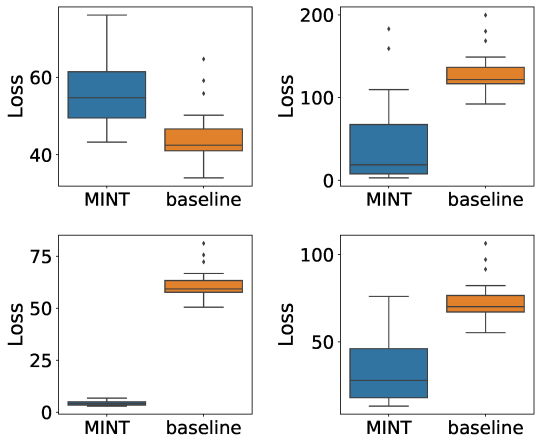

Figure 4: Experimental results on Facebook networks, where $\hat{\mathcal{P}}$ and $\mathcal{P}^*$ were simulated from Spam data. The averaged losses are reported in Table 1. Upper Left: $(0.1, 0.2, 0.7)$**; Upper Right:** $(0.2, 0.7, 0.1)$**; Lower Left:** $(0.7, 0.2, 0.1)$**; Lower Right:** $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$**. Each plot was averaged over** 50 **runs.**

	Baseline	MINT
(0.1,0.2,0.7)	**44.2784**	56.1550
(0.2,0.7,0.1)	128.1051	**41.7881**
(0.7,0.2,0.1)	60.7507	**5.9065**
(1/3,1/3,1/3)	72.3060	**39.5743**

Table 2: Experiments where $\hat{\mathcal{P}}$ and $\mathcal{P}^*$ were simulated from Hate Speech data, using Facebook network data. All the differences are significant.

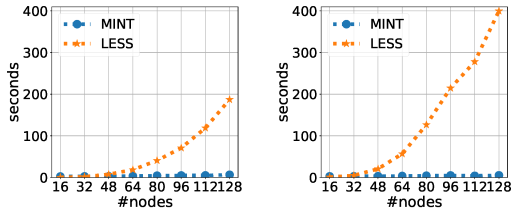

Figure 5: Running time averaged over 15 trials. Left: BA, Right: Small-World.

Next, we compare the running time of LESS and MINT as a function of the number of nodes on the network in Figure 5 for the case where $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$; alternatives generated similar results. Each point in Figure 5 was averaged over 15 trials. The experiments were conducted on a desktop (OS: Ubuntu 14.04; CPU: Intel i7 4GHz 8-core; Memory: 32GB). We can see that MINT is significantly faster than LESS, with the difference increasing in the network size. Indeed, LESS becomes impractical for realistic network sizes, whereas MINT remains quite scalable.

Recall that while MINT assumes knowledge of the distribution $\hat{\mathcal{P}}$, our evaluation above used a simulated ground-truth distribution $\mathcal{P}^*$, thereby capturing the realistic consideration that MINT would be applied using an estimated, rather than actual, distribution. Nevertheless, we now study the sensitivity of MINT to estimation error more systematically. Specifically, we added Gaussian noise $\mathcal{N}(0, \sigma)$ to each estimated malicious probability p_i, which results in $\tilde{\mathcal{P}}$. We varied σ from 0.1 to 0.5. Then we ran MINT on $\tilde{\mathcal{P}}$ and evaluated it on $\mathcal{P}^*$. We used Spam data to simulate $\hat{\mathcal{P}}$ and $\mathcal{P}^*$, and conducted experiments on BA and Small-World network structures. We focused on a specific setting where $(\alpha_1 = 0.1, \alpha_2 = 0.2, \alpha_3 = 0.7)$. Other combinations of weight parameters generated similar results.

The results on BA networks (Figure 6 Left) show that performance of MINT does not significantly degrade even as we introduce a substantial amount of noise, which indicats that MINT is robust against estimation error. The results on Small-World networks, on the other hand, do show that MINT exhibits some degradation with increasing σ. However, even in this case degradation is relatively slow. Altogether, our experiments suggest that MINT is quite robust to estimation error.

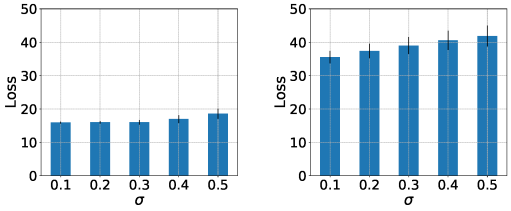

Figure 6: Sensitivity analysis of MINT. Each bar was averaged over 15 **runs. Left: BA. Right: Small-World**

5 CONCLUSION

We considered the problem of removing malicious nodes from a network under uncertainty. We designed a model (loss function) that considers both the likelihood that a node is malicious, as well as the network structure. Our key insight is for the loss function to capture both the *direct* loss associated with false positives and the *indirect* loss associated with cutting connections between benign nodes, and failing to cut connections from malicious nodes to their benign network neighbors. We first showed that this optimization problem is NP-Hard. Nevertheless, we proposed an approach based on convex relaxation of the loss function, which is quite tractable in practice. Finally, we experimentally showed that our algorithm outperforms alternative approaches in terms of loss, including both a simple baseline that trades off only the direct loss (false positives and false negatives) and a state-of-the-art approach, LESS, which uses a graph scan statistic. Moreover, our method is significantly faster than the LESS algorithm.

ACKNOWLEDGEMENT

This research was partially supported by the National Science Foundation (IIS-1905558) and Army Research Office (W911NF-16-1-0069 and MURI W911NF-18-1-0208).

APPENDIX

Proof of Theorem 3.2

Proof. It is direct to show $P^*_{SDP} \leq V^*$, since Eq.(15) is the dual of Eq.(12) which is the dual of Eq. (9). Our main focus is to prove $V^* \leq P^*_{SDP} + \beta$. Notice that we have:

$$V^* \leq \mathbb{E}\big[\boldsymbol{s}^T \boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{s}^T \boldsymbol{b} + c_1\big], \tag{17}$$

which is true by taking expectations in both sides of $V^* \leq \boldsymbol{s}^T \boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{s}^T \boldsymbol{b} + c_1$ which holds since by definition V^* is the optimal objective value of the right-hand side. Then we apply the *randomization procedure* introduced in [9] to sample feasible solutions for Eq. (8). Specifically, solving Eq. (15) results in a positive-semidefinite matrix $\boldsymbol{S}^*$. We define $\bar{\boldsymbol{S}}^* = (\boldsymbol{S}^* + I)$, which is a positive-definite matrix. By the Cholesky decomposition of $\bar{\boldsymbol{S}}^*$ we have:

$$\bar{\boldsymbol{S}}^* = \boldsymbol{V}^T \boldsymbol{V}.$$

We normalize each column of $\boldsymbol{V}$ to have unit length, which results in a set of vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N$:

$$\boldsymbol{V} = \big[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N\big], ||\boldsymbol{v}_i||_2 = 1 \; \forall i$$

Note that $\bar{S}^*_{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$, where $\bar{S}^*_{ij}$ is the ij-th element of $\bar{\boldsymbol{S}}^*$. Now we randomly sample $\boldsymbol{s} \in \{0,1\}^N$, where the i-th element of $\boldsymbol{s}$ is generated as follow:

$$s_i = \begin{cases} 1, & \boldsymbol{v}_i^T \boldsymbol{z} > 0 \\ 0, & \boldsymbol{v}_i^T \boldsymbol{z} \leq 0. \end{cases}$$

The random vector $\boldsymbol{z} \in \mathbb{R}^N$ is sampled from a Gaussian distribution, $\boldsymbol{z} \sim \mathcal{N}(\frac{1}{2}\boldsymbol{1}, \boldsymbol{I})$, where $\boldsymbol{I}$ is the identity matrix in $\mathbb{R}^{N \times N}$. The expectation of s_i is 0.5, since the probabilities of $\boldsymbol{v}_i^T \boldsymbol{z} \geq 0$ and $\boldsymbol{v}_i^T \boldsymbol{z} < 0$ are equal to 0.5, respectively. We let $\boldsymbol{\mu}$ denote the expectation of $\boldsymbol{s}$, so $\boldsymbol{\mu} = \frac{1}{2}\boldsymbol{1}$. The covariance matrix Σ of the random vector $\boldsymbol{s}$ is:

$$\begin{aligned} \Sigma &= \mathbb{E}\big[(\boldsymbol{s} - \boldsymbol{\mu})(\boldsymbol{s} - \boldsymbol{\mu})^T\big] \\ &= \mathbb{E}[\boldsymbol{s}\boldsymbol{s}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T \\ &= \mathbb{E}[\boldsymbol{s}\boldsymbol{s}^T] - \frac{1}{4}J(N,N), \end{aligned}$$

where $J(N,N) \in \mathbb{R}^{N \times N}$ is a matrix with all elements equal to 1. Notice that the (i,j)-th element of the expectation $\mathbb{E}[\boldsymbol{s}\boldsymbol{s}^T]$ is $\mathbb{E}[s_i s_j]$, which is equal to the probability that both s_i and s_j are equal to 1:

$$\begin{aligned} \mathbb{E}[s_i s_j] &= P(s_i = 1, s_j = 1) \\ &= P\big(\boldsymbol{v}_i^T \boldsymbol{z} > 0, \boldsymbol{v}_j^T \boldsymbol{z} > 0\big). \end{aligned} \tag{18}$$

Following the similar argument as Lemma 2.2 of [9] we know:

$$P\big(\boldsymbol{v}_i^T \boldsymbol{z} > 0, \boldsymbol{v}_j^T \boldsymbol{z} > 0\big) = \frac{\pi - \theta}{2\pi},$$

where $\theta = arccos(\boldsymbol{v}_i^T \boldsymbol{v}_j)$. Therefore we have:

$$\begin{aligned} \mathbb{E}[s_i s_j] &= \frac{1}{2} - \frac{\theta}{2\pi} \\ &= \frac{1}{2} - \frac{arccos(\boldsymbol{v}_i^T \boldsymbol{v}_j)}{2\pi} \\ &= \frac{1}{2} - \frac{arccos(\bar{S}^*_{ij})}{2\pi}. \end{aligned}$$

Now we expand the Eq. (17) as the following:

$$\begin{aligned} V^* &\leq \mathbb{E}\big[\boldsymbol{s}^T \boldsymbol{Q}\boldsymbol{s} + 2\boldsymbol{s}^T \boldsymbol{b} + c_1\big] \\ &= \mathbb{E}\big[(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T \boldsymbol{Q}(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1}) + 2\boldsymbol{s}^T \boldsymbol{1} + c_1\big] + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= \mathbb{E}\big[tr\big(\boldsymbol{Q}(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T\big)\big] + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= tr\Big(\boldsymbol{Q}\mathbb{E}\big[(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})(\boldsymbol{s} - \frac{1}{2}\boldsymbol{1})^T\big]\Big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= tr\big(\boldsymbol{Q}\Sigma\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= \sum_{i,j} q_{ij}\Sigma_{ji} + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= \sum_{i,j} q_{ij}\big(\mathbb{E}[s_i s_j] - \frac{1}{4}\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= \sum_{i,j} q_{ij}\big(\frac{1}{4} - \frac{arccos(\bar{S}^*_{ij})}{2\pi}\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &\overset{(*)}{\leq} \sum_{i,j} q_{ij}\bar{S}^*_{ij} + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &\quad \text{(the diagonal elements of } \boldsymbol{Q} \text{ are zeros)} \\ &= \sum_{i,j} q_{ij}S^*_{ij} + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &= tr\big(\boldsymbol{Q}\boldsymbol{S}^*\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1}, \end{aligned}$$

where $(*)$ is true because the inequality:

$$\frac{1}{4} - \frac{arccos(x)}{2\pi} \leq x$$

holds on $0 \leq x \leq 1$ and $q_{ij} \geq 0, \forall i, j$. Since:

$$P^*_{SDP} \leq V^* \leq tr\big(\boldsymbol{Q}\boldsymbol{S}^*\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1},$$

we select a nonnegative constant β such that:

$$\begin{aligned} \beta &\geq \big[tr\big(\boldsymbol{Q}\boldsymbol{S}^*\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1}\big] - P^*_{SDP} \\ &= \boldsymbol{1}^T \boldsymbol{b} + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} - 2\boldsymbol{b}^T \boldsymbol{s}^*, \end{aligned}$$

where $\boldsymbol{s}^*$ is the optimal solution of Eq. (15). Note that since the elements of $\boldsymbol{b}$ and $\boldsymbol{Q}$ are bounded, we can always find such a β. With β we have:

$$\begin{aligned} V^* &\leq tr\big(\boldsymbol{Q}\boldsymbol{S}^*\big) + \boldsymbol{1}^T \boldsymbol{b} + c_1 + \frac{1}{4}\boldsymbol{1}^T \boldsymbol{Q}\boldsymbol{1} \\ &\leq tr\big(\boldsymbol{Q}\boldsymbol{S}^*\big) + 2\boldsymbol{b}^T \boldsymbol{s}^* + c_1 + \beta \\ &= P^*_{SDP} + \beta, \end{aligned}$$

which completes the proof. □

REFERENCES

[1] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36, 2017.

[2] Vinicius Andrade. Facebook, whatsapp step up efforts in brazil's fake news battle. *Bloomberg*. URL https://www.bloomberg.com/news/articles/2018-10-23/facebook-whatsapp-step-up-efforts-in-brazil-s-fake-news-battle.

[3] Ery Arias-Castro, Emmanuel J Candes, and Arnaud Durand. Detection of an anomalous cluster in a network. *The Annals of Statistics*, pages 278–304, 2011.

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

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. In *ICWSM*, pages 61–70, 2015.

[7] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*, 2017.

[8] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[9] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[10] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[11] Moshe Lichman et al. Uci machine learning repository, 2013.

[12] Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8: 935–983, 2007.

[13] Yilin Mo, Tiffany Hyun-Jin Kim, Kenneth Brancik, Dona Dickinson, Heejo Lee, Adrian Perrig, and Bruno Sinopoli. Cyber–physical security of a smart grid infrastructure. *Proceedings of the IEEE*, 100(1):195–209, 2012.

[14] Vidya Narayanan, Vlad Barash, John Kelly, Bence Kollanyi, Lisa-Maria Neudert, and Philip N Howard. Polarization, partisanship and junk news consumption over social media in the us. *arXiv preprint arXiv:1803.01845*, 2018.

[15] Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[16] Jesus Rodriguez. Facebook suspends 115 accounts for 'inauthentic behavior' as polls open. URL https://www.politico.com/story/2018/11/06/facebook-suspends-accounts-polls-2018-964325.

[17] Shane Scott and Mike Isaac. Facebook says it's policing fake accounts. but they're still easy to spot. *The New York Times*. URL https://www.nytimes.com/2017/11/03/technology/facebook-fake-accounts.html.

[18] James L Sharpnack, Akshay Krishnamurthy, and Aarti Singh. Near-optimal anomaly detection in graphs using lovasz extended scan statistic. In *Advances in Neural Information Processing Systems*, pages 1959–1967, 2013.

[19] Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative markov networks. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[20] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, 2016.

[21] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440, 1998.

[22] Yang Yang, Takashi Nishikawa, and Adilson E. Motter. Small vulnerable sets determine large network cascades in power grids. *Science*, 358(886), 2017.