Abstract

The largest eigenvalue of the adjacency matrix of a network (referred to as the spectral radius) is an important metric in its own right. Further, for several models of epidemic spread on networks (e.g., the ‘flu-like’ SIS model), it has been shown that an epidemic dies out quickly if the spectral radius of the graph is below a certain threshold that depends on the model parameters. This motivates a strategy to control epidemic spread by reducing the spectral radius of the underlying network.

In this paper, we develop a suite of provable approximation algorithms for reducing the spectral radius by removing the minimum cost set of edges (modeling quarantining) or nodes (modeling vaccinations), with different time and quality tradeoffs. Our main algorithm, GreedyWalk, is based on the idea of hitting closed walks of a given length, and gives an $O(\log^2 n)$-approximation, where $n$ denotes the number of nodes; it also performs much better in practice compared to all prior heuristics proposed for this problem. We further present a novel sparsification method to improve its running time.

In addition, we give a new primal-dual based algorithm with an even better approximation guarantee ($O(\log n)$), albeit with slower running time. We also give lower bounds on the worst-case performance of some of the popular heuristics. Finally we demonstrate the applicability of our algorithms and the properties of our solutions via extensive experiments on multiple synthetic and real networks.

1 Introduction

Given a contact network, which contacts should we remove to contain the spread of a virus? Equivalently, in a computer network, which connections should we cut to prevent the spread of malware? Designing effective and low cost interventions are fundamental challenges in public health and network security. Epidemics are commonly modeled by stochastic diffusion processes, such as the so-called ‘SIS’ (flu-like) and ‘SIR’ (mumps-like) models on networks (more in Section 2). An important result that highlights the impact of the network structure on the dynamics is that epidemics die out “quickly” if $\rho(G) \leq T$, where $\rho(G)$ is the spectral radius (or the largest eigenvalue) of graph $G$, and $T$ is a threshold that depends on the disease model [13, 39, 30]. This motivates the following strategy for controlling an epidemic: remove edges (quarantining) or nodes (vaccinating) to reduce the spectral radius below a threshold $T$—we refer to this as the spectral radius minimization (SRM) problem, with variants depending on whether edges are removed (the SRME problem) or whether nodes are removed (the SRMN problem). Van Mieghem et al. [27] and Tong et al. [37] prove that this problem is NP-complete. They also study two heuristics for it, one based on the components of the first eigenvector (EIGENS) and another based on degrees (PRODUCTDEGREE). However, no rigorous approximations were known for the SRM or the SRMN problems.

Our main contributions.

1. Lower bounds on the worst-case performance of heuristics: We show that the PRODUCTDEGREE, EIGENS and PAGERANK heuristics (defined formally in Section 2) can perform quite poorly in general. We demonstrate graph instances where these heuristics give solutions of cost $\Omega(\frac{n^2}{\rho(G)^2})$ times the optimal, where $n$ is the number of nodes in the graph.

2. Provable approximation algorithms: We present two bicriteria approximation algorithms for the SRME and SRMN problems, with varying approximation quality and running time tradeoffs. Our first algorithm, GreedyWalk, is based on hitting closed walks in $G$. We show this algorithm has an approximation bound of $O(\log n \log \Delta)$ times optimal for the cost of edges removed, while ensuring that the spectral radius becomes at most $(1 + \epsilon)$ times the threshold, for $\epsilon$ arbitrarily small (here $\Delta$ denotes the maximum node degree in the graph). We also design a variant, GreedyWalkSparse, that performs careful sparsification of the graph, leading to similar asymptotic guarantees, but better running time, especially when the threshold $T$ is small. We then develop algorithm PrimalDual, which improves this approximation bound to an $O(\log n)$ using a more sophisticated primal-dual approach, at the...
expense of a slightly higher (but polynomial) running time.

### 3. Empirical analysis
We conduct an extensive experimental evaluation of GreedyWalk, a simplified version of PrimalDual and different heuristics that have been proposed for epidemic containment on a diverse collection of synthetic and real networks. For SRME, these heuristics involve picking edges $e = (i, j)$ in non-increasing order of some kind of score; the specific heuristics we compare include: (i) PRODUCT-Degree, (ii) EigenScore, (iii) LinePagerank, and (iv) Hybrid, which pick the edge based on either the eigenscore or the product-degree ordering, depending on the maximum decrease in eigenvalue. We find that GreedyWalk performs better than all the heuristics in all the networks we study. In the experiments, we analyze GreedyWalk for walks of length $k = \Theta(\log n)$.

### Organization
The background and notation are defined in Section 2. Sections 3, 4 and 5 cover GreedyWalk, GreedyWalkSparse and PrimalDual algorithms, respectively, for the SMRE problem; the SRMN problem is discussed in section 6. Some of the algorithmic details and proofs are omitted for brevity and are available in [34]. Lower bounds for some heuristics and the experimental results are discussed in Sections 7 and 8, respectively. We discuss the related work in Section 9 and conclude in Section 10.

## 2 Preliminaries

<table>
<thead>
<tr>
<th>Table 1: Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = (V, E)$</td>
</tr>
<tr>
<td>$n =</td>
</tr>
<tr>
<td>$d(v, G)$</td>
</tr>
<tr>
<td>$\Delta(G)$</td>
</tr>
<tr>
<td>$A = A^G$</td>
</tr>
<tr>
<td>$G(E')$</td>
</tr>
<tr>
<td>$\lambda_i(G)$</td>
</tr>
<tr>
<td>$\rho(G) = \rho(A)$</td>
</tr>
<tr>
<td>$c(\cdot)$</td>
</tr>
<tr>
<td>$\beta$</td>
</tr>
<tr>
<td>$\delta$</td>
</tr>
<tr>
<td>$T$</td>
</tr>
<tr>
<td>$\tau$</td>
</tr>
</tbody>
</table>

We consider undirected graphs $G = (V, E)$, and interventions to control the spread of epidemics—vaccination (modeled by removal of nodes) and quarantining (modeled by removal of edges). There can be different costs for the removal of nodes and edges (denoted by $c(v)$ and $c(e)$, respectively), e.g., depending on their demographics, as estimated by [25]. For a set $E' \subseteq E$, $c(E') = \sum_{e \in E'} c(e)$ denotes the total cost of the set $E'$ (similarly for node subsets).

There are a number of models for epidemic spread; we focus on the fundamental SIS (Susceptible-Infectious-Susceptible) model, which is defined in the following manner. Nodes are in susceptible (S) or infectious (I) states. Each infected node $u$ (in state I) causes each susceptible neighbor $v$ (in state S) to become infected at rate $\beta_{uv}$. Further, each infected node $u$ switches to the susceptible state at rate $\delta$. In this paper, we assume a uniform rate $\beta_{uv} = \beta$ for all $(u, v) \in E$; in this case, we define a threshold $T = \delta/\beta$, which characterizes the time to extinction. Let $A = A^G$ denote the adjacency matrix of $G$, and let $n = |V|$. Let $\lambda_i(G)$ denote the $i$th largest eigenvalue of $A$, and let $\rho(A) = \lambda_1(A)$ denote the spectral radius of $A$. Since $G$ is undirected, it follows that all eigenvalues are real, and $\rho(A) > 0$ (see, e.g., Chapter 3 of [26]). Ganesh et al. [13] showed that the epidemic dies out in time $O(\frac{\delta}{\beta n})$, if $\rho(A) < T$ in the SIS model, with high probability; this threshold was also observed by [39]. Prakash et al. [30] showed this condition holds for a broad class of other epidemic models, including the SIR model (which contains the ’Recovered’ state). Now we formally define the SRM problem.

**Definition 2.1. Spectral Radius Minimization problem (SRME and SRMN):** Given an undirected graph $G = (V, E)$, with cost $c(e)$ for each edge $e$, and a threshold $T$, the goal of the SRME($G, c(\cdot), T$) problem is to find the cheapest subset $E' \subseteq E$ such that $\lambda_1(G[E \setminus E']) < T$. We refer to the node version of this problem as SRMN($G, c(\cdot), T$).

We discuss some notation that will be used in the rest of the paper. $E_{\text{OPT}}(T)$ denotes an optimal solution to the SRME($G, c(\cdot), T$) problem. Let $W_k(G)$ denote the set of closed walks of length $k$ in $G$; let $W_k(G) = |W_k(G)|$. For a walk $w$, let $nodes(w)$ denote the number of distinct nodes in $w$. We refer to nodes($w$) as nodes in $w$. A standard result (see, e.g., Chapter 3 of [26]) is the following:

\[
\sum_{w \in W_k(G)} \text{nodes}(w) = \sum_i A_{ii}^k = \sum_{i=1}^n \lambda_i(G)^k.
\]

The number of walks in $W_k(G)$ containing a node $i$ is $A_{ii}^k$. For a graph $G$, let $\text{walks}(e, G, k)$ denote the number of closed $k$-walks in $G$ containing $e = (i, j)$. Then, $\text{walks}(e, G, k) = A_{ij}^{k-1}$. We say that an edge set $E'$ hits a walk $w$ if $w$ contains an edge from $E'$. Similarly, for a node $v$, let $\text{walks}(v, G, k)$ denote the number of closed
k-walks in G containing v. Then, walks(i, G, k) = \(A_k^i\).
Table 1 summarizes the frequently used notations.

3 GreedyWalk: \(O(\log n \log \Delta)\)-approximation

Main idea. Our starting point is the connection between the number of closed walks in a graph and the sum of powers of the eigenvalues in (2.1). We try to reduce the spectral radius by reducing the number of closed walks of length k in the graph, by removing edges (see Algorithm 1). This, in turn, can be viewed as a partial covering problem. Our basic idea extends to the node version as well, as discussed later in Section 6.

Algorithm 1 GreedyWalk (high level description)

Input: G, T, c(·), k even
Output: Edge set \(E'\)
1: Initialize \(E' \leftarrow \emptyset\)
2: while \(W_k(G[E \setminus E']) \geq nT^k\) do
3: \(r \leftarrow W_k(G[E \setminus E']) - nT^k\)
4: Pick \(e \in E[E \setminus E']\) that maximizes \(\min(r, \text{walks}(e, G[E'], k))\)
5: \(E' \leftarrow E' \cup \{e\}\)
6: end while

The Lemma below proves the approximation bound for any solution (say \(E'\)) from GreedyWalk. Let \(G' = G[E \setminus E']\) denote the graph resulting after the removal of edges in \(E'\). Our proof involves three steps: (1) Proving the bound on \(\lambda_1(G')\); (2) Relating \(c(E')\) to the cost of the optimum solution to the partial covering problem which ensures that the number of walks in the residual graph is at most \(nT^k\); (3) Showing that the optimum solution to the SRME problem also ensures that at most \(nT^k\) remain in the residual graph.

Lemma 3.1. Let \(E'\) denote the set of edges found by Algorithm GreedyWalk. Given any constant \(\epsilon > 0\), let \(k\) be an even integer larger than \(\frac{\log n}{\log(1+\epsilon/4)}\). Then, we have \(\lambda_1(G[E \setminus E']) \leq (1 + \epsilon)T\), and \(c(E') = O(c(E_{\text{OPT}}(T))) \log n \log \Delta\).

Proof. We follow the proof scheme mentioned above. By the stopping condition of the algorithm, we have \(W_k(G') \leq nT^k\). From (2.1), we have \(\sum_{i=1}^{n} \lambda_i(G')k = \sum_{i}A_k^i = \sum_{w \in W_k(G')} \text{nodes}(w) \leq kW_k(G')\), which implies \(\sum_{i=1}^{n} \lambda_i(G')k \leq nkT^k\). Further, since \(k\) is even

(by assumption), \(\lambda_i(G') \geq 0\), so that \(\lambda_1(G')k \leq \sum_{i=1}^{n} \lambda_i(G')k \leq nkT^k\). This implies \(\lambda_1(G') \leq e^{(\log n + \log k)/k}T\). Since \(k = \log n / \log(1 + \epsilon/3)\), we have \((\log n + \log k)/k \leq 2\log(1 + \epsilon/3)\), so that \(\lambda_1(G') \leq (1 + \epsilon/3)^2T \leq (1 + \epsilon)T\).

Next, we derive a bound for \(c(E')\). Observe that the algorithm can be viewed as solving a partial cover problem, in which (i) the set \(H\) of elements corresponds to walks in \(W_k(G)\), and (ii) there is a set corresponding to each edge \(e \in E\) consisting of all the walks in \(W_k(G)\) that contain \(e\). Following the analysis of the greedy algorithm for partial cover [35], we have \(c(E') = O(c(E_{\text{HITOPT}}) \log |H|)\), where \(E_{\text{HITOPT}}\) denotes the optimum solution for this covering instance. Since \(\Delta\) denotes the maximum node degree, we have \(H = W_k(G) \leq n\Delta^k\). We show below that \(c(E_{\text{HITOPT}}) \leq c(E_{\text{OPT}}(T))\), it follows that \(c(E') = O(c(E_{\text{OPT}}(T))) \log n \log \Delta\).

Finally, we prove that \(c(E_{\text{HITOPT}}) \leq c(E_{\text{OPT}}(T))\). By definition of \(E_{\text{OPT}}(T)\), we have \(\lambda_1(G[E - E_{\text{OPT}}(T)]) \leq T\). Let \(G'' = G[E - E_{\text{OPT}}(T)]\). Then, we have

\[W_k(G'') \leq \sum_{i=1}^{n} \lambda_i(G'')k < n\lambda_1(G'')k \leq nT^k\]

This implies \(E_{\text{OPT}}(T)\) hits at least \(W_k(G) - nT^k\) walks, so that \(c(E_{\text{HITOPT}}) \leq c(E_{\text{OPT}}(T))\).

Effect of the walk length \(k\). We set the walk length \(k = a \log n\) for some constant \(a\) in Algorithm GreedyWalk: understanding the effect of \(k\) is a natural question. From the proof of Lemma 3.1, it follows that \(\lambda_1(G[E \setminus E'])\) can be bounded by \((nk)^{1/k}T\) for any choice of \(k\), as long as it is even. This bound becomes worse as \(k\) becomes smaller, e.g., it is \(O(\sqrt{n})\) for \(k = 2\).

In order to complete the description of GreedyWalk (Algorithm 1 ), we need to design an efficient method to determine the edge which maximizes the quantity in line 4. We discuss two methods below.

3.1 Matrix multiplication approach for implementing GreedyWalk. Note that walks(e, G, k) = \(A_k^{e-1}\). We use matrix multiplication to compute \(A_k\) once for each iteration of the while loop in line 2 of Algorithm 1. In line 4, we iterate over all edges, in order to compute the edge \(e\) that maximizes the given ratio. For \(k = O(\log n)\), \(A_k\) can be computed in time \(O(n^\omega \log \log n\), where \(\omega < 2.37\) is the exponent for the running time of the best matrix multiplication algorithm [40]. Therefore, each iteration involves \(O(n^\omega \log \log n \cdot m) = O(n^\omega \log \log n\) time. This gives a total running time of \(O(n^\omega \log \log n / E_{\text{OPT}}(T)^2\), since only \(O([E_{\text{OPT}}(T)]^2)\) edges are removed. One draw-
back with this approach is the high (super-linear) space complexity, even with the best matrix multiplication methods, in general.

3.2 Dynamic programming approach for implementing GreedyWalk. When the graphs are very sparse (Θ(n) edges), we adapt a dynamic programming approach to compute walks(e, G, k) for an edge e and more efficiently select the edge that maximizes \( \text{walks}(e, G[E \setminus E'], k)/c(e) \) in line 4 of Algorithm 1. Although, potentially \( \text{walks}(e, G, k) \) needs to be computed for each edge \( e \in E \setminus E' \), in practice it suffices to compute it for only a small subset of \( E \setminus E' \). We make use of the fact that walks(e, G', k) ≤ walks(e, G, k) for any subgraph G'. The approach is briefly as follows. Initially we compute walks(e, G, k) for each \( e \in E \) and arrange the edges in non-decreasing order of their walks(e, G, k) value, \( e_1, e_2, \ldots, e_{|E|} \). After the first edge (i.e. \( e_1 \) in the first iteration) is removed, walks(e, G', k) is computed on the residual graph G' only for some consecutive edges in that order up to some \( e_i \) such that \( \text{walks}(e_i, G', k) > \text{walks}(e_{i+1}, G, k) \). Edges \( e_2, \ldots, e_i \) are reordered based on the recomputed walk numbers, walks(e_i, G', k) and then the same steps are repeated. The approach takes \( O(n) \) space and \( O(n^2k) \) time assuming the number of edges is \( \Theta(n) \) in real world large networks. The algorithmic details and proofs are omitted for brevity.

4 Using sparsification for faster running time: Algorithm GreedyWalkSparse

The efficiency of Algorithm GreedyWalk can be improved if the number of edges in the graph can be reduced. This can be achieved by two pruning steps - pruning edges such that in the residual graph (i) no node has degree more than \( T^2 \), and (ii) there is no \( T \)-core; the \( T \)-core of a graph denotes the maximal subgraph of \( G \) with minimum degree \( T \) (see, e.g., [1]). We will refer to these steps as \( \text{MaxDegreeReduction} \) and \( \text{DensityReduction} \) respectively. This leads to sparser graphs, without affecting the asymptotic approximation guarantees.

**LEMMA 4.1.** Let \( E_1 \) and \( E_2 \) denote the set of edges removed in the pruning steps \( \text{MaxDegreeReduction} \) and \( \text{DensityReduction} \), respectively. Then, \( c(E_1) \) and \( c(E_2) \) are both at most \( 2c(E_{OPT}(T)) \).

The proof of the lemma and algorithmic details of sparsification are omitted for brevity. By Lemma 4.1, it follows that the approximation bounds of Lemma 3.1 still hold. However, the pruning steps reduce the number of edges, thereby speeding the implementation of GreedyWalk. We discuss the empirical performance of pruning in Section 8. We show below that pruning also improves the approximation factor marginally from \( O(\log n \log \Delta) \) to \( O(\log n \log T) \) which could be significant when \( n \) is large and \( T \ll \Delta \).

**LEMMA 4.2.** Let \( E' \) denote the set of edges found by Algorithm GreedyWalkSparse. Given any constant \( \epsilon > 0 \), let \( k \) be an even integer larger than \( \log(1+\epsilon/3)/\log n \). Then, we have \( \lambda_1(G[E \setminus E']) \leq (1 + \epsilon) T \), and \( c(E') = O(c(E_{OPT}(T)) \log n \log T) \).

The proof is omitted for brevity.

5 **PRIMALDUAL**: \( O(\log n) \)-approximation

**Main idea:** The approach of [12] gives an \( f \)-approximation for the partial covering problem, where \( f \) denotes the maximum number of sets that contain any element in the set system. As in the proof of Lemma 3.1, in our reduction from the SRME problem to partial covering, elements correspond to all the closed walks of length \( k = O(\log n) \), while sets correspond to edges; for an edge \( e \), the corresponding set \( S_e \) consists of all the walks \( w \) that are hit by \( e \). In this reduction, each walk \( w \) lies in \( k \) sets; therefore, \( f = O(\log n) \) for this set system. Therefore, the approach of [12] could improve the approximation factor. Unfortunately, our set system has size \( n^{O(\log n)} \), so that the algorithm of [12] cannot be used directly to get a polynomial time algorithm.

**Algorithm 2** PRIMALDUAL(\( T', S', e', \sigma' \))

Output: Edge set \( E'' \)

1: Initialize \( z_\varnothing = 0 \) for all \( S_\varnothing \in S' \), \( C \leftarrow \varnothing \).
2: \( / / u(w) = 0 \) for all walks \( w \) in \( G' \).
3: while \( C \) is not \( \sigma' \)-feasible do
4: \( x = \min_{e \in E \setminus E'} (c(e)/z_e); \) let \( e \) be an edge for which the minimum is reached.
5: \( C \leftarrow C \cup \{ S_e \} \)
6: For each \( e' \in E \setminus E' \): \( z_{e'} = z_{e'} + x \cdot \text{walks}(e', G', k) \)
7: \( / / u(w) = u(w) + x \) for all walks \( w \) in \( G' \) that pass through \( e' \)
8: \( E'' \leftarrow E'' \cup \{ e \} \)
9: end while

The algorithm of Gandhi et al. [12] uses a primal-dual approach, which maintains dual variables \( u(w) \) for each element (i.e., walk); these are increased gradually, and a set (i.e., an edge) is picked if the sum of duals corresponding to the elements in the set equals its cost. We now discuss how to adapt this algorithm to run in polynomial time, and only focus on polynomial time implementation of the PRIMALDUAL subroutine of [12] in detail here. However, we also present the set cover algorithm HITWALKS for completeness. This algorithm iterates over all edges and invokes PRIMALDUAL in each iteration to obtain a candidate set of edges to remove.
and finally chooses the set with minimum cost. \( T', S', c' \) and \( \sigma' \) denote the set of elements (walks) to be covered, the sets (corresponding to edges that can be chosen), the costs corresponding to the sets/edges and the number of elements (walks) that need to be covered, respectively. A subset \( C \subseteq S' \) is \( \sigma' \)-feasible if \( |\cup_{S_i \in C} S_i| \geq \sigma' \). Let \( u(w) \) denote the dual variables corresponding to the walks \( w \); these are not maintained in the algorithm explicitly, but assigned in the comments, for use in the analysis. This algorithm does not explicitly update the dual variables, but the edges are picked in the same sequence as in [12].

**Lemma 5.1.** Given any constant \( \epsilon > 0 \), let \( k \) be an even integer larger than \( \frac{\log n}{\log(1+\epsilon/3)} \). The dual variables \( u(w) \) in algorithm *PrimalDual* are maintained and updated as in [12], and the edge \( e \) picked in each iteration is the same. We have \( c(E') = O(c(E_{OPT}) \log n) \) and \( \lambda_1(G[E \setminus E']) \leq (1+\epsilon)T \).

The proof is omitted for brevity.

### 6 Node Version

Our discussion so far has focused on the SRME problem. We now consider The SRMN problem. Recall the definition of walks \((v, G, k)\) from Section 2. Let \( G[V''] \) denote the subgraph of \( G = (V, E) \) induced by subset \( V'' \subset V \). We modify Algorithm *GreedyWalk* to work for the SRMN problem as shown in Algorithm **GreedyWalkSRMN**.

It can be shown on the same lines as Lemma 3.1 that this gives a solution of cost \( O(c(E_{OPT}(T)) \log n \log \Delta) \), where \( c(E_{OPT}(T)) \) denotes the cost of the optimal solution to SRMN problem. Further, the same running time bounds as in Sections 3.1 and 3.2 hold.

#### 7 Popular heuristics and lower bounds

A number of heuristics have been developed for controlling the spread of epidemics; these are discussed below. All these heuristics involve ordering the edges based on some kind of score, and then selecting the top few edges based on this score. We describe the score function in each heuristic.

1. **ProductDegree** ([27]): The score for edge \( e = (u, v) \) is defined as \( \text{deg}(u) \times \text{deg}(v) \). Edges are removed in non-increasing order of this score.
2. **EigenScore** ([27, 37]): Let \( x \) be the eigenvector corresponding to the first eigenvalue of the graph. The score for edge \( e = (u, v) \) is \( |x(u) \times x(v)| \).
3. **LinePagerank**: This method uses the linegraph \( L(G) = (E, F) \) of graph \( G = (V, E) \), where \( (e, e') \in F \) if \( e, e' \in E \) have a common endpoint. We define the score of edge \( e \in E \) as the pagerank of the corresponding node in \( L(G) \).

As we find in Section 8.2, these heuristics work well for different kinds of networks. We design another heuristic, **Hybrid**, which picks the best of the EigenScore and ProductDegree methods. The edges are ordered in the following manner: (1) Let \( \pi_1, \ldots, \pi_m \) and \( \mu_1, \ldots, \mu_m \) be orderings of edges in the EigenScore and ProductDegree algorithms, respectively. (2) Initialize \( i = 0 \) and \( j = 0 \), and (3) from the edges \( \pi(i) \) and \( \rho(j) \), remove the one which decreases the max eigenvalue of the residual graph more. Increment the corresponding index.

We have examined the worst case performance of these heuristics. Two of these, namely, EigenScore and ProductDegree, have been used specifically for reducing the spectral radius, e.g., [27, 37]. No formal analysis is known for any of these heuristics in the context of the SRME or SRMN problems; some of them seem to work pretty well on real world networks. We show that the worst case performance of these heuristics can be quite poor, in general.

**Theorem 7.1.** Given any sufficiently large positive integer \( n \), there exists a threshold \( T' < a\sqrt{n} \), for
some constant $a < 1$ and a graph of size $n$ for which the number of edges removed by PRODUCTDEGREE, EIGENSCORE, HYBRID and LINEPAGERANK is $\Omega(\sqrt{n})c(E_{OPT})$.

We give here a proof sketch for theorem 7.1. The detailed proof is omitted for brevity. We construct a graph $G$ for which the statement holds. For convenience let us assume that $T'$ is a positive integer. $G$ contains (1) a clique $G_1$ on $T' + 1$ nodes; (2) a caterpillar tree $G_2$, which comprises of a path $v_1v_2 \cdots v_{q-1}$ with $v_i$ adjacent to $T'$ leaves each and (3) $G_3$, a star graph with $(T' + 1)^2$ leaves and central vertex denoted by $v_q$. We connect $G_1$ to $G_2$ by $(v_0, v_1)$ where, $v_0$ is some node in $G_1$ and $G_2$ is connected to $G_3$ by the edge $(v_q, v_{q-1})$. Note that $q = \frac{n-(T'+1)^2-T''}{T'}$ and $\lambda_1(G) \geq \lambda_1(G_3) = T' + 1$. Again, here we assume that $q$ is an integer.

It can be shown that $c(E_{OPT}) \leq 2T' + 3$. Removing the edges $(v_0, v_1)$ and $(v_{q-1}, v_q)$ isolates the components $G_1$, $G_2$ and $G_3$. $G_1$ is a clique on $T' + 1$ nodes and on removing one edge, its spectral radius decreases below $T'$. $G_2$ is a star with $(T' + 1)^2$ leaves and therefore, on removing at most $(T' + 1)^2 - (T'' + 1)$ edges, its spectral radius decreases below $T'$. It can be shown that $\lambda_1(G_2) \leq \sqrt{T'} + 2$.

It can be demonstrated that all the four algorithms score the edges $(v_i, v_{i+1})$, $i = 0, \ldots, q - 2$ above any edge belonging to the clique $G_1$. However, the spectral radius cannot be brought down below $T'$ until at least one edge in $G_1$ is removed. Therefore, at least $q$ edges will be removed by all the algorithms. By the initial assumption that $T' < c\sqrt{n}$, it follows that $q = \Omega(\sqrt{n})$, while, $c(E_{OPT}) = O(T')$, hence the theorem holds.

8 Experiments

8.1 Methods and Dataset We evaluate the algorithms developed in the paper\(^2\) – GREEDYWALK, GREEDYWALKSPARSE and PRIMALDUAL – and compare their performance with the heuristics from literature – EIGENSCORE, PRODUCTDEGREE, LINEPAGERANK and HYBRID (described in Section 7), as a more sophisticated baseline. The networks which we have considered in our empirical analysis are listed in Table 2 spanning infrastructure networks, social networks and random graphs.

8.2 Experimental results

Performance of our algorithms and comparison with other heuristics: We first compare the quality of solutions from our algorithms with the EIGENSCORE, PRODUCTDEGREE, LINEPAGERANK and HYBRID heuristics in Figure 1. We note that GREEDYWALK is consistently better than all other heuristics, especially as the target threshold becomes smaller.

\(^2\)All code at: http://tinyurl.com/l3lgsq7.
HitWalks evaluate it for one iteration of Algorithm \( \text{PrimalDual} \). It has a much higher running time. Therefore, we only consider the \( \text{GreedyWalk} \) heuristic, compared to \( \text{PrimalDual} \). Figure 2 shows that \( \text{PrimalDual} \) is quite close to \( \text{GreedyWalk} \) after just one iteration; we expect running this algorithm fully would further improve the performance, but additional work is needed to improve the running time.

<table>
<thead>
<tr>
<th>Network</th>
<th>nodes</th>
<th>edges</th>
<th>( \lambda_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barabasi-Albert</td>
<td>1000</td>
<td>1996</td>
<td>11.1</td>
</tr>
<tr>
<td>Erdos-Renyi</td>
<td>994</td>
<td>2526</td>
<td>6.38</td>
</tr>
<tr>
<td>P2P (Gnutella05)</td>
<td>8846</td>
<td>31839</td>
<td>23.53</td>
</tr>
<tr>
<td>P2P (Gnutella06)</td>
<td>8717</td>
<td>31525</td>
<td>22.38</td>
</tr>
<tr>
<td>Collab. Net (HepTh)</td>
<td>9677</td>
<td>25998</td>
<td>31.03</td>
</tr>
<tr>
<td>Collab. Net (GrQc)</td>
<td>5242</td>
<td>14496</td>
<td>45.62</td>
</tr>
<tr>
<td>AS (Oregon 1)</td>
<td>10670</td>
<td>22002</td>
<td>58.72</td>
</tr>
<tr>
<td>AS (Oregon 2)</td>
<td>10900</td>
<td>31180</td>
<td>70.74</td>
</tr>
<tr>
<td>Brightkite Net</td>
<td>58228</td>
<td>214078</td>
<td>101.49</td>
</tr>
<tr>
<td>Youtube Network</td>
<td>1134890</td>
<td>2987624</td>
<td>210.4</td>
</tr>
<tr>
<td>Stanford Web graph</td>
<td>281903</td>
<td>1992636</td>
<td>448.13</td>
</tr>
</tbody>
</table>

Table 2: Networks and their sizes. The first two are synthetic random networks; others are taken from [36] and [10].

Compared to the EigenScore, ProductDegree and LinePageRank heuristics, the spectral radius for the solution produced by \( \text{GreedyWalk} \), as a function of the fraction of edges removed, is lower by at least 10-20\%. Our improved baseline, the Hybrid heuristic, works better than the other heuristics, and comes somewhat close the \( \text{GreedyWalk} \) in many networks.

Though \( \text{PrimalDual} \) gives a significantly better approximation guarantee, compared to \( \text{GreedyWalk} \), it has a much higher running time. Therefore, we only evaluate it for one iteration of Algorithm HitWalks. Figure 2 shows that \( \text{PrimalDual} \) is quite close to \( \text{GreedyWalk} \) after just one iteration; we expect running this algorithm fully would further improve the performance, but additional work is needed to improve the running time.

Running time and effect of sparsification: Figure 3 shows the total running time of \( \text{GreedyWalk} \) for three networks. The time decreases with the increase of \( T \), because the while loop in Algorithm \( \text{GreedyWalk} \) needs to be run for fewer iterations. The high running time motivates faster methods. We evaluate the performance of the \( \text{GreedyWalkSparse} \) algorithm. As shown in Figure 4, \( \text{GreedyWalkSparse} \) gives almost the same quality of approximation as \( \text{GreedyWalk} \), but improves the running time by up to an order of magnitude, particularly when \( T \) is small.

SRMN: For the SRMN problem, we compare the adaptation of \( \text{GreedyWalk} \), as discussed in Section 6, with the node versions of the Degree and EigenScore heuristic [37]. As shown in Figure 5, \( \text{GreedyWalk} \) performs consistently better.

Main observations:
1. \( \text{GreedyWalk} \) performs consistently better than existing heuristics in removing nodes or edges.
2. Performance of just one iteration of \( \text{PrimalDual} \) comes very close to \( \text{GreedyWalk} \).
3. Sparsification helps in improving the speed of \( \text{GreedyWalk} \) without affecting the solution quality.

9 Related Work
Related work comes from multiple areas: epidemiology, immunization algorithms and other optimization algorithms. There is general research interest in studying...
dynamic processes on large graphs, (a) blogs and propagations [16, 21], (b) information cascades [14, 15] and (c) marketing and product penetration [33]. These dynamic processes are all closely related to virus propagation.

Epidemiology: A classical text on epidemic models and analysis is by May and Anderson [2]. Most work in epidemiology is focused on homogeneous models [4, 2]. Here we study network based models. Much work has gone into in finding epidemic thresholds (minimum virulence of a virus which results in an epidemic) for a variety of networks [28, 39, 13, 30].

Immunization: There has been much work on finding optimal strategies for vaccine allocation [5, 24, 9]. Cohen et al [11] studied the popular acquaintance immunization policy (pick a random person, and immunize one of its neighbors at random). Using game theory, Aspnes et al. [3] developed inoculation strategies for victims of viruses under random starting points. Kuhlman et al. [20] studied two formulations of the problem of blocking a contagion through edge removals under the model of discrete dynamical systems. As already mentioned Tong et al. [38, 37], Van Mieghem et al. [27], Prakash et al. [29] and Chakrabarti et al. [7] proposed various node-based and edge-based immunization algorithms based on minimizing the largest eigenvalue of the graph. Other non-spectral approaches for immunization have been studied by Budak et al. [6], He et al. [17] and Khalil et al. [19].

Other Optimization Problems: Other diffusion based optimization problems include the influence maximization problem, which was introduced by Dominigos and Richardson [32], and formulated by Kempe et. al. [18] as a combinatorial optimization problem. They proved it is NP-Hard and also gave a simple $1 - 1/e$ approximation based on the submodularity of expected spread of a set of starting seeds. Other such problems where we wish to select a subset of ‘important’ vertices on graphs, include ‘outbreak detection’ [23] and ‘finding most-likely culprits of epidemics’ [22, 31].

10 Conclusions

We study the problem of reducing the spectral radius of a graph to control the spread of epidemics by removing edges (the SRME problem) or nodes (the SRMN problem). We have developed a suite of algorithms for these problems, which give the first rigorous bounds for these problems. Our main algorithm GreedyWalk performs consistently better than all other heuristics for these problems, in all networks we studied. We also develop variants that improve the running time by sparsification, and improve the approximation guarantee using a primal dual approach. These algorithms exploit the connection between the graph spectrum and closed walks in the graph, and perform better than all other heuristics. Improving the running time of these algorithms is a direction for further research. We expect these techniques could potentially help in optimizing other objectives related to spectral properties, e.g., robustness [8], and in other problems related to the design of interventions to control the spread of epidemics.

Acknowledgments. This work has been partially supported by the following grants: DTRA Grant HDTRA1-11-1-0016, DTRA CNIMS Contract HDTRA1-11-D-0016-0010, NSF Career CNS 0845700, NSF ICES CCF-1216000, NSF NETSE Grant CNS-1011769, DOE DE-SC0003957, NSF Grant IIS-1353346 and Maryland Procurement Office contract H98230-14-C0127. Also supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC000337, the US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the US Government.

References