POTION : Optimizing Graph Structure for Targeted Diffusion

Sixie Yu * [†1], Leo Torres[‡2], Scott Alfeld[§3], Tina Eliassi-Rad [¶2], and Yevgeniy Vorobeychik [‖1]

[1]Washington University in St. Louis
[2]Northeastern University
[3]Amherst College

Abstract

The problem of diffusion control on networks has been extensively studied, with applications ranging from marketing to controlling infectious disease. However, in many applications, such as cybersecurity, an attacker may want to attack a *targeted* subgraph of a network, while limiting the impact on the rest of the network in order to remain undetected. We present a model POTION in which the principal aim is to optimize graph structure to achieve such targeted attacks. We propose an algorithm POTION-ALG for solving the model at scale, using a gradient-based approach that leverages Rayleigh quotients and pseudospectrum theory. In addition, we present a condition for certifying that a targeted subgraph is immune to such attacks. Finally, we demonstrate the effectiveness of our approach through experiments on real and synthetic networks.

1 Introduction

Many diverse phenomena that propagate through a network, such as epidemic spread, cascading failures, and chemical reactions, can be modeled by network diffusion models [3, 5, 23, 41, 38]. The problem of controlling diffusion has, as a result, received much attention in the literature, with primary focus on two mechanisms for control: the choice of initial nodes to start the spread [14, 7, 43], and the modification of network structure [16, 33, 42, 34]. To date, most work on diffusion control (either promotion or inhibition) has considered diffusion over the entire network. However, in many problems, the focus is instead on diffusion that is *targeted* to a particular subgraph of the network. For example, in cybersecurity, diffusion commonly represents malware spread, but malware attacks are often targeted at particular subsets of critical devices [12], which should be accounted for when modeling attacking behavior. Congestion cascades of ground traffic or flight networks are other examples, where the goal of resilience may be to ensure that cascades concentrate on a subset of high-capacity nodes that can handle them, limiting the impact on the rest of the network [10, 9]. In another domain, medical treatments for certain diseases such as cancer may leverage a molecular signaling network, with the goal of targeting just the pathogenic portion of it, while limiting the deleterious effects on the rest [39].

We study the problem of targeted diffusion in which an attacker[1] can modify the graph structure $G = (\mathcal{V}, \mathcal{E})$ to achieve two goals: 1) maximize the diffusion spread to a target subgraph $G_{\mathcal{S}}$, and 2) minimize the impact on the remaining graph $G \setminus G_{\mathcal{S}}$. We capture the first goal by maximizing a utility function that incorporates spectral information of the adjacency matrix of G, specifically its largest (in magnitude) eigenvalue, eigenvector centrality, and the normalized cut of the target subgraph. The second goal is achieved by limiting the modifications made outside of the target subgraph. We present a scalable algorithmic framework for solving this problem. Our framework leverages a combination of gradient ascent with the use of Rayleigh quotients and pseudospectrum theory, which yields differentiable approximations of our objective and allows us to avoid projection steps that would otherwise be costly and imprecise. Moreover, we derive a condition that enables us to certify if a network is robust against a broad class of targeted diffusion attacks. Finally, we demonstrate the effectiveness of our approach through extensive experiments.

In summary, our contributions are:

1. We propose POTION (oPtimizing graph structures fOr Targeted diffusION): a model for targeted diffusion attack by optimizing graph structures.

*The first two authors contributed equally to the paper.
†sixie.yu@wustl.edu.
‡leo@leotrs.com
§salfeld@amherst.edu
¶t.eliassirad@northeastern.edu
‖yvorobeychik@wustl.edu

[1]The attacker is the agent who initiates diffusion.

2. We present POTION-ALG : an efficient algorithm to optimize POTION by leveraging Rayleigh quotient and pseudospectrum theory.

3. We describe a condition for certifying that a targeted subgraph is immune to such attacks.

4. We demonstrate the effectiveness and efficiency of POTION and POTION-ALG on synthetic and real-world networks; and against baseline and competing methods.

2 Related Work

Various dynamical processes can be modeled as diffusion dynamics on networks, including the spread of infectious diseases [3, 5], cascading failures in infrastructure networks [23, 41], and information spread (e.g., rumors, fake news) on social networks [19, 20]. One line of research assesses the impact of cascading failures. Yang et al. [41] simulated cascading failures to quantify the vulnerability of the power grid in North America. Fleurquin et al. [10] studied the impact of flight delays as a cascading failure diffusing through the network. Motter and Lai [23] investigated the cascading failures on a network due to the malfunction of a single node. Another line of research concerns diffusion control, for example, selecting a set of nodes such that if the diffusion originated from them, it reaches as many nodes as possible [14, 7, 43]; or modifying network structures to increase or limit some diffusion [33, 28]. However, these lines of research do not differentiate between targeted and non-targeted nodes. Ho et al. [13] studied targeted diffusion controlled by changing nodal status. We focus on the problem where an attacker manipulates underlying network structures in order to achieve targeted diffusion.

Another relevant research thread is network design, which is the problem of modifying network structure to induce certain desirable outcomes. Some prior work [33, 42, 31] considered the containment of spreading dynamics by adding or removing nodes or edges from the network, while others [37, 29, 16, 6, 34] considered limiting the spread of infectious disease by minimizing the largest eigenvalue of the network. Kempe et al. [15] studied modifying network structure to induce certain outcomes from a game-theoretic perspective, but they did not consider diffusion dynamics. Others have studied the problem of manipulating node centrality measures (e.g., eigenvector or PageRank centrality) [2, 1] or node similarity measures (e.g., Katz similarity) [44] through edge perturbation. All of these prior efforts focus on the impact either at the network level or at the node-level properties, while our focus is on the impact of diffusion dynamics on a targeted subgraph of the network.

3 POTION : Proposed Model

We present a model for targeted diffusion through graph structure optimization. We refer to the agent who initiates diffusion as *the attacker*. We use cybersecurity as a running example. Here the attacker initiates the diffusion (e.g., the spread of malware) on a network of computers. We define the impact of the diffusion as the number of infected nodes (e.g., compromised with malware). The attacker has two objectives: 1) she wishes to maximize the impact of the diffusion on a targeted set of nodes (e.g., computing nodes with access to critical assets), and 2) to limit the impact on non-targeted nodes to ensure stealth [12].

Let $G = (\mathcal{V}, \mathcal{E})$ be a connected, weighted or unweighted, undirected graph with no self-loops. Let $n = |\mathcal{V}|$ be the number of nodes in G and $\boldsymbol{A}$ be its adjacency matrix. Throughout this paper, the eigenvalues of $\boldsymbol{A}$ are ranked in descending order $\lambda_1(\boldsymbol{A}) \geq \cdots \geq \lambda_n(\boldsymbol{A})$. Suppose the attacker targets a subgraph $G_{\mathcal{S}}$ where $\mathcal{S} \subseteq \mathcal{V}$ is the node set of $G_{\mathcal{S}}$. Let $\mathcal{S}' = \mathcal{V} \setminus \mathcal{S}$, and its induced subgraph $G_{\mathcal{S}'}$. Throughout the paper we assume $G_{\mathcal{S}}$ is connected, and denote its adjacency matrix by $\boldsymbol{A}_{\mathcal{S}}$. To achieve her objectives, the attacker modifies the structure of G. The modified graph and targeted subgraph are represented by $\tilde{G}$ and $\tilde{G}_{\mathcal{S}}$, respectively. Formally, the attacker's action is to add a perturbation $\boldsymbol{\Delta} \in \mathbb{R}^{n \times n}$ to $\boldsymbol{A}$, which results in the perturbed adjacency matrix $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\Delta}$. The adjacency matrix of $\tilde{G}_{\mathcal{S}}$ is denoted by $\tilde{\boldsymbol{A}}_{\mathcal{S}}$.

3.1 Diffusion Dynamics The status of a node is modeled by the well-known SIS (Susceptible-Infected-Susceptible) diffusion dynamics, where it alternates between "infected" and "susceptible". [2] Due to the malware spread by infected neighbors, a susceptible node becomes infected with probability β. An infected node becomes susceptible again (e.g., malware is removed) with probability δ. Following Chakrabarti et al.[5], this process is modeled by a nonlinear dynamical system. Let π_i be the probability of node i becoming infected (e.g., compromised with malware) in the steady state of this dynamical system, with $\boldsymbol{\pi}$ the vector of these probabilities. A key result in [5] is that when $\lambda_1(\boldsymbol{A}) < \delta/\beta$ the system converges to the steady state $\boldsymbol{\pi} = \mathbf{0}$, which implies that the diffusion process quickly dies out. However, when $\lambda_1(\boldsymbol{A}) \geq \delta/\beta$ the system converges to another steady state $\boldsymbol{\pi} \neq \mathbf{0}$. We leverage this connection between graph structure, dynamical model of epidemic

[2] Due to brevity, a discussion on generalization of our approach to other diffusion dynamics is at `https://arxiv.org/abs/2008.05589`.

spread, and the epidemic threshold, in constructing our *threat model*, as discussed next.

3.2 Threat Model Maximizing the Impact on $G_{\mathcal{S}}$: To maximize the impact of diffusion on $G_{\mathcal{S}}$, the attacker has two goals: 1) ensure that epidemics starting in $G_{\mathcal{S}}$ spread rather than die out, and 2) ensure that epidemics starting outside $G_{\mathcal{S}}$ are likely to reach it. We capture the first goal by maximizing the largest (in modulus) eigenvalue of $G_{\mathcal{S}}$, $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$, which corresponds to the epidemic threshold of the targeted subgraph.[3] The second goal is captured by maximizing the *normalized cut* of $G_{\mathcal{S}}$, $\phi(\mathcal{S})$, where $\mathcal{S}$ is the set of nodes in $G_{\mathcal{S}}$ and $\mathcal{S}'$ are the nodes in the remaining graph. The normalized cut is formally defined as follows:

$$\phi(\mathcal{S}) = \text{cut}(\mathcal{S}, \mathcal{S}') \left(\frac{1}{\text{vol}(\mathcal{S})} + \frac{1}{\text{vol}(\mathcal{S}')} \right), \tag{3.1}$$

where $\text{cut}(\mathcal{S}, \mathcal{S}')$ is the sum of the weights on the edges across $\mathcal{S}$ and $\mathcal{S}'$ (unit weights for unweighted graphs), and $\text{vol}(\mathcal{S})$ (resp. $\text{vol}(\mathcal{S}')$) is the sum of degrees of the nodes in $\mathcal{S}$ (resp. $\mathcal{S}'$). The formal rationale for using the normalized cut is based on Meila et al. [21], which showed that increasing $\phi(\mathcal{S})$ increases the probability that a random walker transitions from $\mathcal{S}'$ to $\mathcal{S}$, if we assume that $G_{\mathcal{S}}$ is smaller than $G_{\mathcal{S}'}$.

Limiting the Impact on $G_{\mathcal{S}'}$: Another important objective of the attacker is to limit the impact on $G_{\mathcal{S}'}$, the non-targeted part of the graph. We capture this goal in two different ways. First, by limiting the likelihood of the epidemic spreading to $G_{\mathcal{S}'}$, which we define as minimizing the impact $I(G_{\mathcal{S}'}) = \sum_{i \in \mathcal{S}'} \pi_i$. Second, by limiting the impact on the spectrum of $\boldsymbol{A}$.

We now demonstrate that minimizing $I(G_{\mathcal{S}'})$ is approximately equivalent to minimizing the eigenvector centrality of $\mathcal{S}'$. Let $\boldsymbol{P}^t$ be the global configuration of the graph at time step t, where P_i^t is the probability that node i is infected (e.g., compromised with malware). Following Mieghem et al. [22] , ignoring higher-order terms and taking the time step to be infinitesimally small, the dynamics of P_i^t is modeled as the following:

$$\frac{dP_i^t}{dt} = \sum_{j \in \mathcal{V}} \beta \tilde{A}_{ij} P_j^t - \delta P_i^t. \tag{3.2}$$

Here, we can think of the two terms on the right side as two competing forces. The first term is the force contributed by the infected neighbors of node i (which increases P_i^t), while the second term is the force due to i's self recovery (which decreases P_i^t). Rewriting in matrix notation yields:

$$\frac{d\boldsymbol{P}^t}{dt} = \left[\beta \tilde{\boldsymbol{A}} - \delta \boldsymbol{I}\right] \boldsymbol{P}^t, \tag{3.3}$$

which gives a linear approximation to the non-linear dynamical system proposed in [5]. The steady state $\boldsymbol{\pi}$ must satisfy $\left[\beta \tilde{\boldsymbol{A}} - \delta \boldsymbol{I}\right] \boldsymbol{\pi} = \boldsymbol{0}$, which is equivalent to $\tilde{\boldsymbol{A}} \boldsymbol{\pi} = (\delta/\beta) \boldsymbol{\pi}$. Suppose $\lambda_1(\tilde{\boldsymbol{A}}) = \delta/\beta$, and $\boldsymbol{\pi}$ is the corresponding eigenvector. Let $\tilde{\boldsymbol{v}}_1$ be the unit eigenvector associated with $\lambda_1(\tilde{\boldsymbol{A}})$. Let $\sigma(\mathcal{S}) = \sum_{j \in \mathcal{S}} \tilde{v}_1[j]$ be the eigenvector centrality of $\mathcal{S}$. Noting that $\boldsymbol{\pi}$ may differ from $\tilde{\boldsymbol{v}}_1$ by up to a multiplicative constant c, the impact on $G_{\mathcal{S}'}$ can be approximated as:

$$I(G_{\mathcal{S}'}) = \sum_{j \in \mathcal{S}'} \pi_j \approx c \sum_{j \in \mathcal{S}'} \tilde{v}_1[j] = c\big(1 - \sigma(\mathcal{S})\big), \tag{3.4}$$

where the last equality is because $\mathcal{S}$ and $\mathcal{S}'$ are disjoint and $\tilde{\boldsymbol{v}}_1$ is an unit vector. Thus, minimizing the impact on $G_{\mathcal{S}'}$ is approximately equivalent to maximizing the eigenvector centrality of $\mathcal{S}$.

Recall that to have an epidemic spread, one needs $\lambda_1(\tilde{\boldsymbol{A}}) \geq \delta/\beta$. Here, we assumed $\lambda_1(\tilde{\boldsymbol{A}}) = \delta/\beta$. In Section 6, we demonstrate that our analysis yields an approach that is effective even when this assumption fails to hold (i.e., when $\lambda_1(\tilde{\boldsymbol{A}}) > \delta/\beta$).

Now we focus on limiting the impact on the spectrum of $\boldsymbol{A}$. [4] Let $\epsilon > 0$ be the attacker's budget. Formally, this notion is captured through the following constraints:

$$|\lambda_i(\tilde{\boldsymbol{A}}) - \lambda_i(\boldsymbol{A})| \leq \epsilon,\ i = 1, \ldots, n. \tag{3.5}$$

In summary, the principal aims to (i) maximize the impact on $G_{\mathcal{S}}$ through maximizing $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ while (ii) limiting the impact on $G_{\mathcal{S}'}$ by maximizing the eigenvector centrality $\sigma(\mathcal{S})$, and satisfying Eq. (3.5). Formally, the principal aims to solve the following optimization problem:

$$\begin{aligned} \max_{\tilde{\boldsymbol{A}}} \quad & \alpha_1 \lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}}) + \alpha_2 \sigma(\mathcal{S}) + \alpha_3 \phi(\mathcal{S}) \\ s.t. \quad & \tilde{\boldsymbol{A}} \in \mathcal{P} = \left\{ \tilde{\boldsymbol{A}} \;\middle|\; \begin{array}{l} |\lambda_i(\tilde{\boldsymbol{A}}) - \lambda_i(\boldsymbol{A})| \leq \epsilon,\ i = 1, \ldots, n, \\ \tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top, \tilde{\boldsymbol{A}}_{ii} = 0,\ \forall i = 1, \ldots, n \end{array} \right\}, \end{aligned} \tag{3.6}$$

where the relative importance of the terms is balanced by the nonnegative constants $\alpha_1, \alpha_2, \alpha_3$, and the restrictions $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top$ and $\tilde{\boldsymbol{A}}_{ii} = 0$, $\forall i = 1, \ldots, n$ ensure that $\tilde{\boldsymbol{A}}$ is a valid adjacency matrix.

[3] If $G_{\mathcal{S}}$ is not connected, we may replace $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ by the largest eigenvalue of the largest connected component of $G_{\mathcal{S}}$.

[4] In cybersecurity, there are natural interpretations of an attack's stealth. For further details, see the extended version at `https://arxiv.org/abs/2008.05589`.

4 POTION-ALG : Proposed Algorithm

To solve the optimization problem in Eq. (3.6), a natural approach would be to use a form of projected gradient ascent. There are, however, two major hurdles to this basic approach: 1) the objective function involves terms that do not have an explicit functional representation in the decision variables, and 2) the projection step is quite expensive, as it involves projecting into a spectral norm ball, which entails an expensive SVD operation [17]. We address these challenges in Algorithm 1, which is our gradient-based solution to the attacker's optimization problem as described in Eq. (3.6).

Algorithm 1 POTION-ALG

1: **Input**: $\boldsymbol{A}, \epsilon, \{\eta_i\}_{i=1} \triangleright \{\eta_i\}_{i=1}$ is a schedule of step sizes
2: **Initialize**: $i = 1, \tilde{\boldsymbol{A}}_1 = \boldsymbol{A}, B_1 = 0$ $\triangleright$ B_i: the amount of budget used just before step i
3: **while** True **do**
4: Set $\boldsymbol{\Delta}_i$ to the gradient of $\alpha_1\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})+\alpha_2\sigma(\mathcal{S})+\alpha_3\phi(\mathcal{S})$ w.r.t. to $\tilde{\boldsymbol{A}}_i$
5: Set the diagonal entries of $\boldsymbol{\Delta}_i$ to zeros
6: **if** $\|\boldsymbol{\Delta}_i\| = \mathbf{0}$ **then** $\triangleright$ a local optimum is found
7: return $\tilde{\boldsymbol{A}}_i$
8: **end if**
9: **if** $B_i + ||\eta_i\boldsymbol{\Delta}_i||_2 \leq \epsilon$ **then** $\triangleright$ one-step look ahead
10: $\tilde{\boldsymbol{A}}_{i+1} = \tilde{\boldsymbol{A}}_i+\eta_i\boldsymbol{\Delta}_i$, $B_{i+1} = B_i+\|\eta_i\boldsymbol{\Delta}_i\|_2$, $i = i+1$
11: **else**
12: return $\tilde{\boldsymbol{A}}_i$
13: **end if**
14: **end while**

A key step of Algorithm 1 is line 4, where we compute the gradient of the attacker's utility function with respect to $\tilde{\boldsymbol{A}}$. This gradient involves terms that do not have an explicit functional form in terms of the decision variable, and we deal with each of these in turn.

First, consider the gradient of the normalized cut $\phi(\mathcal{S})$ w.r.t. $\tilde{\boldsymbol{A}}$. Let $\boldsymbol{x}_{\mathcal{S}}$ be the characteristic vector of $\mathcal{S}$, that is $x_{\mathcal{S}}[i] = 1$ iff $i \in \mathcal{S}$. Let $\tilde{\boldsymbol{D}}$ be the diagonal degree matrix $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$, and let $\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{A}} - \tilde{\boldsymbol{D}}$ be the Laplacian matrix. Using $\tilde{\boldsymbol{D}}$ and $\tilde{\boldsymbol{L}}$ to express $\mathrm{vol}(\mathcal{S})$ and $\mathrm{cut}(\mathcal{S}, \mathcal{S}')$, respectively, we have:

$$\phi(\mathcal{S}) = \boldsymbol{x}_{\mathcal{S}}^\top \tilde{\boldsymbol{L}} \boldsymbol{x}_{\mathcal{S}} \left(\frac{1}{\boldsymbol{x}_{\mathcal{S}}^\top \tilde{\boldsymbol{D}} \boldsymbol{x}_{\mathcal{S}}} + \frac{1}{\boldsymbol{x}_{\mathcal{S}'}^\top \tilde{\boldsymbol{D}} \boldsymbol{x}_{\mathcal{S}'}} \right). \tag{4.7}$$

Clearly, Eq. (4.7) is a differentiable function of $\tilde{\boldsymbol{A}}$. Computing its gradient $\nabla_{\tilde{\boldsymbol{A}}}\phi(\mathcal{S})$ can then be handled by automatic differentiation tools such as PyTorch [26].

Next, we compute the gradient of $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ w.r.t. $\tilde{\boldsymbol{A}}$. A standard way to compute $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ is by using SVD. However, this is both prohibitively expensive ($O(n^3)$), and does not provide us with the necessary gradient information. Instead, we use the power method [11] to compute $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$. Let $\boldsymbol{v}_{\mathcal{S}}$ be the eigenvector associated with the largest eigenvalue $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$. Using Rayleigh quotients [35], we can compute $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ as follows:

$$\boldsymbol{v}_{\mathcal{S}} = \arg\max_{\|\boldsymbol{x}\|_2=1} \boldsymbol{x}^\top \tilde{\boldsymbol{A}}_{\mathcal{S}} \boldsymbol{x} \tag{4.8a}$$

$$\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}}) = \boldsymbol{v}_{\mathcal{S}}^\top \tilde{\boldsymbol{A}}_{\mathcal{S}} \boldsymbol{v}_{\mathcal{S}}. \tag{4.8b}$$

Thus, when $\boldsymbol{v}_{\mathcal{S}}$ is known, the computation of $\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ reduces to matrix multiplications. In addition, $\tilde{\boldsymbol{A}}_{\mathcal{S}}$ is usually sparse, so we can leverage sparse matrix multiplication to speed up the computation.

The remaining challenge is that $\boldsymbol{v}_{\mathcal{S}}$ is an optimal solution of an optimization problem, and we need an explicit derivative of it. Fortunately, our problem has a special structure that we exploit to obtain an approximation of the derivative of $\boldsymbol{v}_{\mathcal{S}}$. From our experiments we find that $\tilde{G}_{\mathcal{S}}$ is nearly always connected. This means that the largest eigenvalue of $\tilde{\boldsymbol{A}}_{\mathcal{S}}$ is simple. In addition, due to the Perron–Frobenius theorem, the absolute value of the largest eigenvalue is strictly greater than the absolute values of others, i.e., $|\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})| > |\lambda_k(\tilde{\boldsymbol{A}}_{\mathcal{S}})|$ for all $k \neq 1$. Under these conditions, we can use the power method to estimate $\boldsymbol{v}_{\mathcal{S}}$ by repeating the formula: $\tilde{\boldsymbol{v}}_{\mathcal{S}}^{(t+1)} = \tilde{\boldsymbol{A}}_{\mathcal{S}}\tilde{\boldsymbol{v}}_{\mathcal{S}}^{(t)}/\|\tilde{\boldsymbol{A}}_{\mathcal{S}}\tilde{\boldsymbol{v}}_{\mathcal{S}}^{(t)}\|_2$. The ℓ_2-norm distance between $\tilde{\boldsymbol{v}}_{\mathcal{S}}^k$ and $\boldsymbol{v}_{\mathcal{S}}$ decreases in a rate $O(\rho^k)$ [11], where $\rho < 1$. In our experiments we found $k = 50$ is enough to give a high-quality estimation for a graph with 986 nodes. Intuitively, we are using a sequence of differentiable operations to approximate the argmax operation. Therefore the computation of $\nabla_{\tilde{\boldsymbol{A}}}\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}})$ can be handled by PyTorch.

We use the same machinery to compute $\nabla_{\tilde{\boldsymbol{A}}}\sigma(\mathcal{S})$. First, we write $\sigma(\mathcal{S})$ in matrix notation:

$$\sigma(\mathcal{S}) = \boldsymbol{v}^\top \boldsymbol{x}_{\mathcal{S}}, \tag{4.9}$$

where $\boldsymbol{v}$ is the unit eigenvector associated with $\lambda_1(\tilde{\boldsymbol{A}})$. Then we apply the power method to compute $\boldsymbol{v}$. Finally, $\sigma(\mathcal{S})$ is just a linear function of $\boldsymbol{v}$. All of these operations are differentiable, and the computation of $\nabla_{\tilde{\boldsymbol{A}}}\sigma(\mathcal{S})$ is handled by PyTorch.

We next address the challenge imposed by the constraints (3.5), which can result in a computationally challenging projection step which can also significantly harm solution quality. We address this challenge as follows. Given a real symmetric matrix $\boldsymbol{X}$, let $\|\boldsymbol{X}\|_2$ denote its spectral norm. To satisfy Eq. (3.5), we use the following result

from pseudospectrum theory (see [36], Theorem 2.2):

$$|\lambda_i(\tilde{\boldsymbol{A}})-\lambda_i(\boldsymbol{A})| \le \epsilon, i = 1,\dots,n \iff \|\tilde{\boldsymbol{A}}-\boldsymbol{A}\|_2 \le \epsilon \tag{4.10}$$

Since $\boldsymbol{\Delta} = \tilde{\boldsymbol{A}}-\boldsymbol{A}$ is real and symmetric, we have $\|\boldsymbol{\Delta}\|_2 = \max\{|\lambda_1(\boldsymbol{\Delta})|, |\lambda_n(\boldsymbol{\Delta})|\}$ and $-\lambda_n(\boldsymbol{\Delta}) = \lambda_1(-\boldsymbol{\Delta})$, which leads to:

$$\tilde{\boldsymbol{A}} \text{ satisfies Eq. (3.5)} \iff \max\{|\lambda_1(\boldsymbol{\Delta})|, |\lambda_1(-\boldsymbol{\Delta})|\} \le \epsilon. \tag{4.11}$$

This equivalence allows the attacker to check whether she is within budget simply by evaluating $\max\{|\lambda_1(\boldsymbol{\Delta})|, |\lambda_1(-\boldsymbol{\Delta})|\}$, i.e., computing the largest eigenvalue of a real symmetric matrix, which can be computed efficiently using, e.g., the power method [11].

Our algorithm leverages this connection as follows. Line 9 in Algorithm 1 is a one step look-ahead, which ensures that the perturbation $\boldsymbol{\Delta}_i$ is only added to $\tilde{\boldsymbol{A}}_i$ when there is enough budget. Recall from Section 3.2 that $\|\boldsymbol{\Delta}_i\|_2 = \max\{|\lambda_1(\boldsymbol{\Delta}_i)|, |\lambda_1(-\boldsymbol{\Delta}_i)|\}$. Thus this step requires us to compute $\lambda_1(\boldsymbol{\Delta}_i)$ and $\lambda_1(-\boldsymbol{\Delta}_i)$, using again the power method. Line 10 tracks the amount of budget used so far. We now show that the output of Algorithm 1 always returns a feasible solution. Suppose Algorithm 1 terminates after $k > 1$ iterations. This means $B_k + \|\eta_k\boldsymbol{\Delta}_k\|_2 > \epsilon$ and $B_k \le \epsilon$. In other words $B_k = \sum_{i=1}^{k-1} \|\eta_i\boldsymbol{\Delta}_i\|_2 \le \epsilon$. Note that the total amount of perturbation added to $\boldsymbol{A}$ is $\boldsymbol{\Delta} = \sum_{i=1}^{k-1} \eta_i\boldsymbol{\Delta}_i$. The triangle inequality implies $\|\boldsymbol{\Delta}\|_2 \le \epsilon$.

For each iteration of Algorithm 1, the most computationally expensive components are the power method and matrix multiplication. Let m be the number of nonzeros in $\tilde{\boldsymbol{A}}_i$; if the graph is unweighted then m is the number of edges at this iteration. By leveraging the sparseness exhibited in $\tilde{\boldsymbol{A}}_i$, the power method runs in $O(m)$ and the matrix multiplications cost $O(mn)$. Thus, the time complexity of each iteration is $O(mn)$, which significantly improves the $O(n^3)$ time complexity of SVD that would otherwise be needed.

Recall that our model for targeted diffusion is applicable to both weighted and unweighted graphs. For weighted graphs, the attacker modifies the weights on existing edges. For unweighted graphs, the attacker adds new edges or deletes existing edges from the graph. The main difference between the two settings is that the latter needs a rounding heuristic to convert a matrix with fractional entries to a binary adjacency matrix. We discuss this heuristic below.

After running Algorithm 1, we obtain a perturbed matrix $\tilde{\boldsymbol{A}}$ with fractional entries. For unweighted graphs, a rounding heuristic is needed to convert $\tilde{\boldsymbol{A}}$ to a valid adjacency matrix. Let $\mathcal{D} = \{(i,j)|\tilde{A}_{i,j} \neq A_{i,j}\}$ be the set of candidate edges that will be added or deleted from G. For each edge $(i,j) \in \mathcal{D}$ define the score $s_{(i,j)} = |\tilde{A}_{i,j} - A_{i,j}|$. Intuitively, $s_{(i,j)}$ indicates the impact that adding or deleting the edge has on the principal's utility. Next, we iteratively modify G, by adding or deleting edges in $\mathcal{D}$, starting with the one with the largest $s_{(i,j)}$. The modification process stops when the budget is exhausted, which results in the desired binary adjacency matrix. For weighted graphs, let $C = \max_{i,j} A_{ij}$ and normalize each entry by C, that is A_{ij}/C. We run Algorithm 1 on the normalized adjacency matrix, which results in $\tilde{\boldsymbol{A}}$. The desired adjacency matrix is obtained by multiplying each $\tilde{A}_{ij}$ by C, $C\tilde{A}_{ij}$. If integer weights are desired (e.g., the number of packages transmitted between two computers), a final rounding step is applied. Our experimental results show that the rounding heuristic is effective in practice.

5 Certified Robustness

This section addresses the following question: what are the limits on the attacker's ability to successfully accomplish her attack? More precisely, we now seek to identify necessary conditions on the attack budget ϵ so the attack succeeds; conversely, we can view a given graph to be *certified* to be robust to attacks that use a smaller budget than the one required.

Let $\texttt{TargetDiff}(\mathcal{S}, G, \epsilon)$ be an instance of the targeted diffusion problem with target subset $\mathcal{S}$, underlying graph G and budget ϵ. The attacker is successful on an instance $\texttt{TargetDiff}(\mathcal{S}, G, \epsilon)$ if she is able to modify G into $\tilde{G}$ within budget ϵ such that $I(\tilde{G}_{\mathcal{S}}) > I(G_{\mathcal{S}})$. We now derive a necessary condition for successful attacks, in the form of a lower bound on ϵ.

To derive the necessary condition on ϵ, we use our experimental observation that in successful attacks the degrees of nodes in the targeted subgraph $G_{\mathcal{S}}$ always increase. This is intuitive: a denser subgraph $G_{\mathcal{S}}$ will tend to increase the propensity of the diffusion (e.g., of malware) to spread within it, which is one of our explicit objectives. Let d_i (resp. $\tilde{d}_i$) be the degree of node i before (resp. after) graph modification. We assume if an attack is successful, the degrees of nodes in $G_{\mathcal{S}}$ are increased, i.e., $\tilde{d}_i \ge d_i$ for $i \in \mathcal{S}$.

Now, observe that computing the exact value of $I(G_{\mathcal{S}})$ is intractable, since the exact computation of π_i is prohibitive (see, e.g., [22], Section IV.B). Mieghem et al. [22] proposed a simple yet effective estimator

for π_i to be $1 - \delta/(\beta d_i)$. The estimator works in the regime $\delta/\beta \leq d_{min}$, where d_{min} is the minimum degree of G. Consequently, an estimator for $I(G_{\mathcal{S}})$ is $\hat{I}(G_{\mathcal{S}}) = \sum_{i \in \mathcal{S}} 1 - \delta/(\beta d_i)$. We focus on the setting where the estimation error is bounded by a small number, i.e., $|\hat{I}(G_{\mathcal{S}}) - I(G_{\mathcal{S}})| \leq \tau$. Note that τ can be estimated from historical diffusion data. The formal statement of the necessary condition is in Theorem 5.1. [5]

THEOREM 5.1. *Given an instance* `TargetDiff`$(\mathcal{S}, G, \epsilon)$, *$I(G_{\mathcal{S}})$ is estimated by $\hat{I}(G_{\mathcal{S}}) = \sum_{i \in \mathcal{S}} 1 - \delta/(\beta d_i)$. Suppose we have an upper bound $|\hat{I}(G_{\mathcal{S}}) - I(G_{\mathcal{S}})| \leq \tau$, the degrees of nodes in $\mathcal{S}$ are increased, i.e., $\tilde{d}_i \geq d_i$ for $i \in \mathcal{S}$, and $\delta/\beta \leq d_{min}$. In order to have $I(\tilde{G}_{\mathcal{S}}) - I(G_{\mathcal{S}}) > 2\tau$, the budget ϵ must satisfy:*

$$\epsilon \geq \sqrt{\frac{|\mathcal{S}|}{n}} \left(\frac{\sum_{i \in \mathcal{S}} d_i^2}{|S|} - \frac{\left(\sum_{i \in \mathcal{S}} d_i\right)^2}{|S|^2} \right)^{1/2}. \tag{5.12}$$

The quantity inside the square root is always nonnegative due to Jensen's inequality. The lower bound involves only structural properties of the graph (node degrees and the size of $\mathcal{S}$) and thus can be easily computed given an arbitrary graph. As mentioned above, we can view this lower bound as a robustness certificate, or guarantee for the given graph. It guarantees, in particular, that when the budget is below the lower bound, the total probability of "infection" (e.g., malware infection) in $G_{\mathcal{S}}$ cannot be increased by more than 2τ. In the special case of perfect estimation ($\tau = 0$), it implies impossibility of increasing the susceptibility of $G_{\mathcal{S}}$ to targeted diffusion.

The proof of Theorem 5.1 does not depend on the specific objective function proposed in this paper. Consequently, the certificate is not specific to our particular objective function. Further, the lower bound is independent of the values of δ and β, as long as $\delta/\beta \leq d_{min}$.

We briefly discuss the settings where the robustness guarantee is most applicable. First, the estimation for the infected ratio on $G_{\mathcal{S}}$ is accurate, i.e., $|\hat{I}(G_{\mathcal{S}}) - I(G_{\mathcal{S}})| \leq \tau$ and τ is small. According to Mieghem et al. [22], this usually happens on graphs with small degree variation. Another setting is where the degrees of nodes in $G_{\mathcal{S}}$ increase as a result of the attack, which is both natural and empirically founded, as we mentioned earlier. We provide experimental results on synthetic networks to verify the robustness guarantee in Section 6.

[5] Due to brevity the proof is in Appendix C at `https://arxiv.org/abs/2008.05589`.

6 Experiments and Discussion

This section presents experimental results on three real-world datasets: an email network, an airport network, and a brain network. [6]

For each network we run POTION with hyper-parameters $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$, which encodes that the attacker's objectives are equally important. To study how the attacker's effectiveness changes with respect to her budget, we set $\epsilon = \gamma\lambda_1(\boldsymbol{A})$ and vary γ from 10% to 50%. A single initially infected node is selected uniformly at random.

Recall we use G and $\tilde{G}$ to denote the original and the modified graphs, respectively. We simulate the spreading dynamics 2000 times on both G and $\tilde{G}$. For unweighted graphs the recovery rate δ and transmission rate β are set to 0.24, 0.06, resp.; for weighted graphs we set $\delta = 0.24$ and $\beta = 0.2$. The spreading dynamics converges exponentially fast to the steady state: empirically, we found 30 time steps to be enough to reach the steady state in most cases. When the simulation finishes, we extract the number of nodes that are "infected". We use I_{original} and I_{modified} to represent the fractions of infected nodes on G and $\tilde{G}$, resp.

We use two other algorithms as baselines for comparison, which we call `deg` and `gel`. The two algorithms work by alternating between modifying $G_{\mathcal{S}}$ and modifying $G_{\mathcal{S}'}$ until the budget is spent. When modifying $G_{\mathcal{S}}$, `deg` chooses the edge (i, j) with the maximum value of $d_i + d_j$. Algorithm `gel` is based on [32], and chooses the edge (i, j) with maximum *eigenscore*, defined as $u[i]v[j]$, where u, v are the left and right principal eigenvectors of $\boldsymbol{A}$, respectively. This edge is chosen from among those edges that are absent (if the graph is unweighted) or present (if the graph is weighted). When modifying $G_{\mathcal{S}'}$, these baselines choose an existing edge of to remove (if unweighted) or decrease its weight (if weighted) with the highest value of $d_i + d_j$ or $u[i]v[j]$, respectively.

Unweighted Graphs: We consider (the largest connected component of) an email network [18] that has 986 nodes. An edge (i, j) indicates that there were email exchanges between nodes i and j. This data set contains ground-truth labels to indicate which community a node belongs to. We pick a community with 15 nodes as $\mathcal{S}$; the results for communities with other sizes are similar.

The overall effectiveness of our approach is shown in Figure 2, top. The difference $I_{\text{modified}} - I_{\text{original}}$ of the

[6] Due to brevity, additional results on real and synthetic networks are at `https://arxiv.org/abs/2008.05589`.

impact on the modified and original graphs is shown for $G_{\mathcal{S}}$ (red line) and $G_{\mathcal{S}'}$ (purple line), respectively. As γ gets larger, the impact on $G_{\mathcal{S}}$ increases, while the impact on $G_{\mathcal{S}'}$ is under control, which demonstrates that the proposed approach is highly effective at both increasing the impact of diffusion on the targeted subgraph, and at the same time preventing the impact on the remaining graph.

Weighted Graphs: We consider an airport network and a brain network. The airport network [24] was collected from the website of Bureau of Transportation Statistics of the U.S., where the nodes represent all of the 1572 airports in the U.S. and the weights on edges encode the number of passengers traveled between two airports in 2010. We scaled the weights on the airport network to $[0, 1]$. The targeted set $\mathcal{S}$ was chosen by first sampling a node i uniformly at random, and then setting $\mathcal{S}$ to be i and all its neighbors. We report experimental results for an $\mathcal{S}$ with 60 nodes. The brain network [8] consists of 638 nodes where each node corresponds to a region in human brain. An edge between nodes i and j indicates that the two regions have co-activated on some tasks. The weight on the edge quantifies the strength of the co-activation estimated by the Jaccard index. The weights on edges lie in $[0, 1]$. The 638 regions are categorized into four areas: default mode, visual, fronto-parietal, and central. Each area is responsible for some functionality of human. We select 100 nodes from the central area as the targeted set $\mathcal{S}$. The results for the airport (resp. brain) network are at the center (resp. right) column of Figure 2. The overall trend is similar to that of the email network.

Comparison against Baselines: The comparisons against the baselines are shown in Figure 2, middle and bottom rows. The moddle row shows the infectious ratios within the targeted subgraphs. It is clear that our algorithm is more effective at increasing the infectious ratios than the baselines. The bottom row shows the infectious ratios within the non-targeted subgraphs. The magnitudes of the differences are negligible, although in some cases our algorithm is significantly better than the baselines (e.g., on airport network when $\gamma = 0.4$).

Verify the Certified Robustness: We run experiments on synthetic networks to verify the certified robustness; the synthetic networks include Barabási-Albert (BA) [4], Watts-Strogatz [40], and Block Two-level Erdős-Rényi (BTER) networks [30]. We use the same experimental setup as described above. Fig. 1 shows the difference of infectious ratios on the modified and original graphs (within targeted subgraphs), as a function of the attacker's budget ϵ. The vertical dashed lines are the

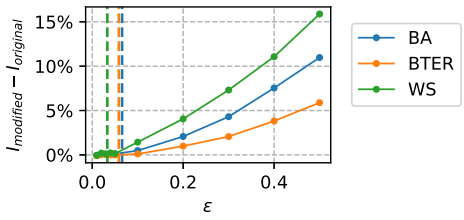

Figure 1: Certified robustness results. Dashed lines mark the lower bounds from Eq. (5.12). Solid lines represent infectious ratios within targeted subgraphs.

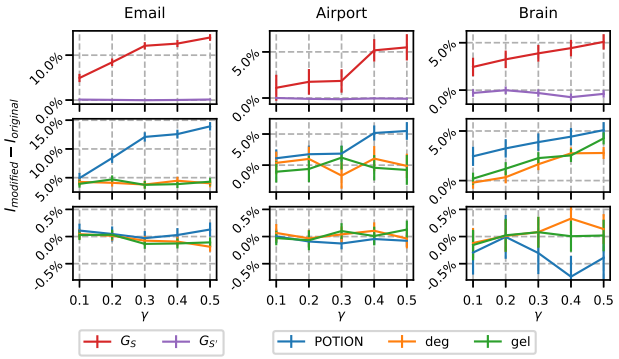

Figure 2: POTION effectively achieves targeted diffusion (**top**) in G_S (red line) without affecting $G_{S'}$ (purple); higher is better. Comparison against deg and gel baselines in G_S (**middle**; higher is better) and $G_{S'}$ (**bottom**; lower is better).

lower bounds on the budget computed using Eq. (5.12). Note that when the budget is less than the lower bound, the differences are close to zero, which means that the network is robust against targeted diffusion.

Running Time: The running time of Algorithm 1 on the three real-world networks is showed in Figure 3. Each point in the figure is the average running time over 10 trials. Intuitively, as the budget γ increases the attacker needs to search a larger space, therefore the running time increases. The numbers of nodes and edges of the three networks are in Table 1.

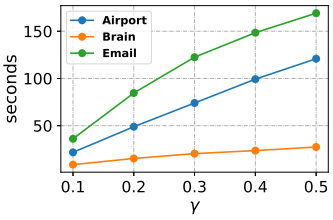

Figure 3: Running time on the real-world networks.

	Email	Airport	Brain
#nodes	986	1572	638
#edges	16064	17214	18625

Table 1: Statistics of the real-world networks.

7 Conclusion

Diffusion control on network has attracted much attention, however, most studies focus on diffusion over the entire network. We address the problem of *targeted* diffusion attack on networks. We present a combination of modeling and algorithmic advances to systematically address this problem. On the modeling side, we present a novel model called POTION that optimizes graph structure to affect such targeted diffusion attacks, which preserves structural properties of the graph. On the algorithmic side, we design an efficient algorithm named POTION-ALG by leveraging Rayleigh quotients and pseudospectrum theory, which is scalable to real-world graphs. We also derive a condition to certify whether a network is robust against a broad class of targeted diffusion. Our experiments on both synthetic and real-world networks show that the model is highly effective in implementing the targeted diffusion attack.

Acknowledgement

SY and YV were partially supported by the National Science Foundation (grants IIS-1903207 and IIS-1910392) and Army Research Office (grants W911NF1810208 and W911NF1910241). LT and TER were supported in part by the National Science Foundation (IIS-1741197) and by the Combat Capabilities Development Command Army Research Laboratory (under Cooperative Agreement Number W911NF-13-2-0045). The authors would like to thank the anonymous reviewers and Chloe Wohlgemuth for their helpful comments.

References

[1] Victor Amelkin and Ambuj K. Singh. Fighting opinion control in social networks via link recommendation. In *KDD*, pages 677–685. ACM, 2019.

[2] Konstantin Avrachenkov and Nelly Litvak. The effect of new links on google pagerank. *Stochastic Models*, 22 (2):319–331, 2006.

[3] Norman TJ Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 1975.

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

[5] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jure Leskovec, and Christos Faloutsos. Epidemic thresholds in real networks. *ACM Trans. Inf. Syst. Secur.*, 10(4): 1:1–1:26, 2008.

[6] Chen Chen, Hanghang Tong, B. Aditya Prakash, Charalampos E. Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. Node immunization on large graphs: Theory and algorithms. *TKDE*, 28(1):113–126, 2016.

[7] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208. ACM, 2009.

[8] Nicolas A Crossley, Andrea Mechelli, Petra E Vértes, Toby T Winton-Brown, Ameera X Patel, Cedric E Ginestet, Philip McGuire, and Edward T Bullmore. Cognitive relevance of the community structure of the human brain functional coactivation network. *PNAS*, 110(28):11583–11588, 2013.

[9] Ernesto Estrada. 'Hubs-repelling' Laplacian and related diffusion on graphs/networks. *Linear Algebra Appl.*, 2020.

[10] Pablo Fleurquin, José J Ramasco, and Victor M Eguiluz. Systemic delay propagation in the US airport network. *Scientific Reports*, 3:1159, 2013.

[11] Gene Golub and Charles Van Loan. *Matrix computations*. Johns Hopkins Studies in Mathematical Sciences, 1996.

[12] Nika Haghtalab, Aron Laszka, Ariel D. Procaccia, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Monitoring stealthy diffusions. *Knowledge and Information Systems*, 2017.

[13] Christopher Ho, Mykel J. Kochenderfer, Vineet Mehta, and Rajmonda S. Caceres. Control of epidemics on graphs. In *CDC*, pages 4202–4207. IEEE, 2015.

[14] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146. ACM, 2003.

[15] David Kempe, Sixie Yu, and Yevgeniy Vorobeychik. Inducing equilibria in networked public goods games through network structure modification. In *AAMAS*, pages 611–619, 2020.

[16] Long T. Le, Tina Eliassi-Rad, and Hanghang Tong. MET: A fast algorithm for minimizing propagation in large graphs with small eigen-gaps. In *SDM*, pages 694–702. SIAM, 2015.

[17] Stamatios Lefkimmiatis, John Paul Ward, and Michael Unser. Hessian schatten-norm regularization for linear inverse problems. *IEEE Trans. Image Process.*, 22(5): 1873–1888, 2013.

[18] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007.

[19] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie S. Glance, and Matthew Hurst. Patterns of cascading behavior in large blog graphs. In *SDM*, pages 551–556. SIAM, 2007.

[20] Jure Leskovec, Lars Backstrom, and Jon M. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, pages 497–506. ACM, 2009.

[21] Marina Meila and Jianbo Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879. MIT Press, 2000.

[22] Piet Van Mieghem, Jasmina Omic, and Robert E. Kooij. Virus spread in networks. *IEEE/ACM Trans. Netw.*, 17 (1):1–14, 2009.

[23] Adilson E Motter and Ying-Cheng Lai. Cascade-based attacks on complex networks. *Phys. Rev. E*, 66(6): 065102, 2002.

[24] Tore Opsahl. US airport network traffic data in 2010. `https://toreopsahl.com/2011/08/12/why-anchorage-is-not-that-important-binary-ties-and-sample-selection/`, 2010.

[25] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

[27] B. Aditya Prakash, Deepayan Chakrabarti, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowl. Inf. Syst.*, 33(3):549–575, 2012.

[28] Victor M. Preciado, Michael Zargham, Chinwendu Enyioha, Ali Jadbabaie, and George J. Pappas. Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks. In *CDC*, pages 7486–7491. IEEE, 2013.

[29] Sudip Saha, Abhijin Adiga, B. Aditya Prakash, and Anil Kumar S. Vullikanti. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *SDM*, pages 568–576. SIAM, 2015.

[30] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. Community structure and scale-free collections of erdős-rényi graphs. *Phys. Rev. E*, 85(5):056109, 2012.

[31] Hanghang Tong, B. Aditya Prakash, Charalampos E. Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. On the vulnerability of large graphs. In *ICDM*, pages 1091–1096. IEEE Computer Society, 2010.

[32] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, pages 245–254. ACM, 2012.

[33] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, pages 245–254. ACM, 2012.

[34] Leo Torres, Kevin S Chan, Hanghang Tong, and Tina Eliassi-Rad. Node immunization with non-backtracking eigenvalues. *arXiv preprint arXiv:2002.12309*, 2020.

[35] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

[36] Lloyd N Trefethen and Mark Embree. *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*. Princeton University Press, 2005.

[37] Piet Van Mieghem, Dragan Stevanović, Fernando Kuipers, Cong Li, Ruud Van De Bovenkamp, Daijie Liu, and Huijuan Wang. Decreasing the spectral radius of a graph by link removals. *Phys. Rev. E*, 84(1):016101, 2011.

[38] Tan Van Vu and Yoshihiko Hasegawa. Diffusion-dynamics laws in stochastic reaction networks. *Phys. Rev. E*, 99:012416, 2019.

[39] Zhihui Wang and Thomas S. Deisboeck. Dynamic targeting in cancer treatment. *Frontiers in Physiology*, 10:1–9, 2019.

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

[41] Yang Yang, Takashi Nishikawa, and Adilson E Motter. Small vulnerable sets determine large network cascades in power grids. *Science*, 358(6365):eaan3184, 2017.

[42] Sixie Yu and Yevgeniy Vorobeychik. Removing malicious nodes from networks. In *AAMAS*, pages 314–322, 2019.

[43] Haifeng Zhang, Yevgeniy Vorobeychik, Joshua Letchford, and Kiran Lakkaraju. Data-driven agent-based modeling, with application to rooftop solar adoption. *JAAMAS*, 30(6):1023–1049, 2016.

[44] Kai Zhou, Tomasz P. Michalak, Marcin Waniek, Talal Rahwan, and Yevgeniy Vorobeychik. Attacking similarity-based link prediction in social networks. In *AAMAS*, pages 305–313, 2019.