Fighting Opinion Control in Social Networks via Link Recommendation

Victor Amelkin
University of Pennsylvania
vctr@seas.upenn.edu

Ambuj K. Singh
University of California, Santa Barbara
ambuj@cs.ucsb.edu

ABSTRACT

The process of opinion formation is inherently a network process, with user opinions in a social network being driven to a certain average opinion. One simple and intuitive incarnation of this opinion attractor is the average $\pi^\intercal x$ of user opinions x_i weighted by the users' eigenvector centralities π_i. This value is a lucrative target for control, as altering it essentially changes the mass opinion in the network. Since any potentially malicious influence upon the opinion distribution in a society is undesirable, it is important to design methods to prevent external attacks upon it.

In this work, we assume that the adversary aims to maliciously change the network's average opinion by altering the opinions of some unknown users. We, then, state an NP-hard problem of disabling such opinion control attempts via strategically altering the network's users' eigencentralities by recommending a limited number of links to the users. Relying on Markov chain theory, we provide perturbation analysis that shows how eigencentrality and, hence, our problem's objective change in response to a link's addition to the network. The latter leads to the design of a pseudo-linear-time heuristic, relying on efficient estimation of mean first passage times in Markov chains. We have confirmed our theoretical and algorithmic findings, and studied effectiveness and efficiency of our heuristic in experiments with synthetic and real networks.

CCS CONCEPTS

• **Theory of computation → Social networks**; **Network optimization**; **Random walks and Markov chains**.

ACM Reference Format:
Victor Amelkin and Ambuj K. Singh. 2019. Fighting Opinion Control in Social Networks via Link Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330960

1 INTRODUCTION

It is well-known that in the absence of the objective means for opinion evaluation, people tend to evaluate their opinions by comparison with the opinions of others [15]. Thus, social networks impact the opinion formation process in the society. It is clearly desirable that this process is natural and fair, with good ideas emerging and spreading, and bad ideas declining and disappearing. However, viral marketing experts may be interested in controlling the opinion formation process with the goal of driving the opinion distribution to a certain business-oriented objective. One well-known way of affecting opinion formation is influence maximization, whose central idea is to strategically change the opinions of some network users with the goal of maximizing the subsequent spread of "right" opinions from these users throughout the network, or, more generally, shifting the opinion distribution towards a desired state. Naturally, the society would benefit from a mechanism preventing such potentially malicious control of the opinion formation process. Our work is dedicated to the design of one such mechanism—a link recommendation algorithm that allows an online social network platform to disable the effect of the attempts to control the opinion distribution in the network.

The notion central to our work is that of the *average opinion* of a social network, characterizing the mass opinion. For example, it can measure to what extent, on average, people prefer one smartphone brand or one political party over the other. Based on the well-established socio-psychological theories, such as Festinger's social comparison [15] and cognitive dissonance [16] theories, the opinions of users in the network are attracted towards the average opinion. However, the basic arithmetic average would not suffice, as the opinions of more "central" users—such as celebrities—are expected to contribute more to the entire network's opinion. Instead, we use the following simple and intuitive definition of the network's average opinion

$$\boxed{\textit{average opinion of the network} = \langle \pi, x \rangle = \pi^\intercal x}$$

which is the sum of users' quantified opinions x_i weighted by the corresponding users' eigenvector centralities π_i.

Clearly, the average opinion is a lucrative target for control, as altering it essentially changes the mass opinion in the network. We assume that there is an external adversary whose goal is, without loss of generality, to maximize the average opinion $\langle \pi, x \rangle$. To that end, the adversary influences some users in the network, changing the opinion distribution $x \to \widetilde{x}$ and, thus, the average opinion $\langle \pi, x \rangle \to \langle \pi, \widetilde{x} \rangle$ (Fig. 1b). Then, our goal—assuming we act on behalf of the social network platform—is to respond to this attack, and restore the average opinion to its original state $\langle \pi, x \rangle$. We, however, cannot directly influence user opinions; the only legitimate opinion control tool available to us is *link recommendation*. We strategically recommend a limited number of links to the networks' users, thereby, affecting these users' eigencentralities $\pi \to \widetilde{\pi}$ and restoring the original average opinion, $\langle \pi, \widetilde{x} \rangle \to \langle \widetilde{\pi}, \widetilde{x} \rangle \approx \langle \pi, x \rangle$ (Fig. 1c).

The work was supported by U.S. Army Research Laboratory and U. S. Army Research Office grant W911NF-15-1-0577 and the National Science Foundation grant IIS-1817046.

KDD'19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330960

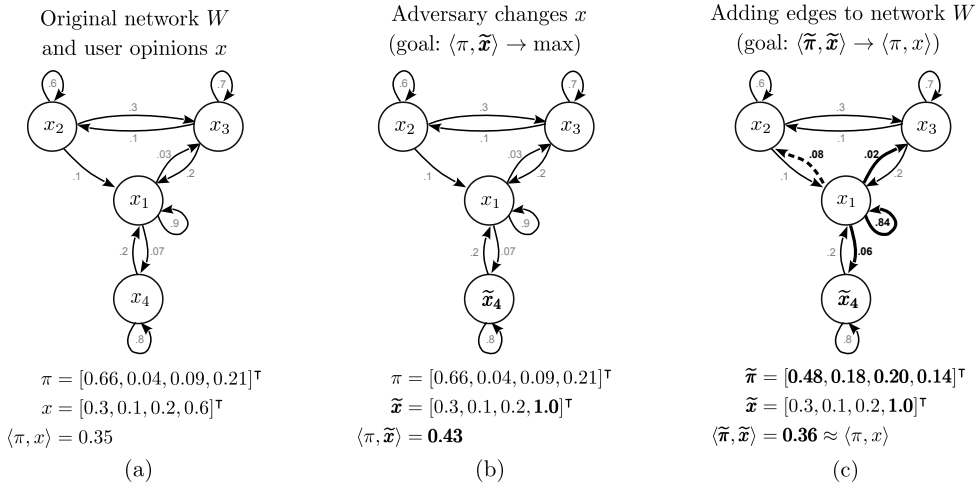

Figure 1: The adversary influences user opinions, $x \to \widetilde{x}$, increasing the average opinion $\langle \pi, x \rangle \to \langle \pi, \widetilde{x} \rangle$, where π is a vector of users' eigencentralities. The network responds by adding links, reducing the average opinion $\langle \widetilde{\pi}, \widetilde{x} \rangle$, driving it back to its original value $\langle \pi, x \rangle$.

The *central goal* of our work is to design a scalable algorithm that, under the above described attack upon the opinion distribution, would identify the links whose recommendation to the social network's users would efficiently drive the maliciously altered average opinion back to its state prior to the attack, nullifying the attack's impact. Our *specific contributions* are:

▷ We define DIVER—a *new NP-hard problem* of disabling external influence in a social network via edge recommendation.

▷ We provide *novel perturbation analysis*, establishing how the network's eigencentrality vector changes when an edge is added to the network. This analysis leads to the definition of an edge score $f_\pi(i, j)$ quantifying the potential impact of the addition of directed edge (i, j) to the network upon DIVER's objective.

▷ We show how to *estimate edge scores f_π in pseudo-constant time* in networks with skewed eigenvector centrality distribution, such as scale-free networks.

▷ We provide a *pseudo-linear-time heuristic for* DIVER relying on edge scores f_π, and experimentally study its effectiveness and efficiency on synthetic and real-world networks.

2 PRELIMINARIES

We are given a sparse directed strongly connected aperiodic social network $G(V, E)$, $|V| = n$, $|E| = O(n)$, having row-stochastic adjacency matrix $W \in [0, 1]^{n \times n}$, $W\mathbb{1} = \mathbb{1}$, $\mathbb{1} = [1, \ldots, 1]^\intercal$. The edge/link weight $w_{ij} \in [0, 1]$ reflects the relative extent to which user i takes into account the opinion of user j while forming his or her opinion, or, alternatively, the relative extent to which user i trusts user j. Aperiodicity can be replaced by the requirement of the network's having at least one self-loop with a non-zero weight, which translates into a natural requirement of having at least one user who puts at least some trust in his or her own opinion.

The average opinion of the network is defined as $\langle \pi, x \rangle = \pi^\intercal x$, where $x \in [0, 1]^n$ are opinions of the users, and $\pi \in \mathbb{R}^n_+$, $\|\pi\|_1 = 1$ is the ℓ_1-normalized dominant left eigenvector of W. From its definition, π is also the vector of eigencentralities of the network's nodes, and can also be viewed as the vector of no-teleportation PageRank scores, or the stationary distribution of the ergodic Markov chain with state transition matrix W. Due to the latter, we may refer to W as a Markov chain, and use the terms node (of a network) and state (of a Markov chain) interchangeably. We will also require the following characteristic of W if viewed as a Markov chain.

Definition 1 (Mean first passage time). *The mean first passage time (MFPT) m_{ij} from state i to state j of a Markov chain is the expected number of steps it takes the chain started at state i to reach state j for the first time. m_{ii} is the mean first return time (MFRT).*

In general networks, m_{ij} reflects how fast, on average, a random walker started at node i passes node j for the first time, when the walk is governed by transition likelihoods W. In social networks, as defined above, m_{ij} measures *relative centrality or importance* of node j for node i's opinion formation. The key difference of m_{ij} from w_{ij} is that w_{ij} reflects direct immediate influence exerted through a single link, while m_{ij} reflects aggregate influence accumulated along all existing pathways in the network and qualified with likelihoods of information spreading along each such pathway.

We will also need the following Theorem 1 and 2 immediately following from Theorems 4.4.4 and 4.4.5 of Kemeny and Snell [21].

Theorem 1 (MFRT and eigenvector centrality). *For any state i of Markov chain W with an aperiodic strongly connected network, $m_{ii} = 1/\pi_i$.*

Thus, MFPT generalizes eigenvector centrality, providing both absolute and relative node importance information.

Theorem 2 (MFPT one-hop conditioning). *For any states i and j of Markov chain W with an aperiodic strongly connected network, $m_{ij} = 1 + \sum_{k \neq j} w_{ik} m_{kj}$.*

3 PROBLEM STATEMENT AND HARDNESS

Given a directed social network, at each point in time, we—acting on behalf of the social network platform—can observe its users' opinions. We assume that at some point the external adversary makes an influence maximization attempt by targeting several users and changing their opinions, with the goal of, without loss of generality,

$\mathbb{1}$	vector of all ones
$\text{diag}(v)$	diagonal matrix with vector v as the main diagonal
e_i	i'th column of the identity matrix
x ($\widetilde{x}$)	user opinions (user opinions altered by the adversary)
W	network's row-stochastic adjacency matrix
$\widetilde{W}$	altered network's row-stochastic adjacency matrix
θ_{ij}	weight of directed edge (i, j) (candidate for addition)
π ($\widetilde{\pi}$)	ℓ_1-normalized left dominant eigenvector of W (of $\widetilde{W}$)
m_{ij}	mean first passage time from state i to state j in chain W

Table 1: Notation Summary

maximizing the average opinion $\langle \pi, x \rangle$. In this work, we assume no knowledge of which users have been attacked, relying only on our ability to detect such attacks using existing methods. The latter methods range from the anomalous event detection in a scalar time series of average opinion values [7], to tracking whether the current cumulative changes in user opinions follow a pattern prescribed by the solution to an influence maximization problem, to network sensing for outbreak detection [24], to opinion dynamics model-based anomaly detection techniques [1]. In what follows, we assume that attack detection—orthogonal to the issue of attack effect elimination that we focus on in the paper—has been successfully performed using one of the mentioned techniques.

Having detected an external influence attempt, we are given the opinion distribution $x \in [0, 1]^n$ preceding the attack as well as the externally altered opinion distribution $\widetilde{x} \in [0, 1]^n$. As a result of the attack, the original average opinion $\langle \pi, x \rangle$ has changed to $\langle \pi, \widetilde{x} \rangle$. Our goal is to strategically add a limited number k of edges to the network and, thereby, change π in such a way, that the resulting average opinion $\langle \widetilde{\pi}, \widetilde{x} \rangle$ is as close as possible to its state $\langle \pi, x \rangle$ before the attack, as it was illustrated in Fig. 1. Formally, the problem of disabling external influence via edge recommendation is defined as

$$\boxed{\text{DIVER}(W, k, x, \widetilde{x}) = \arg\min_{\widetilde{W}} |\langle \widetilde{\pi}(\widetilde{W}), \widetilde{x} \rangle - \langle \pi, x \rangle|,} \quad (1)$$

where the perturbed row-stochastic adjacency matrix $\widetilde{W}$ differs from W by k new edges whose weights θ_{ij} we cannot control—since they correspond to interpersonal trust that users decide upon themselves—yet, can estimate using existing techniques [18], and, hence, assume to be known. Notice that, as edge weights reflect *relative* trust, after new edge addition to a user's out-neighborhood, the existing edges are proportionally downweighted, and $\widetilde{W}$ is kept row-stochastic. For $k = 1$, after addition of directed edge (r, c) with weight θ_{rc}, $\widetilde{W}$ looks as follows (see Fig. 2):

$$\widetilde{W} = W - \theta_{rc}\, \text{diag}(e_r) W + \theta_{rc} e_r e_c^{\mathsf{T}}, \quad (2)$$

where $e_i \in \{0, 1\}^n$ is i'th element of the standard basis.

In what follows, we will focus on deterministic edge addition, and provide a straightforward *probabilistic extension*, fitting the friend recommendation mechanism of popular online social networks, in the Supplement [2].

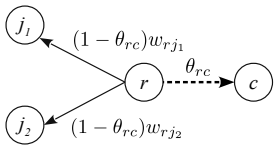

Figure 2: Addition of edge (r, c) to the network.

DIVER's complexity comes along two dimensions—searching for the best edge subset delivering the minimum of the objective, and assessing the impact of a given subset of edges upon the objective. While the latter can be done in polynomial time, the edge subset search cannot and is the root cause of NP-hardness. In the following Theorem 3, proven in the Supplement [2], we establish that, even for the case of undirected networks, DIVER$(W, k, x, \widetilde{x})$ is NP-hard.

Theorem 3. DIVER *is NP-hard for undirected networks.*

The problem's hardness is exacerbated by the fact that standard approximation techniques [28] are not applicable, as DIVER's objective function is not submodular.

4 RELATED WORK

While DIVER is a new problem, and there are no existing methods for solving it, there is a range of problems—in extremal network design as well as in the perturbation analysis of centrality measures and stationary distributions of Markov chains—related to ours either in the nature of the optimized objective or the methods and analyses. In this section, we survey several groups of these works, focusing on analytic problems (combinatorial extremal network design is reviewed in the Supplement [2]).

4.1 Analytic Network Design

Here, we review network design problems, where a network's topology is altered to optimize some analytic property of that network.

Algebraic Connectivity: Ghosh and Boyd [17] studied the problem of maximizing the algebraic connectivity—the second smallest eigenvalue λ_2 of the Laplacian—of an undirected unweighted network via edge addition. The authors use convex relaxation to formulate the problem as a semidefinite program (SDP), which is feasible to solve for small networks. They also provide a greedy perturbation heuristic that picks edges (i, j) based on the largest value of $(v_i - v_j)^2$—the squared difference of the Fiedler vector's components corresponding to each edge's ends. The authors show that, in case of simple λ_2, value $(v_i - v_j)^2$ gives the first-order approximation of the increase in λ_2 if edge (i, j) is added to the network. The authors also derive bounds on algebraic connectivity under single-edge perturbation. More recently, this approach has been employed by Yu et al. [34] for the design of an edge selection heuristic that the authors have augmented with an extra objective—neighborhood overlap-based user similarity (which likely correlates with edge acceptance likelihood).

Spectral Radius: Van Mieghem et al. [32] study the problem of minimizing the spectral radius of an undirected network via edge or node removal. They prove NP-hardness of the problem, and show that the edge selection heuristic that picks edges (i, j) with the largest scores $v_i v_j$—where v is the dominant eigenvector—performs well in practice. More recently, Saha et al. [30] addressed the same spectral radius minimization problem and designed a walk-based algorithm, relying on the link between the sum of powers of eigenvalues of a network and the number of closed walks in it, and provided approximation guarantees for them. Zhang et al. [35] studied spectral radius minimization for directed networks under SIR model, and provided an SDP/LP-based solution, having high polynomial time complexity.

Eigenvalues and Their Functions: Tong et al. [31] target optimization of the diffusion rate—expressed as the largest eigenvalue of the adjacency matrix—through a directed strongly connected unweighted (see [9] for the weighted case) network via edge addition or removal. Similarly to [32], the authors use first-order perturbation theory to assess the effect of deleting k edges $\lambda_{max} - \widetilde{\lambda}_{max} = \sum u_i v_j / \langle u, v \rangle + O(k)$, where u and v are the left and right dominant eigenvectors of the adjacency matrix, respectively. This analysis inspires an edge selection heuristic, with the quality of edge (i, j) being defined as $u_i v_j$, similarly to $v_i v_j$ edge score of [32]. Le et al. [23] extend this result to the networks with small eigengaps. Chan et al. [6] target optimization of natural connectivity—a network robustness measure defined, roughly, as an average of exponentiated eigenvalues of the adjacency matrix—of an undirected network via altering its topology. For edge addition, they focus on the edges between high-centrality nodes.

4.2 Centrality Perturbation and Manipulation

These works study either how eigenvector centrality, or PageRank, or the stationary distribution of a Markov chain changes when a network's structure is perturbed, or how to strategically manipulate centrality by altering the network.

Strategic Centrality Manipulation: Avrachenkov and Litvak [3] analyze to what extent a node can improve its PageRank by creating new out-edges. The authors derive equalities that result in a conclusion that the PageRank of a web-page cannot be considerably improved by restructuring its out-neighborhood. The authors also derive an optimal linking strategy, stating that it is optimal for a web-page to have only one outgoing edge pointing to a web-page with the shortest mean first passage time back to the original page. Similar conclusions can be drawn for eigenvector centrality in an arbitrarily weighted network using Theorem 4 of our work. De Kerchove et al. [14] generalize the results of Avrachenkov and Litvak [3], studying maximization of the sum of PageRanks of a subset of nodes via adding outgoing edges to them. Csáji et al. [13] study the problem of optimizing the PageRank of a given node via directed edge addition. The authors formulate the optimization problem as a Markov decision process and propose a (non-scalable) polynomial-time algorithm for it.

Centrality Perturbation Analysis: Cho and Meyer [11] provide coarse bounds of type $|\pi_i - \widetilde{\pi}_i|/\pi_i \leq \|E\|_\infty \max_{i \neq j} m_{ij}/2$ for the stationary distribution of a generally perturbed Markov chain, where E is an additive perturbation of the state transition matrix. Chien et al. [10] provide an efficient algorithm for incremental computation of PageRank over an evolving edge-perturbed graph, with the analysis' drawing upon the theory of Markov chains. The key idea of their algorithm is to contract the network and localize its part where the nodes are likely to have changed their PageRank scores under the perturbation. Langville and Meyer [22] provide exact equalities for the change in the stationary distribution of a perturbed Markov chain using group inverses. They address the problem of updating the stationary distribution under multi-row perturbation via exact and approximate aggregation, similarly to what Chien et al. [10] did for PageRank. Hunter [19] addresses the same problem of establishing equalities for the change in the stationary distribution, yet, provides an answer that does not involve group inverses and, instead, uses mean first passage times in a Markov chain; our perturbation analysis in Sec. 5.2 builds upon this result. Como and Fagnani [12] provide an upper bound on the perturbation of the stationary distribution of a Markov chain in terms of the mixing time of the chain as well as the entrance time to and the escape likelihood from the states with perturbed out-neighborhoods. Bahmani et al. [5] address the problem of updating PageRank algorithmically. The proposed node probing-based algorithms provide a close estimate of the network's PageRank vector by crawling a small portion of the network. More recently, Li et al. [26] and Chen and Tong [8] addressed a general problem of updating eigenpairs of an evolving network. Chen and Tong provide a linear-time algorithm for tracking top eigenpairs.

5 STRATEGIC EDGE ADDITION

Since, due to Theorem 3, DIVER optimization problem

$$\text{DIVER}(W, k, x, \widetilde{x}) = \arg\min_{\widetilde{W}} |\langle \widetilde{\pi}(\widetilde{W}), \widetilde{x} \rangle - \langle \pi, x \rangle|, \qquad (1)$$

is NP-hard, we need to design a heuristic for it. Our general approach, formalized in Sec. 5.5, is as follows. We will assess candidate edges with respect to how much their addition to the network decreases term $\langle \widetilde{\pi}, \widetilde{x} \rangle$ of (1), and then iteratively add the most promising edges to the network until satisfied with the objective's value.

Hence, our foremost concerns now are: (a) the selection of a small number of candidate edges to assess; and (b) the subsequent assessment of the potential impact of these edges' addition to the network upon the network's eigenvector centrality and, thus, the objective. These concerns are addressed in what follows.

5.1 Candidate Edge Selection

The above described general approach involves assessing candidate edges individually. However, the number of absent edges in a sparse network is $O(n^2)$, and inspecting all of them is unfeasible for large networks. Hence, we will focus on a small number of candidate edges, outgoing from $n_{src} \ll n$ network nodes. This implies that a small number of nodes are being the sources of most—or, at least, a large number of—"good" candidate edges. Intuitively, the nodes having the largest (eigenvector) centrality should be those edge sources; the changes in their out-neighborhoods should have the largest impact upon the centrality distribution in the network, as Fig. 3 suggests. This intuition will find formal support in Corollary 1 in the following Sec. 5.2.

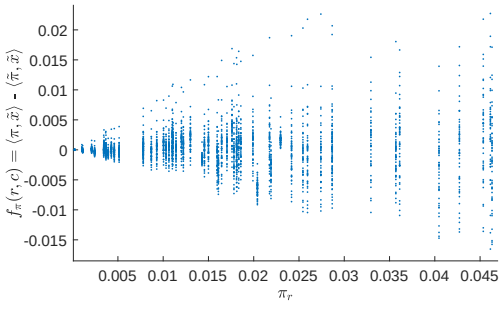

Figure 3: Dependence of the average opinion's reduction $f_\pi(r, c) = \langle \pi, \widetilde{x} \rangle - \langle \widetilde{\pi}, \widetilde{x} \rangle$ after addition of edge (r, c), $\theta_{rc} = const$ to a scale-free network ($n = 100, \gamma = -2.5$) upon the eigenvector centrality π_r of the edge's source node.

Thus, to make sure that addition of candidate edges to the network has a large impact—either positive or negative—upon the average opinion, we can limit ourselves to considering only candidate edges outgoing from high-centrality nodes. Fortunately, the number of such nodes in real-world social networks is small, and most nodes are at the periphery, which justifies our choice of $n_{src} \ll n$.

5.2 Eigenvector Centrality Under Single-Edge Perturbation

In order to tackle DIVER (1), we need to understand how the addition of a single edge $\widetilde{w}_{rc} = \theta_{rc} \in (0, 1)$ from node r to node c changes the network's eigenvector centrality $\pi \to \widetilde{\pi}$. We assume that edge (r, c) is originally absent ($w_{rc} = 0$) and use the same single-edge perturbation model as before (see Fig. 2):

$$\widetilde{W} = W - \theta_{rc}\,\mathrm{diag}(e_r)W + \theta_{rc}e_r e_c^\intercal. \tag{2}$$

In our subsequent perturbation analysis, we will make the following Assumption 1 to improve readability of our theorems.

Assumption 1 (Rational Selfishness). *Users are rationally selfish in that for any user i, $\forall j \neq i : w_{ii} > w_{ij}$. Thus, each user trusts his or her own opinion more than that of any other individual user.*

The following Theorem 4 states how the eigencentrality vector changes under a single-edge perturbation (2).

Theorem 4 (Single-Edge Perturbation). *Under Assumption 1, for a single-edge perturbation (2) of a strongly connected aperiodic network with adjacency matrix W, the network's eigenvector centrality changes as*

$$\widetilde{\pi}_j = \pi_j \left[1 - \frac{\theta_{rc}(m_{cj}(1 - \delta\{j, c\}) - m_{rj} + 1)}{m_{rr} + \theta_{rc}(m_{cr} - m_{rr} + 1)}\right], \tag{3}$$

where m_{ij} is the mean first passage time from state i to j of Markov chain W, and δ is Kronecker delta. In particular,

$$\widetilde{\pi}_r = 1/[m_{rr} + \theta_{rc}(m_{cr} - m_{rr} + 1)]. \tag{4}$$

The proof will rely on the following Theorem 5 due to Hunter [19], provided for reference below.

Theorem 5 ([19, Theorem 4.4]). *Let multiple perturbations occur in r'th row of W. Let $\epsilon_i = \widetilde{W}_{ri} - W_{ri}$, the minimal negative perturbation happen at state a, with $\epsilon_a = -m = \min\{\epsilon_j \mid 1 \leq j \leq n\}$; the maximal positive perturbation occur at state b with $\epsilon_b = M = \max\{\epsilon_j \mid 1 \leq j \leq n\}$. Also, let P be the set of positive perturbation indices, excluding b, and N be the set of negative perturbation indices, excluding a. Then,*

$$\pi_j - \widetilde{\pi}_j = \begin{cases} \pi_a\widetilde{\pi}_r[Mm_{ba} + \sum\limits_{k \in P \cup N} \epsilon_k m_{ka}] & \text{if } j = a, \\ \pi_b\widetilde{\pi}_r[-mm_{ab} + \sum\limits_{k \in P \cup N} \epsilon_k m_{kb}] & \text{if } j = b, \\ \pi_j\widetilde{\pi}_r[-mm_{aj} + Mm_{bj} + \sum\limits_{\substack{k \in P \cup N \\ k \neq j}} \epsilon_k m_{kj}] & \text{if } j \neq a, b. \end{cases}$$

Proof (Theorem 4). Let us apply Theorem 5 to our case of a single-edge perturbation (2). We are adding edge (r, c) with weight θ_{rc} to the network. Due to the form (2) of our single-edge perturbation, the only positive perturbation occurs at the added edge's destination node c, so $b = c$, $\epsilon_c = M = \theta_{rc}$, and $P = \varnothing$. For all the other out-neighbors i of the new edge's source node r, the corresponding perturbations $\epsilon_i = -\theta_{rc}w_{ri}$ are negative. Due to Assumption 1, $\forall i \neq r : w_{rr} > w_{ri}$, so the minimal negative perturbation occurs at $i = r$, and, thus, $a = r$ and $\epsilon_a = -m = -\theta_{rc}w_{rr}$.

Let us first show the validity of (3) in case of $j = r$, that is, (4). According to Theorem 5,

$$\begin{aligned} \pi_r - \widetilde{\pi}_r &= \pi_r\widetilde{\pi}_r\Big[\theta_{rc}m_{cr} + \sum_{k \in P \cup N} (-\theta_{rc}w_{rk})m_{kr}\Big] \\ &= (\text{as } w_{rc} = 0) = \theta_{rc}\pi_r\widetilde{\pi}_r\Big[m_{cr} - \sum_{k \neq r} w_{rk}m_{kr}\Big]. \end{aligned}$$

Using the one-hop conditioning Theorem 2, the obtained expression can be written as

$$\begin{aligned} &\pi_r - \widetilde{\pi}_r = \theta_{rc}\pi_r\widetilde{\pi}_r\,[m_{cr} - m_{rr} + 1] \\ &\Leftrightarrow \widetilde{\pi}_r = \pi_r/(1 + \theta_{rc}\pi_r[m_{cr} - m_{rr} + 1]). \end{aligned}$$

Dividing the numerator and denominator in the right-hand side of the obtained expression by $\pi_r > 0$ (positivity comes from Perron-Frobenius theorem [27, Sec. 8.2]) and using $1/\pi_r = m_{rr}$ (Theorem 1), we obtain (4).

Let us similarly deal with the case $j \neq r, c$. From Theorem 5,

$$\begin{aligned} \pi_j - \widetilde{\pi}_j &= \pi_j\widetilde{\pi}_r\Big[-\theta_{rc}w_{rr}m_{rj} + \theta_{rc}m_{cj} + \sum_{\substack{k \in P \cup N \\ k \neq j}} (-\theta_{rc}w_{rk})m_{kj}\Big] \\ &= \theta_{rc}\pi_j\widetilde{\pi}_r\Big[m_{cj} - w_{rr}m_{rj} - \sum_{k \neq r, c, j} w_{rk}m_{kj}\Big] \\ &= (\text{as } w_{rc} = 0) = \theta_{rc}\pi_j\widetilde{\pi}_r\Big[m_{cj} - \sum_{k \neq j} w_{rk}m_{kj}\Big] \\ &\Leftrightarrow (\text{from Theorem 2}) \Leftrightarrow \widetilde{\pi}_j = \pi_j[1 - \theta_{rc}\widetilde{\pi}_r(m_{cj} - m_{rj} + 1)]. \end{aligned}$$

Substituting (4) in the obtained expression, we get (3) for $j \neq c$. The proof for $j = c$ is similar and, thus, is omitted. □

The following Corollary 1—justifying Sec. 5.1's focus on top-centrality edge source nodes—immediately follows from equation (3) of Theorem 4 used together with Theorem 1.

Corollary 1. *Under perturbation (2) of the network with a single edge (r, c), $\theta_{rc} > 0$, it holds that $\lim_{\pi_r \to 0} \widetilde{\pi} = \pi$, and, thus, $\lim_{\pi_r \to 0} f_\pi(r, c) = \lim_{\pi_r \to 0} (\langle \pi, \widetilde{x} \rangle - \langle \widetilde{\pi}, \widetilde{x} \rangle) = 0$.*

5.3 Average Opinion of the Network Under Single-Edge Perturbation

To solve DIVER, we are interested in adding candidate edges that would result in a large reduction $f_\pi(r, c) = \langle \pi, \widetilde{x} \rangle - \langle \widetilde{\pi}, \widetilde{x} \rangle$ of the average opinion. While Theorem 4 states how different components of the eigencentrality vector change under a single-edge perturbation (2), the following Theorem 6 characterizes the impact of such perturbation upon the value of $f_\pi(r, c)$. The proof of the theorem is immediately obtained by substituting (3) of Theorem 4 into $f_\pi(r, c) = \langle \pi - \widetilde{\pi}, \widetilde{x} \rangle$.

Theorem 6. *Under the rational selfishness Assumption 1, the reduction $f_\pi(r, c) = \langle \pi, \widetilde{x} \rangle - \langle \widetilde{\pi}, \widetilde{x} \rangle$ of the average opinion caused by a single-edge perturbation (2) of W is*

$$f_\pi(r, c) = \theta_{rc} \frac{\sum\limits_{j=1}^{n} \pi_j(m_{cj} \cdot (1 - \delta\{j, c\}) - m_{rj} + 1)\widetilde{x}_j}{m_{rr} + \theta_{rc}(m_{rc} - m_{rr} + 1)}. \tag{5}$$

Essentially, Theorem 6 provides us with an edge score $f_\pi(r, c)$, whose use for candidate edge selection comprises our heuristic for DIVER. Unfortunately, f_π's computation is rather challenging, and is addressed in the following section.

5.4 Efficient Computation of Edge Scores

Computation of candidate edge scores f_π is challenging for two reasons. Firstly, expression (5) involves summation over all n network nodes. Since there are $O(n_{src}n)$ candidate edges—with $n_{src} \ll n$ sources and n destinations—it would result in at least a quadratic-time heuristic for DIVER that would not scale. Secondly, expression (5) involves mean first passage times, whose direct computation is very expensive. We address these challenges separately below.

5.4.1 Focus on a Small Number of Nodes. Our first concern is that expression (5) for f_π contains summation over all n nodes. Intuitively, not all network nodes contribute equally to the value of (5). Indeed, in networks with skewed eigencentrality distribution, such as scale-free networks, f_π is largely determined by a small number of top-centrality nodes. This is illustrated in Fig. 4, that shows the relationship between the exactly computed f_π and its approximations, with different numbers of scale-free network nodes being used in f_π's computation. We can see that, even when we use only

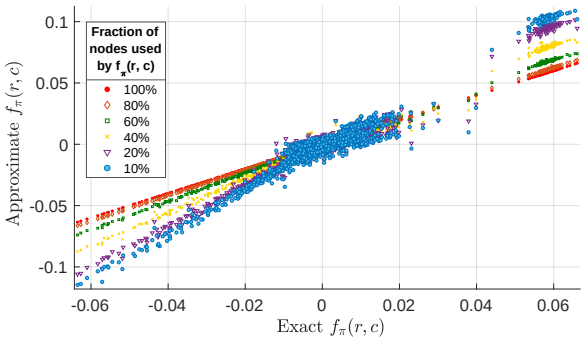

Figure 4: Comparison of exact and approximate candidate edge scores f_π in a scale-free network ($n = 100, \gamma = -2.5$)**.**

10% of nodes, the relative order of f_π for different candidate edges is close to the original, and it is still easy to identify candidate edges (r, c) with the largest values of $f_\pi(r, c)$.

Thus, to efficiently compute $f_\pi(r, c)$, we will use only those j in (5) corresponding to a constant number, e.g., n_{src}, of top-centrality nodes in the network, in addition to $j \in \{r, c\}$.

5.4.2 Efficient Computation of Mean First Passage Times. In the previous section, we have considerably simplified computation of $f_\pi(r, c)$ by leaving only $O(n_{src})$ summands in expression (5). Now, our concern is to actually compute the values of the mean first passage times m_{ij} remaining in (5).

The classic method for exact MFPT computation [21, Theorem 4.4.7] involves computing the fundamental matrix $Z = (I - W + \mathbb{1}\pi^\intercal)^{-1}$ of Markov chain W, and defines MFPTs as

$$M = \{m_{ij}\} = (I - Z + \mathbb{1}\mathbb{1}^\intercal \operatorname{diag}(Z)) \operatorname{diag}^{-1}(\pi).$$

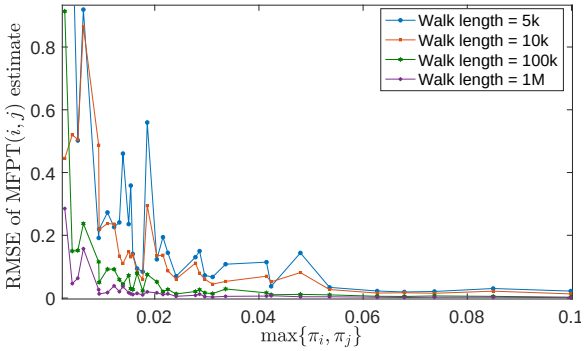

Figure 5: Root mean square error of MFPT estimates in a scale-free network ($n = 100, \gamma = -2.5$) **using walks of different length.**

Computation of the fundamental matrix involves a cubic-time matrix inversion and would not scale. In [20], Hunter surveys alternative methods for MFPT computation, but all of them share the same high complexity. Most importantly, however, all existing methods target computation of all $O(n^2)$ MFPTs between all network nodes.

Let us notice that expression (5) for f_π uses MFPTs either from or to high-centrality nodes: r are top-centrality according to Sec. 5.1; j are top-centrality according to Sec. 5.4.1. There are $n_{src}n \ll n^2$ such MFPTs, where n_{src} is the number of candidate edge source nodes r.

We propose to estimate the MFPTs between a small number of nodes by performing a finite random walk over the network and tracking first passage times between its nodes. The walk starts at an arbitrary node and proceeds for a predefined number of steps following the transition probabilities defined by the adjacency matrix W, viewed here as the state transition matrix of a Markov chain. While performing the walk, we accumulate the passage times between n_{src} candidate edge sources and n candidate edge destinations, and compute the means when the walk is complete. This approach towards MFPT estimation is similar to the k-Step Markov Approach that White and Smyth [33, Sec. 6.4] used for estimation of their MFPT-based relative importance of network nodes.

The key questions here are *whether the proposed method will produce good estimates of MFPTs to and from high-centrality nodes* and, *if so, how long the random walk should be*. We answer these two questions via empirical analysis.

The first insight is that MFPTs to and from high-centrality nodes converge very fast, since the walk visits such nodes most often. This is illustrated in Fig. 5, according to which the error of MFPT estimates m_{ij} noticeably varies with the walk's length when both i and j are low-centrality, and is uniformly low if at least one of i and j is high-centrality. This insight echoes the result of Avrachenkov et al. [4], who show that PageRank of high-centrality nodes estimated via Monte Carlo simulation converge very fast.

Now, we empirically study the question of how long the random walk should be to obtain sufficiently good estimates of MFPTs to and from top-centrality nodes in a scale-free network. The results are reported in Fig. 6, that shows how many steps a random walk should perform in order for 5% of MFPTs to and from top 5% high-centrality nodes to converge within 5% of their true values, while the network's size n and scale-free exponent γ vary. For each pair

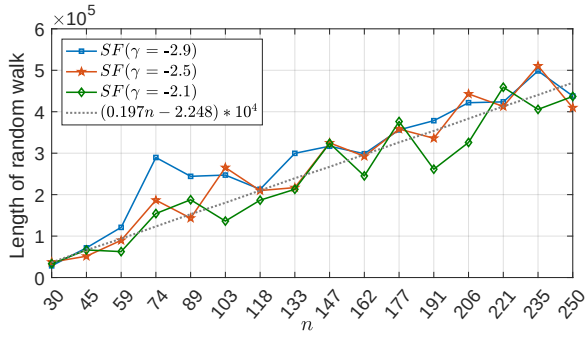

Figure 6: Dependency of the length of a random walk—used for estimating MFPTs to and from top-centrality nodes—on the size and density of the scale-free network.

(n, γ), 100 networks are generated, and the mean walk lengths are reported. The results are reported for 3 specific scale-free exponents, $\gamma \in \{-2.9, -2.5, -2.1\}$. The length of the random walk does not depend on the scale-free exponent, and depends upon the network's size n as $(0.197n - 2.248) \cdot 10^4$. This result allows to make the following statement.

Proposition 1 (Random Walk Length). *In scale-free networks, the length of a random walk sufficient for convergence of $O(n)$ MFPTs to and from $O(1)$ top-centrality nodes is $O(n)$ (in contrast to $O(n^3)$ cost of the direct computation of all MFPTs via the fundamental matrix method).*

5.5 Solving DIVER

In this section, we gather all our results, formally state a heuristic for solving DIVER, and analyze its complexity.

Algorithm 1 Heuristic for DIVER

Input: W—sparse row-stochastic irreducible adjacency matrix of the network; k—number of new edges to add; n_{src}—maximal number of new edges' sources.
Output: sequence $(r_1, c_1), (r_2, c_2), \ldots$ of new edges to add

1: Compute eigencentrality π (dominant left eigenvector of W)
2: Define candidate edge sources: $R \leftarrow n_{src}$ top-centrality nodes
3: Estimate MFPTs $\{m_{ij}\}$ to and from each $r \in R$
4: **for** $r \in R, c \in \{1, \ldots, n\}$ **do**
5: Estimate $f_\pi(r, c)$ using $O(n_{src})$ top-centrality nodes
6: **end for**
7: $S \leftarrow$ candidate edges (r, c) having top-k scores $f_\pi(r, c)$
8: **return** S

Theorem 7. *Time-complexity of Algorithm 1 is $O(n(\text{gap}(W) + n_{src}^2) + n_{src} \log n_{src} + k \log k)$, where $\text{gap}(W)$ is the number of matrix-vector multiplications the power method uses to compute the dominant left eigenvector of W.*

Proof. In step 1 of Algorithm 1, we compute the dominant left eigenvector π of W using the power method, performing $\text{gap}(W)$ matrix-vector multiplications, each of which has a linear time complexity for sparse W. Thus, this step's complexity is $T_1 = O(\text{gap}(W)n)$. The cost of selecting top n_{src} elements out of n at Step 2 is $T_2 = O(n + n_{src} \log(n_{src}))$. In step 3, following Sec. 5.5.2 and, in particular, Proposition 5.1, we estimate MFPTs via a $O(n)$-long finite random walk, so this step's cost is $T_3 = O(n)$. At steps 4-6, we compute $n_{src}n$ edge scores f_π. Following the method of Sec. 5.5.1, each $f_\pi(r, c)$ is computed in time $O(n_{src})$, bringing time complexity of steps 4-6 to $T_{4-6} = O(n_{src}^2 n)$. Finally, selection of top k out of $n_{src}n$ items at step 7 is performed in time $T_7 = O(n_{src}n + k \log k)$. If we collect the expressions for $T_1, \ldots, T_7$, we get $T = O(n(\text{gap}(W) + n_{src}^2) + n_{src} \log n_{src} + k \log k)$. □

In Theorem 7, the number $\text{gap}(W)$ of iterations it takes the power method to converge depends on W's spectral gap λ_2/λ_1, but, in practice, $\text{gap}(W)$ usually can be assumed to be a reasonably small constant [4]. Thus, assuming that $\text{gap}(W)$ is bounded, as well as noticing that we choose both n_{src} and k to be small, that is, $n_{src} \ll n$ and $k \ll n$, it immediately follows from Theorem 7 that *Algorithm 1 is computable in time $O(n)$*.

6 EXPERIMENTAL RESULTS

In this section, we experimentally study Algorithm 1's performance on synthetic and real-world networks. We start with experimental setup, and, then, study performance of our heuristic. Additional experiments studying our heuristic's effectiveness, efficiency, as well as robustness to network noise can be found in the Supplement [2].

6.1 Experimental Setup

6.1.1 Networks. We have experimented with three synthetic and three real-world networks:

▷ **SF**(n, γ): a scale-free network with n nodes and scale-free exponent γ, built using a greedy generator that constructs a directed graph trying to match a given degree distribution[1].

▷ **BA**(n, m_{links}): a Barabási-Albert network on n nodes with a seed SF$(0.01n, -2.5)$ and m_{link} edges created per node / iteration.

▷ **ER**$(n, \mathbb{P}_{edge})$: an Erdős–Rényi network with the edge probability $\mathbb{P}_{edge}$, being an example of a "non-scale-free-like" network, on which our heuristic and baselines perform poorly.

▷ **Karate**: Zachary's Karate Club network.

▷ **Facebook**: a 4k-node part of Facebook graph [25].

▷ **Epinions**: a 32k-node part of the mutual trust network epinions.com [29].

Network	$\lvert V\rvert$	$\lvert E\rvert$	S_π	S_d	K_π	K_d
SF(1024, −2.5)	1024	3.9k	5.30	9.37	54.88	134.56
BA(1024, 3)	1024	7.9k	7.51	7.46	103.34	84.26
ER(1024, 0.25)	1024	261k	0.04	0.10	2.95	3.01
Karate	34	190	1.09	2.00	3.24	6.30
Facebook	4039	181k	4.24	4.52	19.67	57.56
Epinions	32k	476k	9.87	9.02	130.01	160.64

Table 2: Summary of networks.

The characteristics of these networks are summarized in Table 2, where S_π and K_π are skewness and kurtosis, respectively, of the unweighted network's eigencentrality distribution, and S_d and K_d are the same metrics of the same network's total (in- plus out-) degree distribution. All the networks except Facebook are directed. If the original network was not strongly connected, we replace it with its largest strongly connected component. We also add all self-loops—since each user in our case is supposed to have some amount of

[1] EvaluateGraphCreateRandomGraph.cpp of Complex Networks Package.

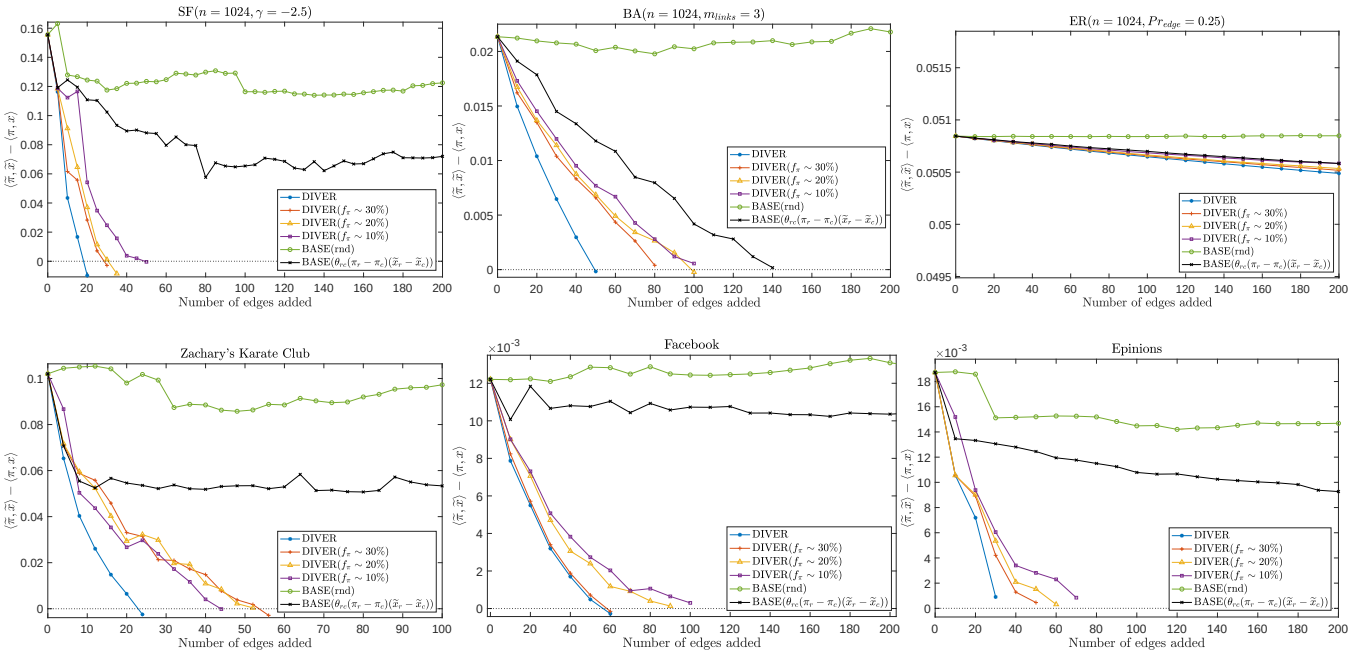

Figure 7: Solving DIVER **over synthetic and real-world networks.**

trust in her or his own opinion—and draw edge weights uniformly at random, while maintaining self-weight dominance—as per Assumption 1—and rescaling the weights in each out-neighborhood to make sure the resulting adjacency matrix is row-stochastic.

6.1.2 Opinions and Attack Upon Them. The user opinions x are drawn uniformly at random from $[0, 1]^n$. To simulate an attack and create a new opinion distribution $\widetilde{x}$, $0.1n$ users (5 for Karate network) are chosen uniformly at random, and their opinions change to 1, while the opinions of other users stay intact. We assume that the social network platform has successfully detected the attack, having recognized that an abnormal change has happened to x, but having no information about the location of the attack. Now, knowing both x and $\widetilde{x}$, the goal is to recommend links to the users, restoring the network's average opinion. To that end, we will compare several methods described in the following section.

6.1.3 Methods. We experiment with several versions of our heuristic, as well as two baselines. Note that there is no state of the art, as no existing method can solve our optimization problem.

▷ DIVER: uses Algorithm 1 with f_π being computed using *all* rather than $O(1)$ top-centrality nodes, and estimates MFPTs using the procedure of Sec. 5.4.2. The heuristic adds k edges at a time, improving the distribution of eigencentralities $\widetilde{\pi}$ either until the average opinion $\langle \widetilde{\pi}, \widetilde{x} \rangle$ gets close enough to its original state $\langle \pi, x \rangle$, or a maximum of k_{max} new edges is reached. The number n_{src} of top-centrality nodes considered in computation is 20 for synthetic, 10 for Karate, 40 for Facebook, and 200 for Epinions networks. This version of Algorithm 1 has quadratic time complexity, and represents "the best case" behavior of DIVER heuristic.

▷ DIVER($f_\pi \sim X\%$), $X \in \{10, 20, 30\}$: similar to DIVER above, except that Algorithm 1 estimates f_π using only a fraction X of all the nodes. The time complexity of these methods is pseudo-linear, as per Theorem 7, and lower values of X correspond to a higher computational efficiency, yet, to a lower effectiveness (more new edges are spent to recover the original average opinion).

▷ BASE(rnd): the worst-case baseline, that selects new edges uniformly at random. It is expected to keep the average opinion at about the same level.

▷ BASE($\theta_{rc}(\pi_r - \pi_c)(\widetilde{x}_r - \widetilde{x}_c)$): this baseline attempts to effectively reduce the average opinion $\langle \pi, \widetilde{x} \rangle$ by adding heavy-weight edges from higher-centrality nodes r having larger opinion values to lower-centrality nodes c having lower opinion values, thereby, reducing the contribution of $\pi_r \widetilde{x}_r$ and increasing that of $\pi_c \widetilde{x}_c$ to the value of $\langle \pi, \widetilde{x} \rangle$. This baseline—similarly to DIVER—ranks only the candidate edges outgoing from n_{src} top-centrality nodes, and, hence, is computable in pseudo-linear time ("pseudo" since it uses π). The main qualitative difference of this baseline from DIVER is that it uses only absolute centrality information π_i, while DIVER takes into account both absolute and relative "centrality" m_{ij}.

6.1.4 Evaluation. For all these methods, we assess their performance based on how many candidate edges a method uses to restore the original average opinion.

6.2 Solving DIVER

In this section, we solve DIVER using the previously described networks and methods. All these methods add $k = 10$ edges at a time, improving the distribution of eigencentralities $\widetilde{\pi}$ either until the average opinion $\langle \widetilde{\pi}, \widetilde{x} \rangle$ gets close enough to its original state $\langle \pi, x \rangle$, or a maximum of $k_{max} = 200$ new edges is reached. The results are displayed in Fig. 7.

We can see that our heuristic works well on all scale-free-like networks, disabling attacks by adding 20-60 candidate edges. In case of ER network, the quality of all absent edges is uniformly

low, which naturally leads to all the methods' performing uniformly poorly. For the case of Karate network—which, similarly to ER network, has rather low skewness and kurtosis of eigencentrality and degree distributions as per Table 2—performance of DIVER heuristic degrades rather fast with the number of nodes used in computation of f_π; for example, using 30% of nodes to compute f_π, it takes DIVER($f_\pi \sim 30\%$) twice as many new edges as DIVER to restore the average opinion. We also see that baseline BASE($\theta_{rc}(\pi_r - \pi_c)(\widetilde{x}_r - \widetilde{x}_c)$) performs well only on BA network, and underperforms on all the other networks. Notice, however, that during the first one-two iterations of edge addition, this baseline performs almost as well as DIVER does, as both methods add edges outgoing from very high-centrality nodes, which appear to be the best according to both the absolute centrality-based assessment of the baseline, and the relative centrality-based assessment of DIVER. However, during the subsequent iterations, the relative centrality information starts playing a crucial role for edge addition impact assessment, and the baseline's performance rapidly degrades, while DIVER continues to perform well.

7 CONCLUSION

In this work, we formulated DIVER—a new problem of strategically recommending links in a social network to disable the effect of malicious external control of user opinions. Due to NP-hardness of this problem, we focused on designing a heuristic for it. To that end, relying on the theory of Markov chains, we provided a perturbation analysis, formally answering the question of how the network nodes' eigencentralities and, thus, DIVER's objective change when a edge is added to the network. This analysis led to the definition of candidate edge scores that quantify the potential impact of candidate edges, allowing to add them to the network in a greedy fashion. We also provided insights into how to compute these edge scores in scale-free-like networks in pseudo-constant time, which resulted in a pseudo-linear-time heuristic for DIVER. One of these insights is related to efficient estimation of mean first passage times in Markov chains. We confirmed our theoretical and algorithmic findings in experiments with synthetic and real-world networks. In particular, we showed the importance of taking into account relative node centrality information when dealing with strategic manipulation of absolute centrality. Future work includes adapting DIVER to the optimization of the opinion distribution rather than the average opinion, deriving formal convergence bounds for MFPT estimation, and evening the quality of MFPT estimates by biasing the random walk. Finally, and more importantly for the society, we should study how to ensure that link recommendation is not misused by online social network platforms as a tool for malicious opinion control.

REFERENCES

[1] Victor Amelkin, Petko Bogdanov, and Ambuj K. Singh. 2017. A Distance Measure for the Analysis of Polar Opinion Dynamics in Social Networks. In *Proc. of Intern. Conf. on Data Engineering (ICDE)*. IEEE, 159–162.

[2] Victor Amelkin and Ambuj K. Singh. 2019. Fighting Opinion Control in Social Networks via Link Recommendation (Supplementary Materials). https://victoramelkin.com/pub/diver/sup/.

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

[4] Konstantin Avrachenkov, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. 2007. Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.* 45, 2 (2007), 890–904.

[5] Bahman Bahmani, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. 2012. PageRank on an evolving graph. In *Proc. of Conf. on Knowl. Discovery and Data Mining (KDD)*. ACM, 24–32.

[6] Hau Chan, Leman Akoglu, and Hanghang Tong. 2014. Make it or break it: Manipulating robustness in large networks. In *Proc. of Intl. Conf. on Data Mining (SDM)*. SIAM, 325–333.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2012. Anomaly detection for discrete sequences: A survey. *Trans. on Knowledge and Data Engineering (TKDE)* 24, 5 (2012), 823–839.

[8] Chen Chen and Hanghang Tong. 2017. On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10, 2 (2017), 121–135.

[9] Chen Chen, Hanghang Tong, B Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2016. Eigen-optimization on large graphs by edge manipulation. *ACM Trans. on Knowledge Discovery from Data* 10, 4 (2016), 49.

[10] Steve Chien, Cynthia Dwork, Ravi Kumar, and D Sivakumar. 2001. Towards exploiting link evolution. In *Workshop on Algorithms and Models for the Web*.

[11] Grace E Cho and Carl D Meyer. 2000. Markov chain sensitivity measured by mean first passage times. *Linear Algebra Appl.* 316, 1-3 (2000), 21–28.

[12] Giacomo Como and Fabio Fagnani. 2015. Robustness of large-scale stochastic matrices to localized perturbations. *IEEE Trans. on Network Science and Engineering* 2, 2 (2015), 53–64.

[13] Balázs Csanád Csáji, Raphaël M Jungers, and Vincent D Blondel. 2014. PageRank optimization by edge selection. *Discrete Applied Mathematics* 169 (2014), 73–87.

[14] Cristobald de Kerchove, Laure Ninove, and Paul Van Dooren. 2008. Maximizing PageRank via outlinks. *Linear Algebra Appl.* 429, 5-6 (2008), 1254–1276.

[15] Leon Festinger. 1954. A theory of social comparison processes. *Human Relations* 7, 2 (1954), 117–140.

[16] Leon Festinger. 1962. *A Theory of Cognitive Dissonance*. Vol. 2. Stanford Press.

[17] Arpita Ghosh and Stephen Boyd. 2006. Growing well-connected graphs. In *Proc. of Conf. on Decision and Control*. IEEE, 6605–6611.

[18] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. 2010. Learning influence probabilities in social networks. In *Proc. of Intern. Conf. on Web Search and Data Mining (WSDM)*. ACM, 241–250.

[19] Jeffrey J. Hunter. 2005. Stationary distributions and mean first passage times of perturbed Markov chains. *Linear Algebra Appl.* 410, 1-3 (2005), 217–243.

[20] Jeffrey J Hunter. 2018. The computation of the mean first passage times for Markov chains. *Linear Algebra Appl.* 549 (2018), 100–122.

[21] John G Kemeny and James Laurie Snell. 1976. *Finite Markov Chains*. Springer.

[22] Amy N Langville and Carl D Meyer. 2006. Updating Markov chains with an eye on Google's PageRank. *SIAM J. Matrix Anal. Appl.* 27, 4 (2006), 968–987.

[23] Long T Le, Tina Eliassi-Rad, and Hanghang Tong. 2015. MET: A fast algorithm for minimizing propagation in large graphs with small eigen-gaps. In *Proc. of Intern. Conf. on Data Mining (SDM)*. SIAM, 694–702.

[24] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proc. of Intern. Conf. on Knowledge Discovery and Data Mining (KDD)*. ACM, 420–429.

[25] Jure Leskovec and Julian J McAuley. 2012. Learning to discover social circles in ego networks. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*. 539–547.

[26] Liangyue Li, Hanghang Tong, Yanghua Xiao, and Wei Fan. 2015. Cheetah: Fast graph kernel tracking on dynamic graphs. In *Proc. of Intern. Conf. on Data Mining (SDM)*. SIAM, 280–288.

[27] Carl D Meyer. 2000. *Matrix Analysis and Applied Linear Algebra*. Vol. 71. SIAM.

[28] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14, 1 (1978), 265–294.

[29] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. 2003. Trust management for the semantic web. In *Proc. of International Semantic Web Conference (ISWC)*. Springer, 351–368.

[30] Sudip Saha, Abhijin Adiga, B Aditya Prakash, and Anil Kumar S Vullikanti. 2015. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *Proc. of Intern. Conf. on Data Mining (SDM)*. SIAM, 568–576.

[31] Hanghang Tong, Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2012. Gelling and melting large graphs by edge manipulation. In *Proc. of Intl. Conf. on Information and Knowledge Management (CIKM)*. ACM, 245–254.

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

[33] Scott White and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proc. of Conf. on Knowl. Discovery and Data Mining (KDD)*. ACM, 266–275.

[34] Zhi Yu, Can Wang, Jiajun Bu, Xin Wang, Yue Wu, and Chun Chen. 2015. Friend recommendation with content spread enhancement in social networks. *Inform. Sciences* 309 (2015), 102–118.

[35] Yao Zhang, Abhijin Adiga, Anil Vullikanti, and B Aditya Prakash. 2015. Controlling propagation at group scale on networks. In *Proc. of Intern. Conf. on Data Mining (ICDM)*. IEEE, 619–628.