Optimal Intervention on Weighted Networks via Edge Centrality

Dongyue Li Tina Eliassi-Rad Hongyang R. Zhang

Abstract

Suppose there is a spreading process propagating on a weighted graph. Denote the graph's weight matrix as W. How would we reduce the number of nodes affected during the process? This question appears in recent studies about counterfactual outcomes of implementing edge-weight interventions on mobility networks (Chang et al. (2021)). A practical algorithm to reduce infections is by removing edges with the highest edge centrality, defined as the product of two adjacent nodes' eigenscores (Tong et al. (2012)). In this work, we design edge-weight reduction algorithms on static and time-varying weighted networks with theoretical guarantees. First, we prove that edge centrality equals the gradient of the largest eigenvalue of $WW^\top$ (over W) and generalize the gradient for the largest r eigenvalues of $WW^\top$. Second, we design a Frank-Wolfe algorithm for finding the optimal edge-weight reduction to shrink the largest r eigenvalues of $WW^\top$ under any reduction budget. Third, we extend our algorithm to time-varying networks with guaranteed optimality. We perform a detailed empirical study to validate our approach. Our algorithm significantly reduces the number of infections compared with existing methods on eleven weighted networks. Further, we illustrate several properties of our algorithm: the benefit of choosing r, fast convergence to the optimum, and a linear-scale runtime per iteration.

1 Introduction

Suppose there is a spreading process such as an epidemic propagating through a graph. Denote the graph as $G = (V, E)$. How would we reduce the number of affected nodes from V during the spreading process? Many studies have considered this question in the network immunization literature [9, 8], motivated by considerations for controlling the outcome of the diffusion process [21]. A principal approach from the existing literature is to optimize spectral properties of G with edge removal procedures. For example, Tong et al. [27] design algorithms to reduce the largest eigenvalue of G's adjacency matrix by removing a budgeted number of edges. Le et al. [16] further study how to reduce the largest r eigenvalues under a budget constraint of edge removals. In this work, we revisit the spectral optimization approach on weighted graphs. Let W denote a nonnegative weight matrix corresponding to the edge weights of G. We consider edge-weight reduction with a budgeted amount B that will create the most drop in the largest r eigenvalues of $WW^\top$.

As an example, weighted graphs have appeared in recent studies about the pandemic. Chang et al. [5] study the counterfactual outcome of implementing edge-weight reduction strategies in mobility networks. Reducing edge weights in mobility networks corresponds to restricting the mobility of population groups.

An effective algorithm for optimizing the spectral properties of a graph is by examining edges with the highest centrality scores. Let $\lambda_1(W)$ denote the largest singular value of W (notice that the largest eigenvalue of $WW^\top$ is equal to the square of $\lambda_1(W)$). Let $\vec{u}_1$ and $\vec{v}_1$ denote the left and right singular vectors corresponding to $\lambda_1(W)$, respectively. The centrality score of an edge (i, j) is equal to $\vec{u}_1(i) \cdot \vec{v}_1(j)$, which are the i-th and j-th entry of each vector. Tong et al. [27] show that removing edges with the highest edge centrality scores effectively reduces $\lambda_1(W)$. Chen et al. [6] further quantify the approximation ratio of this approach with submodular optimization techniques (see also [24]). These works focus on the case of unweighted graphs, for which the spectral optimization problem is NP-hard [8]. Notice that in the case of weighted graphs, the weight of an edge can be reduced by a fraction. Yu et al. [31] apply gradient-based optimization for targeted diffusion which also applies to weighted graphs, with a stopping criterion until the gradient becomes close to zero.

To motivate our approach, we begin by observing that the edge centrality score is equal to the gradient of the largest singular value, up to a scale of $2\lambda_1(W)$:

$$\frac{\partial\Big(\big(\lambda_1(W)\big)^2\Big)}{\partial W_{i,j}} = 2\lambda_1(W) \cdot \vec{u}_1(i) \cdot \vec{v}_1(j).$$

Notice that the above corresponds to the rank-1 SVD of W. More generally, for any rank r, the gradient of the largest r singular values can be efficiently computed via a rank-r SVD of W. Based on the connection between edge centrality and gradients, we minimize the largest r eigenvalues of $WW^\top$ via the Frank-Wolfe algorithm,

Northeastern University, Boston, MA. Correspondence to All Authors ⟨li.dongyu, t.eliassirad, ho.zhang@northeastern.edu⟩.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

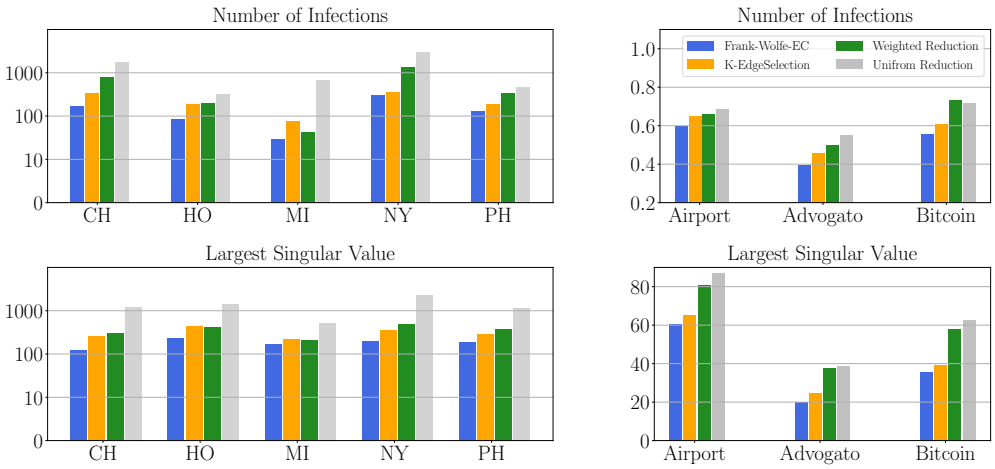


Figure 1: Comparison of our algorithm and several existing approaches; the number of infections ($\times 10^3$) is averaged between fifty simulations. Our approach connects edge centrality with gradients, which leads to an efficient Frank-Wolfe algorithm. The algorithm can be used on static and time-varying weighted graphs.

which involves direction finding and line search. We show an efficient way to find the descent direction by reducing edges with the highest edge centrality (see Theorem 3.1). We then recompute the eigenscores at each iteration, which is also related to the approach of Le et al. [16]. By comparison, our algorithm adapts to weighted graphs and is guaranteed to converge to the global optimum (see Theorem 3.2).

With the connection between edge centrality and gradients, we extend our algorithm to time-varying networks, which are sequences of graphs with evolving weight matrices. We provide the eigenscore for each edge of every graph in the sequence and design an algorithm for optimizing the largest r eigenvalues of the product of all weight matrices in the sequence (cf. Prakash et al. [23, Sec. 4.2]).

We evaluate our algorithms by simulating an epidemic model on eleven weighted graphs. In the static case, our approach achieves on average **25.5**% improvement over baselines during SEIR model simulations. The largest singular value decreases by an average of **25.1**% more than the baselines. See Figure 1 for an illustration. Meanwhile, our approach is also effective for SIR and SIS models. Further, for several time-varying networks, our algorithm reduces the number of infections by over **6.9**%.

The rest of our paper is organized as follows. In Section 2, we formally define the spectral optimization problem on weighted graphs. Then in Section 3, we develop two algorithms for this problem on static and time-varying networks, respectively. We validate our approach with extensive experiments in Section 4. Lastly, we discuss several related literature in Section 5 and questions for future work in Section 6.

2 Preliminaries

Problem setup. Given a spreading process on a network, we are interested in designing algorithms to reduce the number of affected nodes. Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be a weighted and possibly directed graph. Let $\boldsymbol{V}$ be the set of vertices and $\boldsymbol{E}$ be the set of edges. We use W to denote a non-negative weight matrix over the edges, with $W_{i,j}$ being its (i,j)-th entry. Suppose there is an *arbitrary* edge-weight reduction budget B. How should we allocate the budget across the edges?

To answer this question, we consider an eigenvalue optimization approach that has been the basis of prior works for unweighted graphs [3, 22, 27, 9]. The idea behind eigenvalue optimization approaches is to modify the weight matrix W so that its largest eigenvalue is most reduced. We extend the eigenvalue minimization approach to weighted networks as follows. Let M be an n by n matrix, where n is the number of nodes in $\boldsymbol{V}$. Given a rank r, let $\lambda_k(M)$ be the k-th largest singular value of M. We consider the following problem:

$$
\begin{aligned}
(2.1) \qquad \min_{M} \quad & f(M) = \sum_{k=1}^{r} \big(\lambda_k(M)\big)^2 \\
\text{s.t.} \quad & \sum_{(i,j)\in \boldsymbol{E}} \big(W_{i,j} - M_{i,j}\big) \le B \\
& 0 \le M_{i,j} \le W_{i,j}, \, \forall\, (i,j) \in \boldsymbol{E}, \\
& M_{i,j} = 0, \qquad\qquad \forall\, (i,j) \notin \boldsymbol{E}.
\end{aligned}
$$

After solving the above problem, we get a reduced weight matrix M as the solution of our intervention strategy. Notice that we approach this problem from an optimization perspective. Questions including interpreting the solution would be interesting questions for future work. As a remark, the square of $\lambda_k(M)$ is equal to the k-th largest eigenvalue of $MM^\top$. Thus,

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

the objective in equation (2.1) includes the top-r eigenvalues (see also Le et al. [16]). The reason is because the other top eigenvalues could still affect the spreading process in subgraphs of G [1, 12, 16, 31]. In Figure 2, we first illustrate that reducing the largest singular value of G indeed reduces the number of infections during simulated spreading processes. In Section 4.4, we further demonstrate that having the freedom to choose the rank r helps reduce the number of infections.

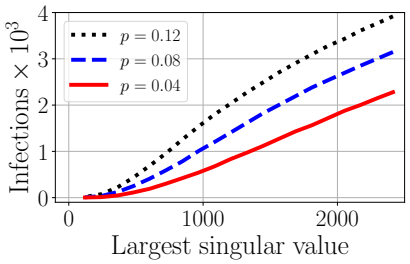


Figure 2: The number of infections strongly correlates with the largest singular value of the graph: more infections are observed for higher values of $\lambda_1(W)$ (by rescaling W). The spreading rate is denoted as p.

Example. To give an example of weighted graphs in epidemic spreading, we can consider mobility networks, which describe the movements from groups of individuals to locations. The graph is weighted by the number of movement records. For instance, Chang et al. [5] introduce a mobility-based modeling approach to fit the observed number of infections. Their approach involves fitting a metapopulation SEIR model with publicly available mobility records. Recall that an SEIR model uses four compartments to capture a spreading process: Susceptible (S), Exposed (E), Infected (I), and Recovered (R). Notice that in their case, the mobility network is bipartite: one side being points of interest (POIs) and the other side being census block groups (CBGs). One way to convert the weighted bipartite network to our problem setup is by joining the traffic across all POIs for every pair of CBGs via a matrix multiplication.

3 Spectral Optimization with Frank-Wolfe

We present a new algorithm to optimize problem (2.1) efficiently. We start by observing that the gradient of $f(M)$ is equivalent to the edge centrality scores. Then, we develop an iterative algorithm with an efficient inner loop that reduces edges with the highest edge centrality. Lastly, we extend our algorithm to time-varying networks.

3.1 Edge centrality as gradient

To motivate our approach, we begin by reviewing the approach of Tong et al. [27], which introduces edge centrality to reduce $f(W)$ for the case of $r = 1$. The edge centrality score is defined as the product of the eigenvector scores from both ends of an edge. Let X be any matrix. Let $\vec{u}_1$ and $\vec{v}_1$ be the left and right singular vector of X, corresponding to $\lambda_1(X)$. Then, for any edge $(i, j) \in \boldsymbol{E}$, its edge centrality score is given by $\vec{u}_1(i) \cdot \vec{v}_1(j)$, where $\vec{u}_1(i)$ denotes the i-th coordinate of $\vec{u}_1$ and $\vec{v}_1(j)$ denotes the j-th coordinate of $\vec{v}_1$.

Edge-weight reduction can be viewed as a continuous relaxation of edge removal since the weight of an edge can be reduced by a fraction. Interestingly, we show that the edge centrality scores are equal to the gradient of the largest eigenvalue of XX^T with respect to the edge weights up to scaling. As a result, we generalize edge centrality scores as the gradient of the largest r singular values of X.

LEMMA 3.1. *Assume that the singular values of X are all distinct. Then, for any $1 \le i, j \le n$, the partial derivative of $(\lambda_1(X))^2$ with respect to $X_{i,j}$ satisfies*

$$\frac{\partial\big((\lambda_1(X))^2\big)}{\partial X_{i,j}} = 2\lambda_1(X) \cdot \vec{u}_1(i) \cdot \vec{v}_1(j). \tag{3.2}$$

More generally, for any $r = 1, 2, \ldots, n$, we have

$$\frac{\partial\big(\sum_{k=1}^{r}(\lambda_k(X))^2\big)}{\partial X_{i,j}} = 2\sum_{k=1}^{r} \lambda_k(X) \cdot \vec{u}_k(i) \cdot \vec{v}_k(j). \tag{3.3}$$

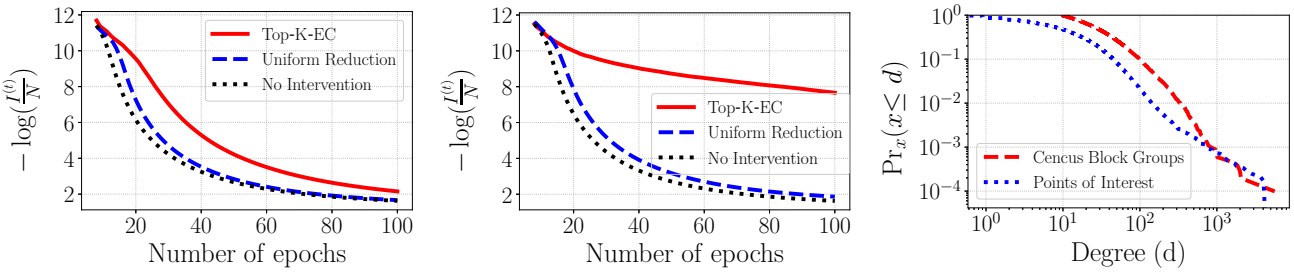
Above, $\vec{u}_k$ and $\vec{v}_k$ are the left and right singular vectors of X corresponding to $\lambda_k(X)$, and the indices correspond to entries of the vectors. The proof of Lemma 3.1 is presented in Appendix A. Given a weight matrix W of a network, we compute the edge centrality scores via the best rank-r approximation of W as $\tilde{W}_r$. Let $\big(\tilde{W}_r\big)_{i,j}$ be the edge centrality score of edge $(i, j) \in \boldsymbol{E}$. We validate that removing edges via top edge centrality scores effectively reduces infections. Figure 3 shows the benefit compared with uniform reduction.

3.2 Global optimization via iterative greedy

We now develop the Frank-Wolfe edge centrality minimization algorithm, or Frank-Wolfe-EC, specified in Algorithm 1. The high-level idea is iteratively applying a greedy selection of edges with the highest edge centrality scores while recomputing the scores:

- **Input:** The primary inputs are graph $\boldsymbol{G}$ with weight matrix W, an arbitrary budgeted reduction amount B, and an arbitrary rank $r \le n$.
- **Output:** An n by n weight matrix M with reduced edge weights from W.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited



(a) Budget is 1% of total edge weights. (b) Budget is 20% of total edge weights. (c) CCDF of nodes.

Figure 3: Comparison of greedy selection and uniform edge-weight reduction on a mobility network. Top-K-EC is more effective in reducing the infected proportion throughout the SEIR model simulation. Moreover, the groups and the points of interest in the graph both follow heavy-tailed degree distributions, supporting our selection using edge centrality scores.

Derivation of the algorithm: At every iteration t from 1 to T, let M_t be the currently modified weight matrix. Let $\nabla f(M_t)$ be the gradient of $f(M_t)$. The Frank-Wolfe algorithm [10, 19] computes a descent direction of M_t by minimizing the correlation between the gradient and the iterate subject to the same constraints as problem (2.1):

$$
\begin{aligned}
(3.4)\quad G_t^\star \leftarrow \arg\min_{X} \quad & \langle X, \nabla f(M_t)\rangle = \mathrm{Tr}\left[\nabla f(M_t)^\top X\right] \\
\text{s.t.} \quad & \sum_{(i,j)\in \boldsymbol{E}} \Big(W_{i,j} - X_{i,j}\Big) \le B \\
& 0 \le X_{i,j} \le W_{i,j},\ \forall (i,j) \in \boldsymbol{E}, \\
& X_{i,j} = 0, \qquad \forall (i,j) \notin \boldsymbol{E}.
\end{aligned}
$$

The core of our approach is to prove that the optimal descent direction for problem (3.4) is essentially by removing edges via top edge centrality scores. Let X be the best rank-r approximation of M_t. Let $(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)$ be the edges in descending order of their edge centrality scores, where m is the number of edges in the graph. Consider the first k edges whose total weight exceeds the reduction budget B. Then, the weight of the first $k-1$ edges is reduced to zero. The weight of the last edge decreases with the remaining budget.

Let us call this procedure Top-K-EdgeCentrality (cf. Alg. 1). The following result proves that this greedy procedure yields an optimal solution to problem (3.4)!

THEOREM 3.1. *The optimal solution $G_t^\star$ (cf. 3.4) is equal to the output of Top-K-EdgeCentrality$(W, B; M_t)$.*

The proof can be found in Appendix A. After finding the descent direction $G_t^\star$, the next step of the Frank-Wolfe algorithm is setting a learning rate η_t in a range between 0 and 1. This follows standard procedures from the Frank-Wolfe algorithm [19]. See Algorithm 1 for the complete pseudo code.

Running time analysis: Next, we examine the number of iterations needed for Alg. 1 to converge to the global optimum of problem (2.1). A well-established result is that for convex objectives, the Frank-Wolfe algorithm will converge to the global minimum under mild conditions [19]. Note that objective (2.1) is indeed convex. Therefore, our algorithm will provably converge to the global minimum of problem (2.1), denoted as f^{OPT}.

THEOREM 3.2. *Let κ be the minimum of $\lambda_r(M_t) - \lambda_{r+1}(M_t)$ over $t = 0, 1, \ldots, T-1$. Assume that κ is strictly positive. Then, the following holds for M_T:*

$$
(3.5)\qquad f(M_T) - f^{\mathrm{OPT}} \le \frac{40\Big(\sum_{(i,j)\in \boldsymbol{E}} W_{i,j}^2\Big)\alpha_2}{T},
$$

where $\alpha_2 = \kappa^{-1} r^{1/2}\big(\max_{t=1}^{T} \lambda_1(M_t)\big) + r + C$, for a fixed value $C > 0$.

The convergence rate of $O(T^{-1})$ in statement (3.5) is obtained following recent literature (e.g., [14]). This result guarantees that our algorithm will converge to the global minimum solution under mild conditions. See Appendix A for the proof. The constants inherited from the previous guarantee in statement (3.5) can be quite large. However, in our experiments, we observe that less than 30 iterations are sufficient for the algorithm to converge (at the global optimum).

To recap, the running time of our algorithm is T times the running time of each iteration, including:

- Computing a truncated rank-r SVD of a sparse matrix with m nonzeros; this requires a time complexity of $O(mr\log(m))$ [18].
- Sorting an array of size m; this requires $O(m\log(m))$ time complexity.

By comparison, running a linear program solver for problem (3.4) requires at least $O(mn)$ time complexity [19]. Thus, our approach is most efficient for small r.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Algorithm 1 Frank-Wolfe for Static Networks

Input: A graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ with weight matrix W; Budget B.
Parameters: Rank r; Iterations T; Range of learning rate H.
Output: A weight matrix M modified from W.

procedure FRANK-WOLFE-EDGECENTRALITY($W, B; T, H$)
  Let $M_0 = W$
  for $t = 0, 1, \ldots, T-1$ **do**
    $G_t^\star$ = TOP-K-EDGECENTRALITY($W, B; M_t$)
    Set η_t by minimizing $f\big((1-\eta_t)M_t + \eta_t G_t^\star\big)$ for $\eta_t \in H$
    $M_{t+1} = (1-\eta_t)M_t + \eta_t G_t^\star$
  end for
  if there is unused budget in M_T **then**
    $B' = B - \mathrm{sum}(W - M_T)$
    $M^\star$ = TOP-K-EDGECENTRALITY($M_T, B'; M_T$)
  end if
  return $M^\star$
end procedure

procedure TOP-K-EDGECENTRALITY($W, B; M$)
  Let $\tilde{M}_r$ be the rank-r SVD of M
  Sort the edges in $\boldsymbol{E}$ by their edge centrality scores from $\tilde{M}_r$; let k be the first value such that the total top-k edge weights in W exceed B
  Reduce the first $k-1$ edges' weight to zero and the last edge's weight by the remaining budget
  return the updated W
end procedure

Algorithm 2 Frank-Wolfe for Time-Varying Networks

Input: A sequence of graphs with weight matrix $\mathcal{W}$ in s steps.
Parameters: Same as the static case.
Output: A sequence of matrices $\mathcal{M}$ modified from $\mathcal{W}$.

procedure FRANK-WOLFE-TIMEVARYING($\mathcal{W}, B; T, H$)
  Let $\mathcal{M}_0 = \mathcal{W}$
  for $t = 0, 1, \ldots, T-1$ **do**
    $\mathcal{G}_t = \{G_t^{\star(i)}\}_{i=1}^s$ = TOP-K-TIMEVARYING($\mathcal{W}, B; \mathcal{M}_t$)
    Set η_t by minimizing $f\big((1-\eta_k)\mathcal{M}_t + \eta_t \boldsymbol{G}_k\big)$ for $\eta_k \in H$
    $\mathcal{M}_{t+1} = \{M_{t+1}^{(i)} = (1-\eta_t)M_t^{(i)} + \eta_t G_t^{\star(i)} : 1 \le i \le s\}$
  end for
  if there is unused budget in $\mathcal{M}_T$ **then**
    $B' = B - \sum_{i=1}^s \mathrm{sum}(W^{(i)} - M_T^{(i)})$
    $\mathcal{M}^\star$ = TOP-K-TIMEVARYING($\mathcal{M}_T, B'; \mathcal{M}_T$)
  end if
  return $\mathcal{M}^\star$
end procedure

procedure TOP-K-TIMEVARYING($\mathcal{W}, B; \mathcal{M}$)
  Let $\tilde{X}_r$ be the rank-r SVD of $X = \prod_{i=1}^s M^{(i)}$
  Sort the edges in the *union* of $\boldsymbol{E}^{(1)}, \boldsymbol{E}^{(2)}, \ldots, \boldsymbol{E}^{(s)}$ by their edge centrality scores (cf. Eq. 3.8); let k be the first value such that the total top-k edge weights from $\mathcal{W}$ exceed B
  Reduce the first $k-1$ edges' weight to zero and the last edge's weight by the remaining budget
  return the updated $\mathcal{W}$
end procedure

3.3 Optimization on time-varying networks

Our study has focused on mitigating the spread in a static network. Another consideration is that the network topology evolves over time. Therefore, an important question is how to tackle such temporal evolution. Next, we show how to extend our optimization algorithm to time-varying networks.

Derivation of the algorithm: Let the weight matrices of a sequence of graphs be denoted as $\mathcal{W} = \{W^{(1)}, W^{(2)}, \ldots, W^{(s)}\}$. Motivated by the work of Prakash et al. [23] which shows the epidemic threshold of time-varying networks, we extend the eigenvalue minimization problem on time-varying networks. Let $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(s)}\}$ be a sequence of modified weight matrices. We aim to find $\mathcal{M}$ that shrinks the largest eigenvalues of a product matrix:

$$
\begin{aligned}
(3.6)\quad \min_{\mathcal{M}} \quad & f(\mathcal{M}) = \sum_{k=1}^{r} \Big(\lambda_k\Big(\prod_{t=1}^{s} M^{(t)}\Big)\Big)^2 \\
\text{s.t.} \quad & \sum_{t=1}^{s} \sum_{(i,j)\in \boldsymbol{E}^{(t)}} \big(W_{i,j}^{(t)} - M_{i,j}^{(t)}\big) \le B \\
& 0 \le M_{i,j}^{(t)} \le W_{i,j}^{(t)}, \ \forall (i,j) \in \boldsymbol{E}^{(t)}, t = 1, \ldots, s, \\
& M_{i,j}^{(t)} = 0, \qquad \forall (i,j) \notin \boldsymbol{E}^{(t)}, t = 1, \ldots, s.
\end{aligned}
$$

Above, $\boldsymbol{E}^{(t)}$ represents the set of edges in the t-th graph of the sequence. Based on Prakash et al. [23, Theorem 2], the weight matrix that determines the epidemic threshold process in time-varying networks is the joint product of each weight matrix in the sequence: $X = \prod_{t=1}^{s} M^{(t)}$. This is why we minimize the largest eigenvalues of the product matrix in $f(\mathcal{M})$.

Following Lemma 3.1, we derive the gradient of the largest r eigenvalues of $X^\top X$ with respect to $M_{i,j}^{(t)}$, for any $1 \le i, j \le n$. By the chain rule, we have:

$$
(3.7)\qquad \frac{\partial f(\mathcal{M})}{\partial M_{i,j}^{(t)}} = \Big\langle \frac{\partial\Big(\sum_{k=1}^{r} \big(\lambda_k(X)\big)^2\Big)}{\partial X}, \frac{\partial X}{\partial M_{i,j}^{(t)}} \Big\rangle.
$$

Notice that the first term above on the right is precisely the edge centrality scores we have derived in Lemma 3.1. The second term is the product of the rest of the weight matrices in $\mathcal{W}$ except that $M^{(t)}$ is replaced by an indicator matrix, which is the derivative of $M^{(t)}$ with respect to its (i,j)-th entry. Let $\tilde{X}_r = U_r D_r V_r^\top$ be the rank-r SVD of X. We get (cf. Appendix A):

$$
(3.8)\qquad \frac{\partial f(\mathcal{M})}{\partial M^{(t)}} = 2\Big(\prod_{k=1}^{t-1} M^{(k)}\Big)^\top \tilde{X}_r \Big(\prod_{k=t+1}^{s} M^{(k)}\Big)^\top.
$$

Matrix (3.8) encodes the edge centrality scores for every edge of $\boldsymbol{E}^{(t)}$, at any step t. Thus, we can develop an algorithm for time-varying networks as the static case. The complete procedure is described in Algorithm 2.

Running time analysis: Similar to Theorem 3.2, one can then prove that Algorithm 2 is guaranteed to converge to the optimum solution of problem (3.6) at the rate of $O(T^{-1})$ after T iterations. The details of this extension can be found in Appendix A.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

4 Experiments

We evaluate our proposed approaches on a range of weighted graphs and mobility networks. Our experiments seek to address the following questions: First, does our approach reduce the infections and the largest singular values well compared to methods from prior works? Second, what are the effects of each component in our approach, e.g., setting the rank r, running iterative greedy selection, and setting the budget? Third, does our approach run efficiently in practice? We present positive results to answer these three questions, validating the practical benefit of our algorithm.

4.1 Experimental setup

Datasets. We use three weighted graphs in our model simulations on static networks: (i) An airport traffic network of flights among commercial airports in the world. (ii) A trust network of users on the Advogato platform; (iii) A trust network of users on a Bitcoin platform. The edge weights in the Airport network denote the number of flight routes between two airports. Edge weights in the last two networks denote different levels of declared trust among users. The edge weights on the Advogato network are between 0 and 1. The edge weights on the Bitcoin network range from -10 to 10. We scale the weights to positive by $\exp(w/5)$. The statistics of the networks are listed in Appendix C.

Besides, we use eight mobility networks constructed with the procedure described in Chang et al. [5]. We generate the mobility networks based on mobility patterns of eight cities. The edge weights denote the amount of population that moves from a group to a location from March 2, 2020, to May 10, 2020. Overall, the mobility patterns cover 25,341 census block groups with over 65 million people and 147,638 points of interest. We report the statistics of the mobility networks in Table 1. We defer a comprehensive discussion of the construction procedure to their paper.

For time-varying networks, we use two sequences of weighted trust networks from Bitcoin-Alpha and Bitcoin-OTC platforms. Each sequence contains ten trust relationship networks corresponding to five periods. The edge weights are processed in the same way as in the static Bitcoin network. We also construct time-varying mobility networks corresponding to ten weeks of the same period above for Chicago and Houston. We describe data sources for the networks in Appendix C.

Baseline methods. The experiments of spreading on static networks involve the following baseline methods: (1) K-EdgeDeletion: Delete a set of edges with the highest edge centrality scores according to the best rank-1 approximation of W [27]. (2) Weighted reduction: Reduce the weight of every edge by a ratio that is proportional to its weight. (3) Uniform reduction: Uniformly reduce the weight of every edge by the same fraction. (4) Max occupancy capping: Reduce the cumulative weights at each POI proportional to its max occupancy. (5) Capping by POI category: Cap the maximum occupancy of a particular category of POIs. The last three baselines are adapted from Chang et al. [5].

For time-varying networks, we consider a similar set of baseline methods, including uniform reduction, weighted reduction, and the K-EdgeDeletion method [27]. The difference from methods on static networks is that edge weight reduction strategies are applied to all edges in the sequence of networks.

Implementation. We simulate an SEIR model on each weighted network. On weighted graphs, a node can get infected by its infectious neighbors with a probability equal to the edge weight times the transmission rate. We use a transmission rate 0.05 and an initial exposed ratio 0.01. On mobility networks, we follow the procedure of Chang et al. [5] to simulate a metapopulation SEIR model in each network where one SEIR model is instantiated for each CBG. We calibrate the parameters of SEIR models so that the simulated cases approximate the reported cases from New York Times COVID-19 data. Besides, we also evaluate our algorithm on other variants of epidemic models, including SIR and SIS with the same parameters. We describe the simulation setup details in Appendix C. For completeness, a brief description of the epidemic models is provided in Appendix B.

In Algorithm 1 and 2, we search the rank parameter r in $[1, 50]$ and the number of iterations in $[5, 30]$. For each result reported in Section 4, we search the two hyper-parameters 50 times. We use an edge-weight reduction budget as 5% of the total edge weights. Results of using other budget amounts are consistent and are discussed in Section 4.3. We use 30 values from the range of $[10^{-3}, 10^{-1}]$ as the range of learning rate H. For weighted graphs, we directly use the weight matrix as W. For mobility networks, we compose the weight matrix W by multiplying the bipartite network matrix and its transpose. All the experiments are conducted on an AMD 24-Core CPU machine.

4.2 Experimental results We show that both of our algorithms are effective in controlling infections by reducing the largest singular value on a range of static and time-varying networks. We observe consistent results across various epidemic models, including SEIR, SIR, and SIS models.

- **Drop in the largest singular value:** Figure 1 illustrates the largest singular value of the modified weight matrix of the three weighted graphs. FRANK-

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Table 1: **Top:** Dataset statistics for eight mobility networks. **Middle:** Comparison of the largest singular value of the edge-weight reduced matrix. **Bottom:** Comparison of the total number of infected populations ($\times 10^3$) in SEIR model simulations. We report the averaged number of infections from fifty independent simulations.

Graphs	AT	CH	DA	HO	MI	NY	PH	DC
Nodes	11,232	32,390	19,069	38,895	17,858	34,216	18,649	10,590
Edges	154,729	439,262	283,928	671,217	276,109	463,719	260,279	107,733
Avg. edge weight	5.258	4.659	4.921	4.951	4.833	4.749	4.864	4.848
Largest singular value	**AT**	**CH**	**DA**	**HO**	**MI**	**NY**	**PH**	**DC**
No Intervention	5526	1296	2093	14677	555	2413	12032	1406
Uniform Reduction	5250	1231	1988	1394	527	2292	1143	1336
Weighted Reduction	1254	302	564	420	213	4818	374	365
Max Capping	5250	1231	1988	1394	527	2292	1143	1336
POI Category	5526	1295	2073	1467	555	2270	1202	1375
K-EdgeDeletion	1565	257	417	447	216	355	282	227
Top-k-EC	1565	257	417	447	216	355	282	226
Ours (Alg. 1)	**1191**	**125**	**308**	**235**	**169**	**197**	**190**	**188**
Infected populations	**AT**	**CH**	**DA**	**HO**	**MI**	**NY**	**PH**	**DC**
No Intervention	48±3	1858±46	91±21	366±26	752±26	3146±21	492±20	41±2
Uniform Reduction	46±2	1762±64	84±11	312±26	671±23	2996±40	463±12	41±1
Weighted Reduction	43±2	782±86	66±3	194±18	43±12	1336±60	342±10	40±1
Max Capping	44±2	1741±65	82±8	315±33	675±26	2990±45	455±15	41±1
POI Category	46±3	1728±62	77±8	283±31	687±25	2950±38	458±17	41±1
K-EdgeDeletion	44±2	346±40	64±2	186±18	78±8	352±27	185±10	39±1
Top-k-EC	45±3	355±46	64±2	187±21	78±7	362±36	178±11	39±1
Ours (Alg. 1)	**40±1**	**166±16**	**62±2**	**86±10**	**8±2**	**301±88**	**129±13**	**39±1**

Wolfe-EC reduces the largest singular value more than baselines by **11.4**% on average. Additionally, Table 1 reports the largest singular value of modified mobility networks. Frank-Wolfe-EC is **30.7**% more effective than the best baseline on average.

- **Reduced number of infections:** Figure 1 compares our algorithm to baseline intervention strategies on three weighted graphs. Overall, our algorithm reduces the number of infected nodes by **10.4**% more than baselines on average. Table 1 compares the total number of infected populations on eight mobility networks. Note that ours outperform other baselines by **30.1**% on average and up to **80.3**%.
- **Results for time-varying networks:** We find that on time-varying networks, Frank-Wolfe-TV also outperforms other baselines. The number of infections is smaller by **6.9**% averaged over both time-varying weighted graphs and mobility networks.
- **Simulation using SIS and SIR:** We show that our approach also helps reduce infections in SIR and SIS epidemic models. We observe that Frank-Wolfe-EC reduces the number of infections by **14.7**% and **10.8**% more on average over the eight static mobility networks, respectively.

4.3 Ablation studies We ablate the parameters in our approach and provide further insights into properties of our algorithm.

- *Benefit of choosing ranks:* Recall that our algorithm requires specifying the rank r–the number of top singular values–in Equation 2.1. We hypothesize that varying the rank r would lead to different intervention results. We ablate the performance of our algorithm by using different r in a range of $[1, 50]$. The results show that the performance of the best choice r outperforms using $r = 1$ by **40.2%** averaged over all networks. This result justifies our formulation of the network intervention problem as an optimization for the sum of largest-r singular values instead of only the largest single value.
- *Benefit of being iterative:* The greedy selection algorithm Top-k-EC can be viewed as a special case of Frank-Wolfe-EC with $T = 1$. Notice that the iterative approach is necessary to get the observed empirical performance. In Table 1, Frank-Wolfe-EC outperforms Top-k-EC by **31.4**% on average, and the largest singular value is reduced by **33.1**% more.
- *Varying budget B:* We have also observed similar results by varying the budget for mobility reduction. We vary the budget from 1% to 20% using the New York mobility network. We find that our algorithm outperforms the baselines consistently using different budgets, similarly for the largest singular value. Interestingly, when the level of budget is small (e.g., 1%), Frank-Wolfe-EC reduces the largest singular value more significantly than baseline methods.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

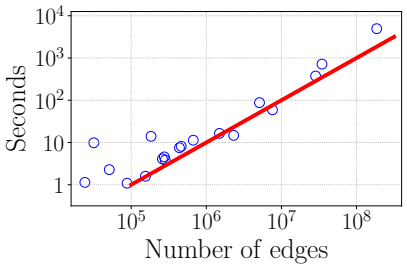


Figure 4: Runtime of Frank-Wolfe-EC in log-log scale for one iteration. The number of edges ranges from 10^4 to 10^8 and the number of nodes ranges from 10^3 to 10^6.

4.4 Runtime report Across all eleven graphs, our approach converges within 30 iterations (or 17 on average). Each iteration requires an SVD step that takes less than 3 seconds. The other steps in each iteration require less than 2.7 seconds. For larger graph instances, we run our method on seven graphs with the number of edges included: com-Orkut (117M), com-LiveJournal (34M), wiki-topcats (28M), web-BerkStan (7.6M), web-Google (5.1M), web-Stanford (2.3M), and web-NotreDame (1.4M) from the SNAP datasets. Figure 4 reports the runtime for one iteration of our algorithm. Notice that the runtime scales nearly-linear with the number of edges. Our algorithm takes 4943 seconds on the largest graph with 117M edges and 3M nodes. These results show that our algorithm runs efficiently on large-scale graphs.

5 Related Work

There is a significant amount of work about diffusion processes on networks. A detailed survey from an epidemic perspective can be found in Pastor-Satorras et al. [21]. A key result in the literature is that the largest eigenvalue of the adjacency matrix (a.k.a. the spectral radius) characterizes the epidemic threshold for many propagation models [30, 11, 22]. An important implication of this result is that the epidemic dies out if the spectral radius decreases, and this has motivated many works on epidemic control [28, 16, 8]. Because eigen-optimization problems via edge additions or deletions are NP-hard [15], both heuristic solutions and principled approximation algorithms have been investigated. A practical approach in the literature is following the greedy algorithm with a centrality notion (see also [20]). There is also a related line of work studying diffusion control in the Firefighter problem [2]. Besides epidemic spreading, diffusion processes are also studied in social networks (e.g., [17, 13]).

Our work applies the Frank-Wolfe algorithm which is a classic algorithm for constrained optimization [10, 19] to study spectral optimization on graphs. The Frank-Wolfe algorithm and its theoretical property are well-studied in the machine learning and optimization literature (see, e.g., Jaggi [14], Tajima et al. [26], and the references therein). We observe a connection between edge centrality and gradients which significantly speeds up the Frank-Wolfe algorithm compared with a naive implementation using a linear program solver. One relevant application for our approach is to consider node-level intervention measures. For mobility networks, reducing the weight of a node means restricting a particular group or location's mobility. Our approach can be naturally extended to node-level reduction by similarly deriving node centrality scores as gradients. Besides, there are also methods for speeding up eigenscore computation on dynamic graphs [7]. It is conceivable that one could combine this method with our approach to achieve the best of both worlds. Finally, there are studies on the design of vaccine distribution for pandemic control [32, 25] and optimization for network robustness [4]. It would be interesting to use the new tools developed in this paper to study these related problems.

6 Conclusion

This work considered controlling diffusion processes on weighted graphs. We study minimizing the largest eigenvalues of the graph and design an efficient algorithm that is guaranteed to converge to the global minimum. We observe a connection between edge centrality scores and gradients, which provides a new way to derive spectral optimization algorithms on graphs. We show how to derive them for static and time-varying networks. Experiments show that our algorithms are effective on various epidemic models and weighted graphs.

We mention two open questions for future work. First, it would be interesting to better understand the metapopulation SEIR model of Chang et al. [5] such as its epidemic threshold. Second, it would be interesting to better understand how eigenvalues affect the diffusion process besides λ_1. We hope our work inspires further algorithmic and theoretical studies about epidemics.

References

[1] R. Andersen, F. Chung, and K. Lang. "Local graph partitioning using pagerank vectors". In: *FOCS*. 2006.

[2] E. Anshelevich, D. Chakrabarty, A. Hate, and C. Swamy. "Approximation algorithms for the firefighter problem: Cuts over time and submodularity". In: *ISAAC*. Springer, 2009.

[3] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. "Epidemic thresholds in real networks". In: *TISSEC*. 2008.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

[4] H. Chan and L. Akoglu. "Optimizing network robustness by edge rewiring: a general framework". In: *Data Min Knowl Disc* (2016).

[5] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec. "Mobility network models of COVID-19 explain inequities and inform reopening". In: *Nature*. Nature Publishing Group, 2021.

[6] C. Chen, R. Peng, L. Ying, and H. Tong. "Network connectivity optimization: Fundamental limits and effective algorithms". In: *KDD*. 2018.

[7] C. Chen and H. Tong. "On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms". In: *SADM*. 2017.

[8] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. "Eigen-optimization on large graphs by edge manipulation". In: *TKDD*. ACM, 2016.

[9] C. Chen, H. Tong, B. A. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. "Node immunization on large graphs: Theory and algorithms". In: *TKDE*. IEEE, 2015.

[10] M. Frank and P. Wolfe. "An algorithm for quadratic programming". In: *Naval Research Logistics Quarterly*. 1956.

[11] A. Ganesh, L. Massoulié, and D. Towsley. "The effect of network topology on the spread of epidemics". In: *INFOCOM*. IEEE, 2005.

[12] D. F. Gleich and C Seshadhri. "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods". In: *KDD*. 2012.

[13] N. Haghtalab, A. Laszka, A. D Procaccia, Y. Vorobeychik, and X. Koutsoukos. "Monitoring stealthy diffusion". In: *Knowl Inf Syst*. 2017.

[14] M. Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *ICML*. 2013.

[15] E. B. Khalil, B. Dilkina, and L. Song. "Scalable diffusion-aware optimization of network topology". In: *KDD*. 2014.

[16] L. Le, T. Eliassi-Rad, and H. Tong. "MET: A fast algorithm for minimizing propagation in large graphs with small eigen-gaps". In: *ICDM*. 2015.

[17] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos. "Rise and fall patterns of information diffusion: model and implications". In: *KDD*. 2012.

[18] C. Musco and C. Musco. "Randomized block krylov methods for stronger and faster approximate singular value decomposition". In: *NIPS*. 2015.

[19] J. Nocedal and S. Wright. *Numerical optimization*. 2006.

[20] N Parotsidis, E. Pitoura, and P. Tsaparas. "Centrality-aware link recommendations". In: *WSDM*. ACM, 2016.

[21] R. Pastor-Satorras, C. Castellano, P. Mieghem, and A. Vespignani. "Epidemic processes in complex networks". In: *Rev Mod Phys*. 2015.

[22] B. A. Prakash, D. Chakrabarti, N. Valler, M. Faloutsos, and C. Faloutsos. "Threshold conditions for arbitrary cascade models on arbitrary networks". In: *Knowl Inf Syst* (2012).

[23] B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos. "Virus propagation on time-varying networks: Theory and immunization algorithms". In: *ECML PKDD*. 2010.

[24] S. Saha, A. Adiga, B. A. Prakash, and A. K. Vullikanti. "Approximation algorithms for reducing the spectral radius to control epidemic spread". In: *SDM*. 2015.

[25] P. Sambaturu, B. Adhikari, B. A. Prakash, S. Venkatramanan, and A. Vullikanti. "Designing effective and practical interventions to contain epidemics". In: *AAMAS*. Springer, 2020.

[26] K. Tajima, Y. Hirohashi, E. Zara, and T. Kato. "Frank-Wolfe algorithm for learning SVM-type multi-category classifiers". In: *SDM*. 2021.

[27] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. "Gelling, and melting, large graphs by edge manipulation". In: *CIKM*. 2012.

[28] P. Van Mieghem, D. Stevanović, F. Kuipers, C. Li, R. Van De Bovenkamp, D. Liu, and H. Wang. "Decreasing the spectral radius of a graph by link removals". In: *Physical Review E*. APS, 2011.

[29] T. Vu, E. Chunikhina, and R. Raich. "Perturbation expansions and error bounds for the truncated singular value decomposition". In: *Linear Algebra and its Applications*. Elsevier, 2021.

[30] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. "Epidemic spreading in real networks: An eigenvalue viewpoint". In: *SRDS*. IEEE, 2003.

[31] S. Yu, L. Torres, S. Alfeld, T. Eliassi-Rad, and Y. Vorobeychik. "POTION: Optimizing Graph Structure for Targeted Diffusion". In: *SDM*. 2021.

[32] Y. Zhang and B. A. Prakash. "Scalable vaccine distribution in large graphs given uncertain data". In: *CIKM*. 2014.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Acknowledgement

Thanks to Bijaya Adhikari for bringing reference [23] to the authors' attention. We would also like to thank Aditya Prakash for helpful comments and discussions.

Table 2: A table of notations used in our paper for reference.

Symbol	Definition
$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$	Weighted and possibly directed graph
W	A nonnegative weight matrix of $\boldsymbol{G}$
$W_{i,j}$	The (i,j)-th entry of W
$\lambda_k(W)$	The k-th largest singular value of a matrix W
$\vec{u}_k$	The left singular vector of a weight matrix W corresponding to $\lambda_k(W)$
$\vec{v}_k$	The right singular vector of a weight matrix W corresponding to $\lambda_k(W)$
$\vec{v}(i)$	The i-th coordinate of the vector $\vec{v}$
$\tilde{X}_r$	The best rank-r approximation of X
$\mathcal{W}$	A sequence of weight matrices from timestamp 1 to s
$\boldsymbol{E}^{(t)}$	The set of edges in the t-th graph of the sequence
$W^{(t)}$	A nonnegative square weight matrix for the graph at timestamp t
$\|\cdot\|$	The ℓ_2 norm of a vector or the spectral norm of a matrix
$\|\cdot\|_F$	The Frobenius norm of a matrix
$\langle\cdot,\cdot\rangle$	The matrix inner product between two matrices

A Proofs

This section lays out the proofs for our statements in Section 3. First, we prove the connection between edge centrality and gradients.

Proof of Lemma 3.1. Consider a singular value λ_k of X, for any k. Let $\vec{u}_k$ and $\vec{v}_k$ be the left and right singular vectors of X corresponding to λ_k, respectively. By the chain rule, it suffices to show that $\frac{\partial \lambda_k(X)}{\partial X_{i,j}} = \vec{u}_k(i) \cdot \vec{v}_k(j)$. First, we have $\vec{u}_k^\top X = \lambda_k \vec{v}_k^\top$. We differentiate over X on both sides of the above equation:

$$\mathrm{d}(\vec{u}_k^\top)X + \vec{u}_k^\top \mathrm{d}(X) = \mathrm{d}(\lambda_k)\vec{v}_k^\top + \lambda_k \mathrm{d}(\vec{v}_k^\top). \tag{A.1}$$

Since $\vec{v}_k$ is a unit length vector,

$$\mathrm{d}(\|\vec{v}_k\|^2) = 2\langle \vec{v}_k, \mathrm{d}(\vec{v}_k)\rangle = 2\,\mathrm{d}(\vec{v}_k^\top)\vec{v}_k = 0. \tag{A.2}$$

Thus, by multiplying both sides of equation (A.1) with $\vec{v}_k$, we get

$$\mathrm{d}(\vec{u}_k^\top)X\vec{v}_k + \vec{u}_k^\top \mathrm{d}(X)\vec{v}_k = \mathrm{d}(\lambda_k)\vec{v}_k^\top \vec{v}_k + \lambda_k \mathrm{d}(\vec{v}_k^\top)\vec{v}_k, \tag{A.3}$$

which is equal to $\mathrm{d}(\lambda_k)$ since equation (A.2) holds and v_k is a unit length vector. Looking at equation (A.3), we observe

$$\mathrm{d}(\vec{u}_k^\top)X\vec{v}_k = \mathrm{d}(\vec{u}_k^\top)\lambda_k\vec{u}_k = \lambda_k \mathrm{d}(\vec{u}_k^\top)\vec{u}_k = 0, \tag{A.4}$$

where the last step follows similarly to equation (A.2), since $\vec{u}_k$ is also a unit length vector. In summary, we have shown $\vec{u}_k^\top \mathrm{d}(X)\vec{v}_k = \mathrm{d}(\lambda_k)$. This implies that the derivative of λ_k over $X_{i,j}$ is equal to $\vec{u}_k(i) \cdot \vec{v}_k(j)$. Since this holds for any k, we thus conclude that equations (3.2) and (3.3) are both true. □

Second, we prove that greedy selection is optimal for the inner loop of the Frank-Wolfe algorithm.

Proof of Theorem 3.1. By Lemma 3.1, for every edge $(i,j) \in \boldsymbol{E}$, the gradient of $f(M_t)$ over this edge is given by the *edge centrality* scores. Since $X_{i,j} = 0$ for any $(i,j) \notin \boldsymbol{E}$, the optimization objective is:

$$\langle X, \nabla f(M)\rangle = \sum_{(i,j)\in \boldsymbol{E}} 2X_{i,j}\Big(\sum_{k=1}^{r} \lambda_k \cdot \vec{u}_k(i) \cdot \vec{v}_k(j)\Big). \tag{A.5}$$

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Above, each variable $X_{i,j}$ is multiplied precisely by the edge centrality of the edge (i,j) (cf. line (15)). Consider minimizing the equivalent objective (A.5) with the constrains of Problem (3.4). The minimizer, $G_t^\star$, is achieved by reducing the weight of the edges with the highest edge centrality to zero until the budget B gets exhausted. This is precisely the procedure of Top-K-EC from lines (15)-(17). Thus, we have proved this result. □

Third, we derive the convergence guarantee of Algorithm 1.

Proof of Theorem 3.2. We complete the convergence analysis of our algorithm. First, we show that the objective function $f(M)$ is convex in M. Second, we invoke the result of Jaggi [14], specifically Lemma 7 and Theorem 1, which show that as long as the gradient $\nabla f(M)$ is Lipschitz-continuous and the constraint set has bounded diameter, the Frank-Wolfe algorithm will converge to the optimum at a rate of $O(\frac{1}{t})$ after t iterations.

We first show that the sum of top singular values $g(M) = \sum_{k=1}^{r} \lambda_k(M)$ is convex. With the variational characterization of singular values, $g(M)$ is equal to

$$g(M) = \max_{U^\top U = V^\top V = \mathrm{Id}_r:\ U\in\mathbb{R}^{n\times r}, V\in\mathbb{R}^{m\times r}} \langle UV^\top, M\rangle. \tag{A.6}$$

Thus, for any n by m matrix M_1, M_2, and any $\alpha \in [0,1]$, let $\tilde{U}$ and $\tilde{V}$ be the maximizer of the above for $f\big(\alpha M_1 + (1-\alpha)M_2\big)$. Therefore,

$$\begin{aligned} g\big(\alpha M_1 + (1-\alpha)M_2\big) &= \langle \tilde{U}\tilde{V}^\top, \alpha M_1 + (1-\alpha)M_2\rangle \\ &\le \alpha\langle \tilde{U}\tilde{V}^\top, M_1\rangle + (1-\alpha)\langle \tilde{U}\tilde{V}^\top, M_2\rangle \\ &\le \alpha g(M_1) + (1-\alpha) g(M_2), \end{aligned}$$

which implies that $g(M)$ is convex. Next, we show that $f(M)$ is convex. For any $\alpha \in [0,1]$,

$$\begin{aligned} &f(\alpha M_1 + (1-\alpha)M_2) = g\Big((\alpha M_1 + (1-\alpha)M_2)^T(\alpha M_1 + (1-\alpha)M_2)\Big) \\ &\le \alpha^2 g(M_1^\top M_1) + (1-\alpha)^2 g(M_2^\top M_2) + 2\alpha(1-\alpha) g(M_1^\top M_2). \end{aligned}$$

Let $\tilde{U}$ and $\tilde{V}$ be the maximizer of (A.6) for $M_1^\top M_2$. We have

$$\begin{aligned} &2g(M_1^\top M_2) = 2\langle \tilde{U}\tilde{V}^\top, M_1^\top M_2\rangle = 2\langle M_1\tilde{U}, M_2\tilde{V}\rangle \\ \le &\left\|M_1\tilde{U}\right\|_F^2 + \left\|M_2\tilde{V}\right\|_F^2 = \langle M_1^\top M_1, \tilde{U}\tilde{U}^\top\rangle + \langle M_2^\top M_2, \tilde{V}\tilde{V}^\top\rangle \\ \le &g(M_1^\top M_1) + g(M_2^\top M_2). \end{aligned}$$

Therefore, $f(\alpha M_1 + (1-\alpha)M_2)$ is less than $\alpha \cdot g(M_1^\top M_1) = \alpha \cdot f(M_1)$ plus $(1-\alpha)\cdot g(M_2^\top M_2) = (1-\alpha)\cdot f(M_2)$.

Second, we verify that $\nabla f(M)$ is α_2 Lipschitz continuous in the Frobenius norm. The proof is based on matrix perturbation bounds. Let $\tilde{M} = M + E$ be a perturbation of M. Let $M_r = U_r D_r V_r^\top$ be the top-r SVD of M. Let μ_1 be the largest singular value of M. Let $\tilde{M}_r = \tilde{U}_r\tilde{D}_r\tilde{V}_r^\top$ be the top-r SVD of $\tilde{M}$. First, consider $\|E\|_2 \le \kappa/2$. By matrix perturbation bounds on the truncated SVD of a matrix (e.g., Theorem 1 of Vu et al. [29]; the condition is satisfied since κ is the spectral gap between the r-th and $(r+1)$-th largest singular values), we have

$$\|M_r - \tilde{M}_r\|_F^2 \le 2\|E\|_F^2 + \frac{4\lambda_1^2}{\kappa^2}\|E\|_F^2 + C\|E\|_F^2.$$

When $\|E\|_2 \ge \kappa/2$, notice that

$$\begin{aligned} \|M_r - \tilde{M}_r\|_F^2 &= \|U_r D_r V_r^\top - \tilde{U}_r\tilde{D}_r\tilde{V}_r^\top\|_F^2 \\ &\le 2\|D_r\|_F^2 + 2\|\tilde{D}_r\|_F^2 \\ &\le 2r\lambda_1^2 + 2r(\lambda_1 + \|E\|_2)^2, \end{aligned}$$

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

which is at most $2r(3\lambda_1^2 + 2\|E\|_2^2)$. The step above uses the Weyl's Theorem that $\|D_r - \tilde{D}_r\|_2 \leq \|E\|_2$. Taken together, we conclude that $\nabla f(M)$ must be

$$\sqrt{\max\left(2 + \frac{4\lambda_1^2}{\kappa^2} + C, \frac{24r \cdot \lambda_1^2}{\kappa^2} + 4r\right)}$$

Lipschitz-continuous. Lastly, the diameter of the constraint set is at most $\sqrt{\sum_{(i,j)\in \boldsymbol{E}} W_{i,j}^2}$, since for every $(i,j) \in \boldsymbol{E}$, the search space is bounded between 0 and $W_{i,j}$. Taken together, we have proved that: $f(M)$ is convex, $\nabla f(M)$ is α_2 Lipschitz continuous, and the diameter of the constrained space of problem (2.1) is $\sqrt{\alpha_1/8}$. Using Lemma 7 and Theorem 1 of Jaggi [14], the proof is complete. □

Extension to time-varying networks. Notice that the time-varying extension is a special case of the above result. Therefore, the same convergence rate of $O(T^{-1})$ holds for Algorithm 2 towards the global optimum of problem (3.6).

Lastly, we derive the gradient of the largest r eigenvalues of $X^\top X$ where X is the product of the weight matrices in the sequence of time-varying networks (cf. Section 3.3).

Derivation of Equation 3.8. Let $\{M^{(1)}, M^{(2)}, \ldots, M^{(s)}\}$ be a sequence of modified weight matrices and $X = \prod_{t=1}^{s} M^{(t)}$. Following Lemma 3.1, we derive the gradient of the largest r eigenvalues of $X^\top X$ with respect to $M_{i,j}^{(t)}$, for any $1 \leq i, j \leq n$. By the chain rule, we have:

$$\frac{\partial f(\mathcal{M})}{\partial M_{i,j}^{(t)}} = \Big\langle \frac{\partial\Big(\sum_{k=1}^{r} \big(\lambda_k(X)\big)^2\Big)}{\partial X}, \frac{\partial X}{\partial M_{i,j}^{(t)}} \Big\rangle. \tag{A.7}$$

Notice that the first term above on the right is precisely the edge centrality scores we have derived in Lemma 3.1. The second term is essentially the product of the rest of the weight matrices in $\mathcal{W}$ except that $M^{(t)}$ is replaced by an indicator matrix, which is the derivative of $M^{(t)}$ with respect to its (i,j)-th entry.

Let $\tilde{X}_r = U_r D_r V_r^\top$ be the rank-r SVD of X. Let the product of weight matrices from 1 to $t-1$ as $A = \prod_{k=1}^{t-1} M^{(k)}$ and the product of weight matrices from $t+1$ to s as $B = \prod_{k=t+1}^{s} M^{(k)}$. A is equal to identity matrix when $t = 1$, and B is equal to identity matrix when $t = s$. Let $J^{i,j}$ as a single-entry indicator matrix where its (i,j)-th entry is 1 and rest of entries are equal to 0. Then, we can rewrite the gradient as follows:

$$\frac{\partial f(\mathcal{M})}{\partial M_{i,j}^{(t)}} = 2\Big\langle \tilde{X}_r, AJ^{i,j}B \Big\rangle = 2 \sum_{1\leq p,q\leq n} \big(\tilde{X}_r\big)_{p,q} \big(AJ^{i,j}B\big)_{p,q} = 2 \sum_{1\leq p,q\leq n} \big(\tilde{X}_r\big)_{p,q} A_{p,i} B_{j,q} = 2\Big(A^\top \tilde{X}_r B^\top\Big)_{i,j} \tag{A.8}$$

Thus, we get the gradient of $f(\mathcal{M})$ with respect to the weight matrix $M^{(t)}$ as follows:

$$\frac{\partial f(\mathcal{M})}{\partial M^{(t)}} = 2A^\top \tilde{X}_r B^\top = 2\Big(\prod_{k=1}^{t-1} M^{(k)}\Big)^\top \tilde{X}_r \Big(\prod_{k=t+1}^{s} M^{(k)}\Big)^\top. \tag{A.9}$$

The derivation of statement (3.8) is now completed. □

B Epidemic Models

We provide a description of the epidemic models that are considered in our experiments. One widely used model of epidemic spread is the SEIR compartmental model. An SEIR model uses four compartments to capture a spreading process: Susceptible (S), Exposed (E), Infected (I), and Recovered (R). Every node must belong to one of the four states during the process. At every time t,

- $S^{(t)}$ denotes the set of susceptible nodes at time t. A node may get exposed if its incoming neighbors are infectious. The probability depends on the edge weights and the virus transmission rate.
- $E^{(t)}$ denotes the nodes who have been exposed to the virus but are not infectious at time t. In expectation, a node remains exposed for δ_E periods.
- $I^{(t)}$ denotes the nodes who are infectious at time t. Each node remains infectious for δ_I periods in expectation.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

- $R^{(t)}$ denotes the nodes who have recovered at time t.

For weighted graphs, we simulate an SEIR model. At each time t, we calculate the infection probability for node i based on the edge weights and transmission rate β_{Base}:

$$p_i = 1 - \prod_{(i,j)\in E: j\in I^{(t)}} \max\Big(1 - W_{i,j}\beta_{\text{base}}, 0\Big).$$

For mobility networks, we follow the procedure in Chang et al. [5] to simulate the metapopulation SEIR model. At time t, the transitions between the four states (for c_i) are sampled as follows:

$$N^{(t)}_{S_{c_i}\to E_{c_i}} \sim \text{Poisson}\Big(\frac{S^{(t)}_{c_i}}{N_{c_i}}\lambda^{(t)}\Big) + \text{Binomial}\Big(S^{(t)}_{c_i}, \lambda^{(t)}_{c_i}\Big). \tag{B.10}$$

$$N^{(t)}_{E_{c_i}\to I_{c_i}} \sim \text{Binomial}\Big(E^{(t)}_{c_i}, \frac{1}{\delta_E}\Big). \tag{B.11}$$

$$N^{(t)}_{I_{c_i}\to R_{c_i}} \sim \text{Binomial}\Big(I^{(t)}_{c_i}, \frac{1}{\delta_I}\Big). \tag{B.12}$$

where $\lambda^{(t)}$ is the aggregate transmission rate over the points of interest; $\lambda^{(t)}_{c_i}$ is the base transmission rate within c_i; δ_E represents the mean latency period; δ_I is the mean infectious period.

In equation (B.10), $\lambda^{(t)}_{c_i}$ is given by the product of the base transmission rate β_{base} and the proportion of infectious individuals in CGB c_i: $\lambda^{(t)}_{c_i} = \beta_{\text{base}}\frac{I^{(t)}_{c_i}}{N_{c_i}}$. The infection rate across all the POIs is

$$\lambda^{(t)} = \sum_{j=1}^{n} \lambda^{(t)}_{p_j} W^{(t)}_{i,j};\ \lambda^{(t)}_{p_j} = \beta^{(t)}_{p_j}\frac{I^{(t)}_{p_j}}{\sum_{i=1}^{m} W^{(t)}_{i,j}}.$$

where $\lambda^{(t)}_{p_j}$ is the infection rate for POI p_j at time t. $\beta^{(t)}_{p_j}$ is the transmission rate at POI p_j and $I^{(t)}_{p_j}$ is the number of infectious individuals in p_j at time t. The parameters are estimated as follows. (i) $\beta^{(t)}_{p_j}$ is estimated by the physical area of p_j: $\beta^{(t)}_{p_j} = \psi \cdot \cdot d^2_{p_j} \cdot \frac{V^{(t)}_{p_j}}{a_{p_j}}$ in which ψ is a transmission constant; a_{p_j} is the physical area of p_j; $V^{(t)}_{p_j} = \sum_{i=1}^{m} W^{(t)}_{i,j}$ represents the number of visitors to p_j at time t. (ii) $I^{(t)}_{p_j}$ is estimated in proportion to the infectious population from each CBG and their number of visits to p_j: $I^{(t)}_{p_j} = \sum_{k=1}^{m}\frac{I^{(t)}_{c_k}}{N_{c_k}} W^{(t)}_{k,j}$.

There are many variants of the SEIR model (cf. Prakash et al. [22]). We consider SIR and SIS that share similar spreading process as the SEIR model. We describe their differences as follows. The SIR model uses three compartments as the SEIR model except the exposed state. It assumes that there is no latent period of the disease. Nodes are capable of infecting susceptible nodes directly after being infected. The SIS model uses two states (Susceptible and Infectious) in a spreading process. It assumes that recovery does not bring immunity and nodes who have recovered will become susceptible again.

C Experiment Details

Simulation setup. For the weighted graphs, we simulate an SEIR model on each graph. We use a transmission rate $\beta_{\text{Base}} = 0.05$ and a initial exposed ratio $p_0 = 0.01$. To avoid infecting all the graph nodes, we simulate for 50 epochs. We use a slightly higher edge-weight reduction budget as 20% of the total edge weights because the average edge weight in these three graphs is smaller than the mobility networks.

For the experiments concerning mobility networks, we follow the procedures of Chang et al. [5] to simulate a metapopulation SEIR model in each network. We calibrate the parameters of the SEIR model following their method. On static mobility networks, We simulate 100 epochs to be consistent with the simulation of Chang et al. [5]. The results are consistent throughout the simulation. We compare the Frank-Wolfe-EC algorithm with baseline methods using an edge-weight reduction budget as 5% of the total edge weights. The results of using other budget amounts are consistent. We use the same set of parameters for SIR and SIS model simulations.

On time-varying mobility networks experiments, we simulate the metapopulation SEIR model on a sequence of ten networks for 70 epochs or seven epochs for every network. We set the edge-weight reduction budget as 5% of the total edge weights of the sequence.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Table 3: **Left:** Dataset statistics for three weighted graphs. **Right:** Dataset statistics for four time-varying networks. Each time-varying network sequence has ten networks.

	Airport	Advogato	Bitcoin
Nodes	7,977	6,541	3,783
Edges	30,501	51,127	24,186
Avg. edge weight	1.45	0.83	1.46

	Bitcoin-Alpha	Bitcoin-OTC	Chicago	Houston
Nodes	3,783	5,881	32,390	38,895
Edges	24,186	35,591	975,569	1,586,683
Avg. edge weight	1.46	1.51	4.27	4.42

Model validation. We calibrate the following parameters for the metapopulation SEIR model on mobility networks: (i) the transmission constant in POIs, ψ; (ii) the base transmission rate, β_{base}; and (iii) the ratio of initial exposed people, p_0. We use grid search to find the parameters with the smallest root mean square error compared to the reported number of infected cases. We calibrate an SEIR model for every MSA independently. We compare the predicted cases of our simulated SEIR model with the reported cases from New York Times COVID-19 data. The root mean squared error of all the epochs is 295.17 averaged over eight mobility networks. The error is within 3% compared to the overall infected population which is at the scale of 10^4. These results reaffirm the finding of Chang et al. [5].

Data availability. The three weighted graphs are available in the following sources: Airport[1], Adavogato[2], and Bitcoin[3]. The two weighted time-varying graphs are available in the following sources: Bitcoin-Alpha[4] and Bitcoin-OTC[5]. We report the network statistics in Table 3. The mobility network data is freely available to researchers, non-profit organizations, and governments through the SafeGraph COVID-19 Data Consortium.[6] The construction of mobility networks requires the following data sources: (i) Mobility patterns from the Monthly Pattern[7] and Weekly Pattern datasets, [8] (ii) The geometry dataset,[9] and (iii) The Open Census. Dataset[10] The New York Times COVID-19-data is publicly available online.[11] The code for reproducing the experiments is available in the link below.[12]

[1] http://opsahl.co.uk/tnet/datasets/openflights.txt
[2] https://downloads.skewed.de/mirror/konect.cc/files/download.tsv.advogato.tar.bz2
[3] http://snap.stanford.edu/data/soc-sign-bitcoinalpha.html
[4] https://snap.stanford.edu/data/soc-sign-bitcoinalpha.csv.gz
[5] https://snap.stanford.edu/data/soc-sign-bitcoinotc.csv.gz
[6] https://www.safegraph.com/covid-19-data-consortium
[7] https://docs.safegraph.com/docs/monthly-patterns
[8] https://docs.safegraph.com/docs/weekly-patterns
[9] https://docs.safegraph.com/docs/geometry-data
[10] https://docs.safegraph.com/docs/open-census-data
[11] https://github.com/nytimes/covid-19-data
[12] https://github.com/lidongyue12138/Optimization-Edge-Centrality

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited