Gelling, and Melting, Large Graphs by Edge Manipulation

Hanghang Tong
IBM T.J. Watson Research
Hawthorne, NY, USA
htong@us.ibm.com

B. Aditya Prakash
Virginia Tech.
Blacksburg, VA, USA
badityaprakash@gmail.com

Tina Eliassi-Rad
Rutgers University
Piscataway, NJ, USA
eliassi@cs.rutgers.edu

Michalis Faloutsos
Univ. of California Riverside
Riverside, CA, USA
michalis@cs.ucr.edu

Christos Faloutsos
Carnegie Mellon University
Pittsburgh, PA, USA
christos@cs.cmu.edu

ABSTRACT

Controlling the dissemination of an entity (e.g., meme, virus, etc) on a large graph is an interesting problem in many disciplines. Examples include epidemiology, computer security, marketing, etc. So far, previous studies have mostly focused on removing or inoculating *nodes* to achieve the desired outcome.

We shift the problem to the level of edges and ask: which edges should we add or delete in order to speed-up or contain a dissemination? First, we propose effective and scalable algorithms to solve these dissemination problems. Second, we conduct a theoretical study of the two problems and our methods, including the *hardness* of the problem, the *accuracy* and *complexity* of our methods, and the *equivalence* between the different strategies and problems. Third and lastly, we conduct experiments on real topologies of varying sizes to demonstrate the effectiveness and scalability of our approaches.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – Data Mining

General Terms

Algorithm, experimentation

Keywords

edge manipulation, immunization, scalability, graph mining

1. INTRODUCTION

Managing the dissemination of an entity (e.g., meme, virus, etc) on a large graph is a challenging problem with applications in various settings and disciplines. In its generality, the propagating entity can be many different things, such as a meme, a virus, an idea, a new product, etc. The propagation is affected by the topology and the properties of the entity: its 'virality', its speed, its 'stickiness' or the duration of the infection of a node. Our focus here is the *topology*, since we assume that we cannot alter the properties of the propagating entity.

The problem we address is how we can affect the propagation by modifying the *edges* of the graph. In fact, we address two different problems. First, in the ***NetMelt* problem**, we want to contain the dissemination by removing a given number of edges. For example, we can consider the distribution of malware over a social network. Deleting user accounts may not be desirable, but deleting edges ('unfriending' people) may be more acceptable. More specifically, we want to delete a set of k edges from the graph to minimize the infected population. Second, in the ***NetGel* problem**, we want to enable the dissemination by adding a given number of edges. Specifically, we want to add a set of k new edges into the graph to maximize the population that adopt the information. For example, we could extend the social network scenario using the recent 'arab spring' which often used Facebook and Twitter for coordinating events: we may want to maximize the spread of a potential piece of information. Note that an additional, key requirement for both problems is computational efficiency: the solution should scale to large graphs.

Both problems are challenging for slightly different reasons. For the *NetMelt* problem, most of the existing methods operate on the *node-level*, e.g., deleting a subset of the nodes from the graph to minimize the infected population from a propagating virus. In the above social spam example, this means that we might have to shut-down some legitimate user accounts. Can we avoid this by operating on a finer granularity, that is, only deleting a few edges between users to slow down the social spam spreading? For the *NetGel* problem, things are even more challenging because of its high intrinsic time complexity. Let n be the number of the nodes in the graph. There are almost n^2 non-existing edges since many real graphs are very sparse. In other words, even if we only want to add one single new edge into the graph, the solution space is $O(n^2)$. This complexity 'explodes' if we aim to add multiple new edges collectively, where the solution space becomes *exponential*. To date, there does not exist *any* scalable solution for the *NetGel* problem.

The overarching contribution of this paper is the formulation and theoretical study of the dissemination management via edge manip-

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

ulation: how to place a set of edges[1] to achieve the desired outcome. The main contributions of the paper can be summarized as follows:

- *Algorithms.* We propose effective and scalable algorithms to optimize the leading eigenvalue, the key graph parameter that controls the information dissemination processes for both *NetMelt* and *NetGel*, respectively;
- *Proofs and Analysis.* We show the *accuracy* and the *complexity* of our methods; the *hardness* of the problem, and *equivalence* between the different strategies;
- *Experimental Evaluations.* Our evaluations on real large graphs show that our methods are both effective and scalable (see Fig. 1 as an example).

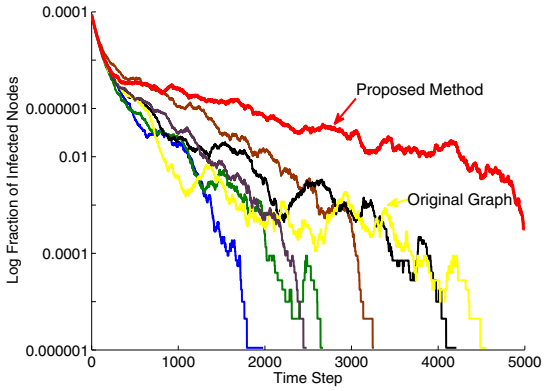


Figure 1: Comparison of maximizing the outcome of the information dissemination process. Larger is better. The proposed method (red) leads to the largest number of 'infected' nodes (e.g., having more people in the social networks to adopt a piece of good idea, etc). Notice that all the alternative methods are mixed with the result on the original graph (yellow), which means that they fail to affect the outcome of the dissemination process. See Section 6 for detailed experimental setting.

The rest of the paper is organized as follows. We introduce notation and formally define the *NetGel* and *NetMelt* problems in Section 2. We present and analyze the proposed algorithms in Section 3 and Section 4, respectively. We provide experimental evaluations in Section 5. We review the related work in Section 6 and conclude in Section 7.

2. PROBLEM DEFINITIONS

Table 1 lists the main symbols used throughout the paper. We consider directed, irreducible unipartite graphs. For ease of presentation, we discuss the unweighted graph scenario although the algorithms we propose can be naturally generalized to the weighted case. We represent a graph by its adjacency matrix. Following the standard notation, we use bold upper-case for matrices (e.g., $\mathbf{A}$), bold lower-case for vectors (e.g., $\mathbf{a}$), and calligraphic fonts for sets (e.g., $\mathcal{I}$). We denote the transpose with a prime (i.e., $\mathbf{A}'$ is the transpose of $\mathbf{A}$). Also, we represent the elements in a matrix using a convention similar to Matlab, e.g., $\mathbf{A}(i,j)$ is the element at the i^{th} row and j^{th} column of the matrix $\mathbf{A}$, and $\mathbf{A}(:,j)$ is the j^{th} column of $\mathbf{A}$, etc.

[1]In this paper, we use the terms 'link' and 'edge' interchangeably.

Table 1: Symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (bold upper case)
$\mathbf{A}(i,j)$	the element at the i^{th} row and the j^{th} column of $\mathbf{A}$
$\mathbf{A}(i,:)$	the i^{th} row of matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	the j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\mathbf{a}, \mathbf{b}, \ldots$	vectors
$\mathcal{I}, \mathcal{J}, \ldots$	sets (calligraphic)
λ	the largest (in module) eigenvalue of $\mathbf{A}$
$\mathbf{u}, \mathbf{v}$	the $n \times 1$ left eigenvector and right eigenvector associated with λ.
n	the number of the nodes in the graph
m	the number of the edges in the graph
k	the budget (i.e., the number of deleted or added edges)

When we discuss the relationship between the two different strategies (node deletion vs. edge deletion) for the *NetMelt* problem, it is helpful to introduce the concept of line graph, where the nodes represent the edges in the original graph. Formally, each edge in the original graph $\mathbf{A}$ becomes a node in the line graph $L(\mathbf{A})$; and there is an edge from one node to the other in the line graph if the target of the former edge is the same as the source of the latter edge in the original graph $\mathbf{A}$. It is formally defined as follows:

DEFINITION 1 (**LINE GRAPH**). *Given a directed graph $\mathbf{A}$, its directed line graph $L(\mathbf{A})$ is a graph such that each node of $L(\mathbf{A})$ represents an edge of $\mathbf{A}$, and there is an edge from a node e_1 to e_2 in $L(\mathbf{A})$ iff for the corresponding edges $\langle i_1, j_1\rangle$ and $\langle i_2, j_2\rangle$ in $\mathbf{A}$, $j_1 = i_2$.*

With the notation of the line graph $L(\mathbf{A})$, we have two equivalent ways to represent an edge. Let e_x $(e_x = 1, ..., m)$ be the index of the nodes (i.e., the edges in $\mathbf{A}$) in the line graph. We can also represent the edge e_x by the pair of its source and target nodes in the original graph $\mathbf{A}$: $\langle i_x, j_x\rangle$, i.e., the edge e_x starts with the node i_x and ends at node j_x.

In order to design an effective strategy to optimize the graph structure to affect the outcome of an information dissemination process, we need to answer the following three questions. (1) *(Key graph parameters/metrics)* What are key graph metrics/parameters that determine/control the dissemination process? (2) *(Graph operations)* What types of graph operations (e.g., deleting nodes/edges, adding edges, etc) are we allowed to change the graph structure? (3) *(Affecting algorithms)* For a given graph operation, how can we design effective, scalable algorithms to optimize the corresponding key graph parameters?

For information dissemination on real graphs, a major finding [41, 33] is that, for a large family of dissemination processes, the largest (in module) eigenvalue λ of the adjacency matrix $\mathbf{A}$ or an appropriately defined system matrix is the *only* graph parameter that determines the tipping point of the dissemination process, i.e., whether or not the dissemination will become an epidemic (see Section 6 for a review of related work). In principle, this gives a clear guidance on the algorithmic side, that is, *an ideal, optimal strategy to affect the outcome of the information dissemination process should*

change the graph structure so that the leading eigenvalue λ is minimized or maximized.

Based on this observation, now we can transform the original problem of affecting the dissemination process to the **eigenvalue optimization problem**, that is,

(1) minimize the leading eigenvalue λ for *NetMelt*;

(2) maximize the leading eigenvalue λ for *NetGel*.

In this paper, we focus on operating on the *edge-level* to design affecting algorithms. With the above notation, our problems can be formally defined as the following two sub-problems:

PROBLEM 1. ***NetMelt* (Edge Deletion)**
Given: *A large $n \times n$ graph $\mathbf{A}$ and an integer (budget) k;*

Output: *A set of k edges from $\mathbf{A}$ whose deletion from $\mathbf{A}$ creates the largest decrease of the leading eigenvalue of $\mathbf{A}$.*

PROBLEM 2. ***NetGel* (Edge Addition)**
Given: *A large $n \times n$ graph $\mathbf{A}$ and an integer (budget) k;*

Find: *A set of k non-edges of $\mathbf{A}$ whose addition to $\mathbf{A}$ creates the largest increase of the leading eigenvalue of $\mathbf{A}$.*

As we will show soon, both problems are combinatorial.

3. PROPOSED ALGORITHM FOR *NetMelt*

In this section, we address the *NetMelt* problem (Prob. 1), that is, to delete k edges from the original graph $\mathbf{A}$ so that its leading eigenvalue λ will decrease as much as possible. We first study the relationship between two different strategies (edge deletion vs. node deletion), and then present our algorithm, followed by the analysis of its effectiveness as well as efficiency.

3.1 Edge Deletion vs. Node Deletion

Roughly speaking, in the *NetMelt* Problem (Edge Deletion), we want to find a set of k 'important' edges from the graph $\mathbf{A}$ to delete. With the notation of the line graph $L(\mathbf{A})$, intuitively, such 'important' edges in $\mathbf{A}$ might become 'important' nodes in the line graph $L(\mathbf{A})$. In this section, we briefly present the relationship between these two strategies (node deletion vs. edge deletion).

Our main result is summarized in Lemma 1, which says that the eigenvalues of the original graph $\mathbf{A}$ are also the eigenvalues of its line graph $L(\mathbf{A})$.

LEMMA 1. **Line Graph Spectrum.** *Let λ be an eigenvalue of the graph $\mathbf{A}$. Then λ is also the eigenvalue of the line graph $L(\mathbf{A})$.*

PROOF. Omitted for brevity. □

By Lemma 1, it seems that edge deletion (Prob. 1) can be transformed to the node deletion problem on the line graph – that is, select a subset of k nodes from the line graph $L(\mathbf{A})$ whose deletion creates the largest decrease in terms of the leading eigenvalue of $L(\mathbf{A})$. However, by the following lemma, the node deletion problem itself is still a challenging task.

LEMMA 2. **Hardness of Node Deletion.** *It is NP-Complete to find a set of k nodes from a graph $\mathbf{A}$, whose deletion will create the largest decrease of the largest eigenvalue of the graph $\mathbf{A}$.*

PROOF. The proof can be done by the reduction from the independent node set problem, which is known to be NP-Complete [17]. The detailed proof is omitted for brevity. □

That said, we seek an effective algorithm that directly solves the *NetMelt* problem next.

3.2 Proposed K-EDGEDELETION Algorithm

The key to solving Prob. 1 (*NetMelt*) is to quantify the impact of deleting a set of edges in terms of the leading eigenvalue λ. The naive way is to recompute the leading eigenvalue λ after deleting the corresponding set of edges - the smaller the new eigenvalue, the better the subset of the edges. But it is computationally infeasible for large graphs since it takes $O(m)$ time for each of the $\binom{m}{k}$ possible sets, as in general, the impact for a given set of the edges (in terms of decreasing the leading eigenvalue λ) is *not* equal to the summation of the impact of deleting each individual edge.

Let $\mathbf{u}$ and $\mathbf{v}$ be the leading left eigenvector and right eigenvector of the graph $\mathbf{A}$, respectively. Intuitively, the left eigen-score $\mathbf{u}(i)$ and the right eigen-score $\mathbf{v}(j)$ $(i, j = 1, ..., n)$ provide some importance measure for the corresponding nodes i and j. The core idea of the proposed K-EDGEDELETION algorithm is to quantify the impact of each edge by the corresponding left and right eigen-scores *independently* (step 9) . Our upcoming analysis in the next subsection shows that this strategy (1) leads to a good approximation of the actual impact wrt decreasing the leading eigenvalue; and (2) naturally de-couples the dependence among the different edges. As a result, we can avoid the combinatorial enumeration in Prob. 1 by picking the top-*k* edges with the highest individual impact scores (step 9).

Note that steps 2-7 in Alg. 1 are to ensure that all the eigen-scores (i.e., $\mathbf{u}(i), \mathbf{v}(j)(i, j = 1, ..., n)$) are non-negative. According to the Perron-Frobenius theorem [10], such eigenvectors $\mathbf{u}$ and $\mathbf{v}$ always exist.

Algorithm 1 K-EDGEDELETION

Input: the adjacency matrix $\mathbf{A}$ and the budget k
Output: k edges
1: compute the leading eigenvalue λ of $\mathbf{A}$; let $\mathbf{u}$ and $\mathbf{v}$ be the corresponding left and right eigenvectors, respectively;
2: **if** $\min_{i=1,...,n} \mathbf{u}(i) < 0$ **then**
3: assign $\mathbf{u} \leftarrow -\mathbf{u}$
4: **end if**
5: **if** $\min_{i=1,...,n} \mathbf{v}(i) < 0$ **then**
6: assign $\mathbf{v} \leftarrow -\mathbf{v}$
7: **end if**
8: **for** each edge $e_x : (i_x, j_x)$ $e_x = 1, ..., m; i_x, j_x = 1, ..., n$ **do**
9: score$(e_x) = \mathbf{u}(i_x)\mathbf{v}(j_x)$;
10: **end for**
11: return top-*k* edges with the highest score(e_x)

3.3 Proofs and Analysis

Here, we analyze the accuracy and the efficiency of the proposed K-EDGEDELETION algorithm.

The accuracy of the proposed K-EDGEDELETION is summarized in Lemma 3. According to Lemma 3, the first-order matrix perturbation theory, together with the fact that many real graphs have large eigen-gap, provides a good approximation to the impact of a set of edges in terms of decreasing the leading eigenvalue. What is more important, with such an approximation, the impact of the different edges are now de-coupled from each other. Therefore, we can avoid the combinatorial enumeration of Prob. 1 by simply returning the top-*k* edges with the highest individual impact scores (step 9 in Alg. 1).

Notice that by Lemma 3, there is an $O(k)$ gap between the approximate and the actual impact of a set of edges in terms of decreasing the leading eigenvalue. Our experimental evaluations show

that the correlation between the approximate and the actual impact is very high (See Section 6 for details), indicating that it indeed provides a good approximation for the actual decrease of the leading eigenvalue.

LEMMA 3. *Let $\hat{\lambda}$ be the (exact) first eigenvalue of $\hat{\mathbf{A}}$, where $\hat{\mathbf{A}}$ is the perturbed version of $\mathbf{A}$ by removing all of its edges indexed by the set $\mathcal{S}$. Let $\delta = \lambda - \lambda_2$ be the eigen-gap of the matrix $\mathbf{A}$ where λ_2 is the second eigenvalue of $\mathbf{A}$, and $c = 1/(\mathbf{u}'\mathbf{v})$. If λ is the simple first eigenvalue of $\mathbf{A}$, and $\delta \geq 2\sqrt{k}$, then $\lambda - \hat{\lambda} = c\sum_{e_x \in \mathcal{S}} \mathbf{u}(i_x)\mathbf{v}(j_x) + O(k)$.*

PROOF. Let $\lambda_i (i = 1, ..., n)$ be the ordered eigenvalues of $\mathbf{A}$ (i.e., $|\lambda| = |\lambda_1| \geq |\lambda_2| ... \geq |\lambda_n|$). Let $\tilde{\lambda}_i (i = 1, ..., n)$ be the corresponding eigenvalues of $\hat{\mathbf{A}}$. Notice that we omitted the subscripts for the leading eigenvalues (i.e., $\lambda_1 = \lambda$, and $\tilde{\lambda}_1 = \tilde{\lambda}$).

Let $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{E}$. We have $\|\mathbf{E}\|_{Fro} = \sqrt{k}$.

According to the first-order matrix perturbation theory (p.183 [38]), we have

$$\begin{aligned} \tilde{\lambda}_1 &= \lambda_1 + \frac{\mathbf{u}'\mathbf{E}\mathbf{v}}{\mathbf{u}'\mathbf{v}} + \mathrm{O}(\|E\|^2) \\ &= \lambda_1 - c\sum_{e_x \in \mathcal{S}} \mathbf{u}(i_x)\mathbf{v}(j_x) + O(k) \end{aligned} \quad (1)$$

Next, we will show that $\tilde{\lambda}_1$ is indeed the leading eigenvalue of $\hat{\mathbf{A}}$. To this end, again by the matrix perturbation theory (p.203 [38]), we have

$$\begin{aligned} \tilde{\lambda}_1 &\geq \lambda_1 - \|\mathbf{E}\|_2 \geq \lambda_1 - \|\mathbf{E}\|_{Fro} \geq \lambda_1 - \sqrt{k} \\ \tilde{\lambda}_i &\leq \lambda_i + \|\mathbf{E}\|_2 \leq \lambda_i + \|\mathbf{E}\|_{Fro} \leq \lambda_i + \sqrt{k} (i \geq 2) \end{aligned} \quad (2)$$

Since $\delta = \lambda_1 - \lambda_2 \geq 2\sqrt{k}$, we have $\tilde{\lambda}_1 \geq \tilde{\lambda}_i (i = 2, ..., n)$. In other words, we have that $\tilde{\lambda}_1 = \hat{\lambda}$ is the leading eigenvalue of $\hat{\mathbf{A}}$. Therefore,

$$\lambda - \hat{\lambda} = c\sum_{e_x \in \mathcal{S}} \mathbf{u}(i_x)\mathbf{v}(j_x) + O(k) \quad (3)$$

which completes the proof. □

The efficiency of the proposed K-EDGEDELETION is summarized in the following lemma, which says that with a fixed budget k, K-EDGEDELETION is *linear* wrt the size of the graph for both time and space cost.

LEMMA 4. **Efficiency of K-EDGEDELETION.** *The time cost of Alg. 1 is $O(mk + n)$. The space cost of Alg. 1 is $O(n + m + k)$.*

PROOF. Using the power method, step 1 takes $O(m)$ time. Steps 2-7 take $O(n)$ time. Steps 8-10 take $O(m)$ time. Step 11 takes $O(mk)$ time. Therefore, the overall time complexity of Alg. 1 is $O(mk + n)$, which completes the proof of the time cost.

We need $O(m)$ to store the original graph $\mathbf{A}$. It takes $O(n)$ and $O(1)$ to store the eigenvectors and eigenvalue, respectively. We need additional $O(m)$ to store the scores (Step 9) for all the edges. Finally, it takes $O(k)$ for the selected k edges. Therefore, the overall space complexity of Alg. 1 is $O(m + n + k)$, which completes the proof of the space cost. □

4. PROPOSED ALGORITHM FOR *NetGel*

In this Section, we address the *NetGel* problem (Prob. 2), where we want to *add* a set of new links into the graph $\mathbf{A}$ so that its leading eigenvalue λ will increase as much as possible. We first present the proposed K-EDGEADDITION algorithm, and then analyze its accuracy as well as efficiency.

4.1 Proposed K-EDGEADDITION Algorithm

Let $\mathcal{T}$ be a set of non-existing edges in $\mathbf{A}$, that is, for each $e_x : \langle i_x, j_x \rangle \in \mathcal{T}$, we have $\mathbf{A}(i_x, j_x) = 0$. Let $\hat{\lambda}$ be the leading eigenvalue of the new adjacency matrix $\hat{\mathbf{A}}$ by introducing the new edges indexed by the set $\mathcal{T}$. By the similar procedure as in the proof of Lemma 3, we can show that the impact of the new set of edges $\mathcal{T}$ in terms of increasing the leading eigenvalue $\hat{\lambda} - \lambda$ can be approximated as

$$\hat{\lambda} - \lambda \approx \sum_{e_x \in \mathcal{T}} \mathbf{u}(i_x)\mathbf{v}(j_x) \quad (4)$$

Therefore, it seems that we could use a similar procedure as K-EDGEDELETION to solve the *NetGel* problem (referred to as '*Naive-Add*'): for each non-existing edge $e_x : \langle i_x, j_x \rangle$, calculate its score as $\text{score}(e_x) = \mathbf{u}(i_x)\mathbf{v}(j_x)$; and pick top-*k* non-existing edges with the highest scores.

However, many real graphs are very sparse, i.e., $m << n^2$. Therefore, we have $O(n^2 - m) \approx O(n^2)$ possible non-existing edges. In other words, *Naive-Add* requires *quasi-quadratic* time wrt the number of the nodes (n) in the graph, which does not scale to large graphs.

To address this issue, we propose an efficient algorithm, which is summarized in Alg 2. The core idea of K-EDGEADDITION is to prune a large portion of the non-existing edge pairs based on their left and right eigen-scores. As in Alg. 1, we take the same procedure to make sure that the left and right eigenvectors $(\mathbf{u}, \mathbf{v})$ are non-negative. We omit these steps in Alg 2 for brevity.

Algorithm 2 K-EDGEADDITION

Input: the adjacency matrix $\mathbf{A}$ and the budget k
Output: k non-existing edges

1: compute the left ($\mathbf{u}$) and right ($\mathbf{v}$) eigenvectors of $\mathbf{A}$ that correspond to the leading eigenvalue ($\mathbf{u}, \mathbf{v} \geq 0$);
2: calculate the maximum in-degree (d_{in}) and out-degree (d_{out}) of $\mathbf{A}$, respectively;
3: find the subset of $k + d_{in}$ nodes with the highest left eigenscores $\mathbf{u}_i$. Index them by $\mathcal{I}$;
4: find the subset of $k + d_{out}$ nodes with the highest right eigenscores $\mathbf{v}_j$. Index them by $\mathcal{J}$;
5: **for** each edge $e_x : \langle i_x, j_x \rangle$ $i_x \in \mathcal{I}, j_x \in \mathcal{J}, \mathbf{A}(i_x, i_j) = 0$ **do**
6: score$(e_x) = \mathbf{u}(i_x)\mathbf{v}(j_x)$. Index them by $\mathcal{P}$;
7: **end for**
8: return top-*k* non-existing edges with the highest scores among $\mathcal{P}$.

4.2 Proofs and Analysis

Here, we analyze the accuracy and efficiency of the proposed K-EDGEADDITION.

The accuracy of the proposed K-EDGEADDITION is summarized in Lemma 5, which says that K-EDGEADDITION selects the same set of edges as *Naive-Add*.

LEMMA 5. **Effectiveness of K-EDGEADDITION.** *Alg. 2 outputs the same set of non-existing edges as* Naive-Add.

PROOF. Omitted for brevity. □

The efficiency of the proposed K-EDGEADDITION is summarized in the following lemma.

LEMMA 6. **Efficiency of K-EDGEADDITION.** *The time cost of Alg. 2 is $O(m + nt + kt^2)$. The space cost of Alg. 2 is $O(n + m + t^2)$, where $t = max(k, d_{in}, d_{out})$.*

PROOF: Using the power method, step 1 takes $O(m)$ time. Step 2 takes $O(m+n)$ time. Steps 3-4 take $O(n(d_{in}+k))$ and $O(n(d_{out}+k))$ time respectively, both of which can be written as $O(nt)$. Steps 5-7 take $O((k+d_{in})(k+d_{out})) = O(t^2)$ time. Step 8 takes $O((k+d_{in})(k+d_{out})k) = O(kt^2)$. Therefore, the overall time cost is $O(m+nt+kt^2)$, which completes the proof of the time complexity.

We need $O(m)$ to store the original graph $\mathbf{A}$. It takes $O(n)$ to store the eigenvectors $\mathbf{u}$ and $\mathbf{v}$. Step 2 takes additional $O(n+1)$ space. Steps 3-4 take $O(d_{in}+k)$ and $O(d_{out}+k)$ space respectively, both of which can be simplified as $O(t)$. Steps 5-7 take at most $O((k+d_{in})(k+d_{out})) = O(t^2)$ space. Step 9 takes $O(k)$ space. Therefore, the overall space cost (by omitting the smaller terms) is $O(m+nt+kt^2)$, which completes the proof of the space complexity. □

5. EXPERIMENTAL EVALUATIONS

In this section, we provide empirical evaluations for the proposed K-EDGEDELETION and K-EDGEADDITION algorithms. Our evaluations mainly focus on (1) the effectiveness and (2) the efficiency of the proposed algorithms.

5.1 Experimental Setup

Data sets. We used a popular set of real graphs for our experiments - the Oregon AS (Autonomous System) router graphs, which are AS-level connectivity networks inferred from Oregon route-views[2]. These were collected once a week, for 9 consecutive weeks. Table 2 summarizes the nine graphs we used in our evaluations.

Evaluation criteria. As mentioned before, the leading eigenvalue λ of the graph is the only graph parameter that determines the epidemic threshold for a large family of information dissemination processes. Therefore, we report the change of the leading eigenvalue for the effectiveness comparison - for both *NetMelt* and *NetGel* problems. A larger change of the leading eigenvalue is better, which suggests that we can affect the outcome of the dissemination process more. In addition, we also run virus propagation simulations to compare how different methods affect the actual outcome of the propagation. For the computational cost and scalability, we report the wall-clock time.

Machine configurations. All the experiments ran on the same machine with four 2.4GHz AMD CPUs and 48GB memory, running Linux (2.6 kernel).

5.2 Effectiveness of K-EDGEDELETION

Approximation Quality. For both K-EDGEDELETION and K-EDGEADDITION, we want to approximate the actual change of the leading eigenvalue by the first order matrix perturbation theory. This is the *only* place we introduce the approximation. By Lemma 3, it says that the quality of such an approximation depends on both the budget k as well as the eigengap of the original graph, with an $O(k)$ gap. Here, let us experimentally evaluate how good this approximation is on real graphs. We compute the linear correlation coefficient between the actual and approximate leading eigenvalue after we randomly remove k ($k = 10, 50, 100, 500, 1000$) edges. The results are shown in table 3. It can be seen that the approximation is very good - in all the cases, the linear correlation coefficient is greater than 0.92, and often it is very close to 1.

The Impact of Decreasing the Leading Eigenvalue. Here, we evaluate the effectiveness of the proposed K-EDGEDELETION in terms of decreasing the leading eigenvalue λ of the graph. Lemma 1 suggests that the 'important' edges on the original graph $\mathbf{A}$ might become 'important' nodes on the line graph $L(\mathbf{A})$. We follow this intuition to design the following comparative strategies: (1) randomly select k edges from the original graph $\mathbf{A}$ (referred to as 'Rand'); (2) select k edges with the highest degrees in the line graph $L(\mathbf{A})$ (referred to as 'Line-Deg'); (3) select k edges with the highest eigen-scores in the line graph $L(\mathbf{A})$ (referred to as 'Line-Eig'); and (4) select k edges with the highest PageRank scores in the line graph $L(\mathbf{A})$ (referred to as 'Line-Page'). For 'Rand', we run the experiments 100 times and report the average result. For 'Line-Deg', we have two variants by using out-degree or in-degree. In our evaluation, we found that these two variants give the similar results. Therefore, we only report the results by out-degree. For the same reason, we only report the results by the right eigen-scores for 'Line-Eigs'. For 'Line-Page', there is an additional parameter of the teleport probability. We run the experiments with the different teleport probabilities and report the best results.

For brevity, we only present the results on *Oregon-A*, *Oregon-B* and *Oregon-C* since the results on the rest six graphs are similar. From Fig. 2, it can be seen that our K-EDGEDELETION always leads to the biggest decrease in terms of the leading eigenvalue. For example, on *Oregon-C* graph, the proposed K-EDGEDELETION decreases the leading eigenvalue by *3.8* with the budget $k = 50$, which is almost double of the second best method (e.g., *2.0* by 'Line-Deg'). Therefore, we expect that K-EDGEDELETION would affect the outcome of the dissemination processes better than the alternative choices, e.g., having less number of infected nodes in the graph, etc. We validate this next.

Affecting Virus Propagation. Next, we evaluate the effectiveness of the proposed K-EDGEDELETION in terms of minimizing the outcome of the information dissemination processes. To this end, we simulate the virus propagation for the SIS model (susceptible-infective-susceptible) on the graph [41]. For each method, we delete $k = 200$ edges from the original graph. Let $s = \lambda b/d$ be the normalized virus strength (bigger s means stronger virus), where b and d are the infection rate and death rate, respectively. The results are presented in Fig. 3, which is averaged over 1,000 runs. It can be seen that the proposed K-EDGEDELETION is always the best - its curve is always the lowest which means that we always, as desired, have the least number of infected nodes in the graph with this strategy. In Fig. 3, 'Original' (the yellow curve) means that we simulate the virus propagation on the *original* graph without deleting any edges. Notice that when the virus becomes stronger (Fig. 3(b)), all the curves except the proposed method mix with 'Original', which means that they all fail to affect the virus propagation in this case. In contrast, our proposed method (the red curve) can still significantly reduce the number of infected nodes.

Node Deletion vs. Edge Deletion. Finally, in some applications, e.g., to stop malware propagation on the computer networks, both node deletion (e.g., shutting down some machines) and edge deletion (e.g., blocking some links between machines) are feasible. In this case, we want to know which strategy (node deletion or edge deletion) is more effective in affecting the outcome of such propagation process. To this end, we use an effective node immunization algorithm [39] to delete $\tilde{k} = 1, 10$ nodes respectively (referred to as 'Node-Del'). For each $\tilde{k}$, we then use our proposed K-EDGEDELETION to delete the same amount of edges from the original graph (referred to as 'Edge-Del'). We compare the decrease of the leading eigenvalues of the two methods. The results are summarized in Fig. 4. It can be seen that 'Edge-Del' always leads to a bigger decrease of the leading eigenvalue - which suggests that by operating on the edge level, we can design a more

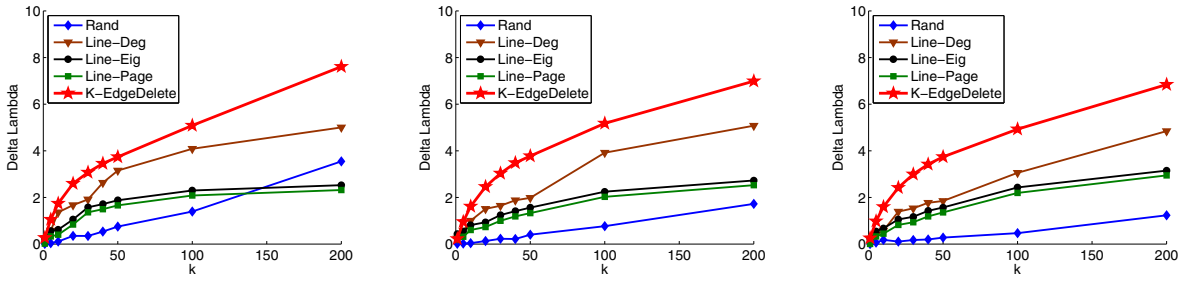
[2] http://topology.eecs.umich.edu/data.html

Dataset	n	m
Oregon-A	633	2,172
Oregon-B	1,503	5,620
Oregon-C	2,504	9,446
Oregon-D	2,854	9,864
Oregon-E	3,995	15,420
Oregon-F	5,296	20,194
Oregon-G	7,352	31,330
Oregon-H	10,860	46,818
Oregon-I	13,947	61,168

Table 2: Dataset summary.

Dataset	$k = 10$	$k = 50$	$k = 100$	$k = 500$	$k = 1000$
Oregon-A	0.999	0.997	0.995	0.973	0.924
Oregon-B	0.999	0.999	0.998	0.993	0.988
Oregon-C	1.000	0.999	0.999	0.996	0.991
Oregon-D	0.999	0.999	0.999	0.994	0.988
Oregon-E	1.000	0.999	0.999	0.998	0.995
Oregon-F	1.000	0.999	0.999	0.998	0.997
Oregon-G	1.000	0.999	0.999	0.999	0.998
Oregon-H	1.000	1.000	0.999	0.999	0.999
Oregon-I	1.000	1.000	0.999	0.999	0.999

Table 3: Evaluations on the approx. quality. Larger is better.



(a) *Oregon-A* (b) *Oregon-B* (c) *Oregon-C*

Figure 2: The decrease of the leading eigenvalue vs. the budget k. Larger is better. The proposed K-EDGEDELETION always leads to the biggest decrease of the leading eigenvalue.

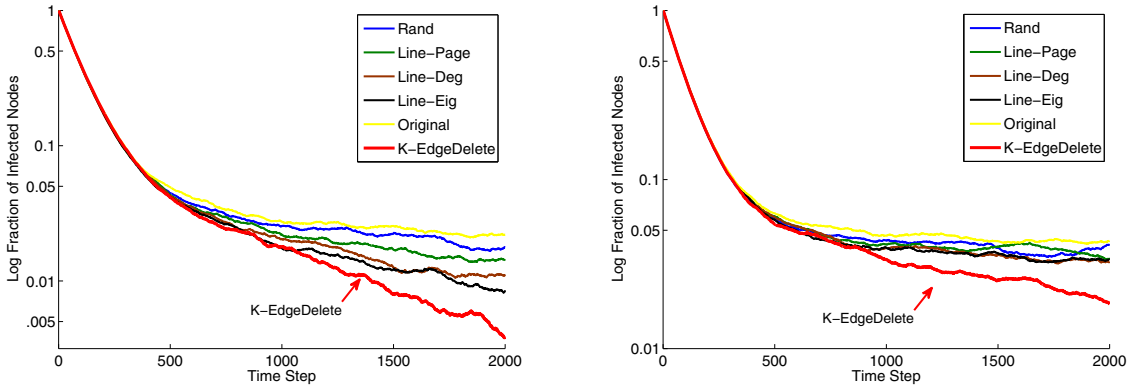
effective algorithm with the same budget to affect the outcome of the information dissemination process. The results are consistent with the intuition - not all the edges adjacent to the 'important' nodes, which the node immunization algorithm aims to delete, are also 'important' (e.g., many edges adjacent to an 'important' node might link to/from some degree-1 nodes). In other words, edge deletion enables us to optimize the underlying graph structure on a finer granularity by picking each individual edge one by one.

5.3 Effectiveness of K-EDGEADDITION

To our best knowledge, there are no existing methods to add k new links into an existing graph in order to increase its leading eigenvalue. Let $\bar{\mathbf{A}}$ be the complementary graph of $\mathbf{A}$, which has the same node set as $\mathbf{A}$, and $\bar{\mathbf{A}}(i,j) = 1$ iff $\mathbf{A}(i,j) = 0$. With the notation of the complementary graph, we use the following intuition to design the comparative methods: to select k 'important' edges from the *complementary graph* $\bar{\mathbf{A}}$ and add them into the original graph $\mathbf{A}$. More specifically, we compare the proposed K-EDGEADDITION with the following strategies: (1) randomly select k edges (referred to as 'Rand'); (2) select k edges with the highest out-degrees in the line graph of the complementary graph $\bar{\mathbf{A}}$ (referred to as 'CompDeg'); (3) select k edges with the highest right eigen-scores in the line graph of the complementary graph $\bar{\mathbf{A}}$ (referred to as 'CompEigs'); (4) select k edges with the highest PageRank scores in the line graph of the complementary graph $\bar{\mathbf{A}}$ (referred to as 'CompPage'); and (5) select k edges by running K-EDGEDELETION in the complementary graph $\bar{\mathbf{A}}$ (referred to as 'CompDelete'). Again, for 'Rand', we run the experiments 100 times and report the average result. We only report the results of 'CompDeg' by out-degree and those of 'CompEig' by right eigen-scores, respectively, since the other variants give the similar performance. For 'CompPage', we run the experiments with the different teleport probabilities and report the best results.

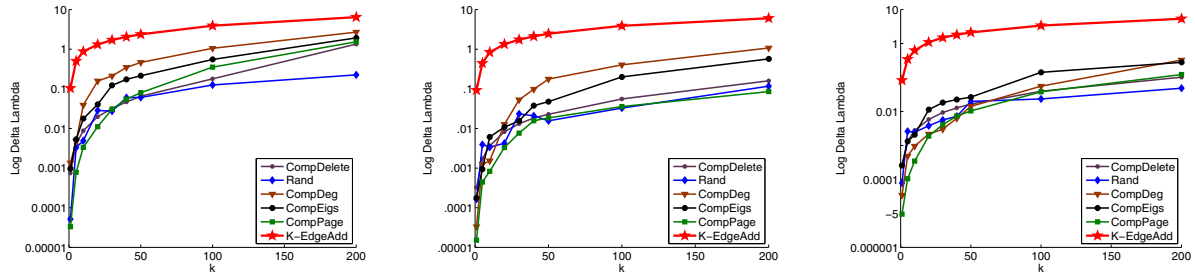
The Impact of Increasing the Leading Eigenvalue. We first evaluate the effectiveness of the proposed K-EDGEADDITION in terms of *increasing* the leading eigenvalue of the graph. For brevity, we only present the results on *Oregon-A*, *Oregon-B* and *Oregon-C* since the results on the rest of the graphs are similar. From Fig. 5, it can be seen that the proposed K-EDGEADDITION always leads to the biggest increase in terms of the leading eigenvalue of the graph. Notice that for all the comparative methods, they behave like 'Rand' (blue curve), especially when the budget k is small.

Affecting Virus Propagation. We also evaluated the effectiveness of the proposed K-EDGEADDITION in terms of *maximizing* the outcome of the information dissemination process. To this end, again, we simulate the virus propagation for the SIS model on the graph. For each method, we add $k = 200$ new edges into the graph. Again, let $s = \lambda b/d$ be the normalized virus strength, with bigger s being stronger virus. Here, our goal is to *increase* the number of 'infected' nodes (e.g., having more people in the social networks to adopt a piece of good idea, etc) by introducing a set of new links into the graph. The result is presented in Fig. 6, which is averaged over 1,000 runs. It can be seen that the proposed K-EDGEADDITION is always the best - its curve is always the highest which means that we always have the largest number of 'infected' nodes in the graph with this strategy. Notice that when the strength of the virus is weak (Fig. 6(a)), all the curves except the proposed method mix with or are very close to 'Original' (yellow curve), which means that they have little impact to boost the outcome of the propagation in this case. In contrast, our proposed method (the red curve) can still significantly increase the number of 'infected' nodes. Therefore, we conclude that our proposed K-



(a) $s = 2.8$ (b) $s = 4.3$

Figure 3: Comparison of minimizing the outcome of the virus propagation. Fraction of infected nodes vs. time stamp. Lower is better. The proposed K-EDGEDELETION always leads to the least number of infected nodes. Notice that y-axis is in the logarithmic scale.



(a) *Oregon-A* (b) *Oregon-B* (c) *Oregon-C*

Figure 5: The increase of the leading eigenvalue vs. the budget k. Larger is better. The proposed K-EDGEADDITION always leads to the largest increase of the leading eigenvalue. Notice that y-axis is in the logarithmic scale.

EDGEADDITION is much more effective to guild the outcome of the dissemination process.

5.4 Scalability

We use the subsets of the largest data set *Oregon-I* to evaluate the scalability of the proposed algorithms. The results are presented in Fig. 7. We can see that the proposed K-EDGEDELETION and K-EDGEADDITION scale almost near-linearly wrt m, which means that they are suitable for large graphs. Notice that for both cases, we also observe a slight super-linear trend. This is due to the following two reasons: (1) for both K-EDGEDELETION and K-EDGEADDITION, we use the power method to compute the leading eigenvalue and the corresponding eigenvectors. When m increases, the actually iteration number in the power method also tends to increase; (2) for K-EDGEADDITION when m increases, the maximum degree ($\max(d_{in}, d_{out})$) also increases even though we fix the number of the nodes (n).

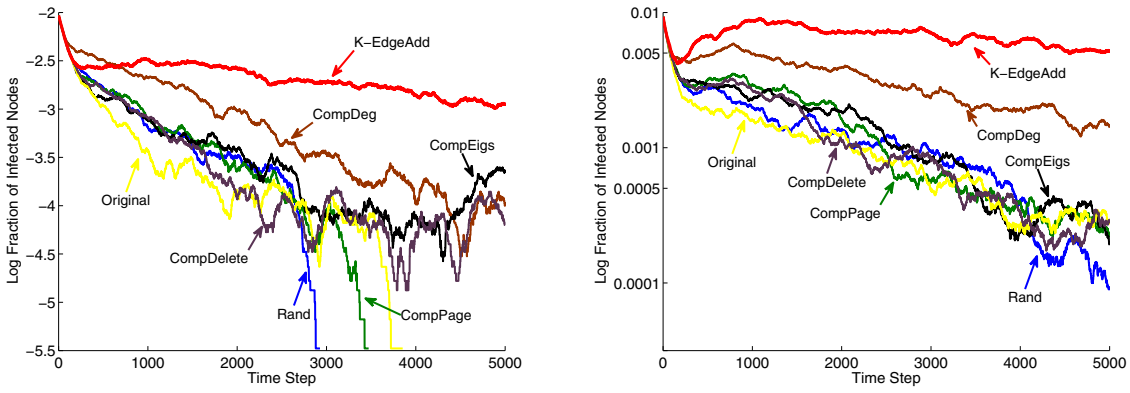
6. RELATED WORK

In this section, we review the related work, which can be categorized into three parts: information dissemination, affecting algorithms and node/edge importance measure.

Information Dissemination. Many research works in virus propagation have been devoted to studying the so-called epidemic threshold, that is, to determine the condition under which an epidemic will break out. While earlier works [13] focus on some specific types of graph structure (e.g., random graphs, power-law graphs, etc), Wang et al. [41] and its follow-up paper by Ganesh et al. [8] found that, for the flu-like SIS model, the epidemic threshold for any *arbitrary, real* graph is determined by the leading eigenvalue of the adjacency matrix of the graph. Prakash et. al. [33] further discovered that the leading eigenvalue (and a model-dependent constant) is the only parameter that determines the epidemic threshold for *all* virus propagation models (more than 25 models, including H.I.V.) in the standard literature. In this work, we aim to take one step further, i.e., how to optimize (minimize or maximize) the leading eigenvalue of the graph by deleting or adding a set of links.

There are also many research interest in studying other types of information dissemination processes on large graphs, including (a) information cascades [1, 9], (b) blog propagations [24, 11, 21, 35], and (c) viral marketing and product penetration [18, 23].

Affecting Algorithms. Hayashi et al. [12] derived the extinction conditions under random and targeted immunization for the SHIR model (Susceptible, Hidden, Infectious, Recovered). Tong et



(a) $s = 1.1$ (b) $s = 1.3$

Figure 6: Comparison of maximizing the outcome of virus propagation. Fraction of 'infected' nodes vs. time stamp. Larger is better. The proposed K-EDGEADDITION always leads to the largest number of 'infected' nodes. Notice that y-axis is in the logarithmic scale.

al. [39] proposed an effective node immunization strategy for the SIS model by approximately minimizing the leading eigenvalue. Briesemeister et al. [3] studied the defending policy in power-law graphs. Prakash et. al. [34, 40] proposed effective algorithms to perform node immunization on time-varying graphs. Other algorithms to affect the outcome of the information dissemination include the influence maximization [18, 6, 5], finding effectors in social networks [22], etc. Notice that all these works focus on operating on the node level (i.e., delete or inoculate a set of 'best' nodes) to affect the outcome of the dissemination. In contrast, we study the equally important, but much less studied affecting algorithms by operating on the edge level.

There exist some *empirical evaluations* on edge removal strategies for slightly different purposes, such as, slowing down the influenza spreading [26], minimizing the average infection probability [36], evaluating and comparing the attack vulnerability [14], etc. The closest related work to our K-EDGEDELETION algorithm is [2], which proposed a convex optimization based approach to approximately minimize the leading eigenvalue of the graph. However, the method is based on semi-definite programming and does not scale to large graphs. Moreover, for all these methods, it remains unclear if they can be generalized to address the even more challenging *NetGel* problem, where we want to *add* new edges to promote the information dissemination.

Measuring the Importance of Nodes and Edges. In the literature, there are a lot of node importance measurements, including betweenness centrality, both the one based on the shortest path [7] and the one based on random walks [29, 16] PageRank [30], HITS [19], and coreness score [28]. Our work is also related to the so-called k-vital edges problem, which aims to delete a set of links from the graphs to increase the shortest path length [25] or the weight of the minimum spanning tree of the remaining graph [37]. K-vital edge problem itself is known to be NP-Hard. Other remotely related work includes graph augmentation [31, 4], graph sparsification [20], network inhibition [32] and network-interdiction [42, 15]. Both network inhibition and network interdiction are NP-Hard.

7. CONCLUSION

In this paper, we study the problem of how to optimize the link structure to affect the outcome of information dissemination processes. The main contributions of the paper are:

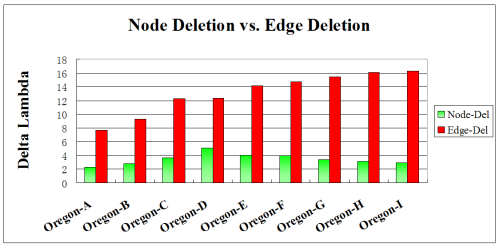
- *Algorithms.* We observe that for a large family of information dissimilation processes, the problem boils down to the eigenvalue optimization problem. We propose an effective, scalable algorithm to optimize such a key graph parameter (i.e., the leading eigenvalue) that controls the information dissemination process, for both *NetMelt* and *NetGel*, respectively;

- *Proofs and Analysis.* We show the *accuracy* (Lemma 3 and Lemma 5) and the *complexity* of our methods (Lemma 4 and Lemma 6); the *hardness* of the problem (Lemma 2), and the *equivalence* between the different strategies (Lemma 1, Lemma 7 and Lemma 8);

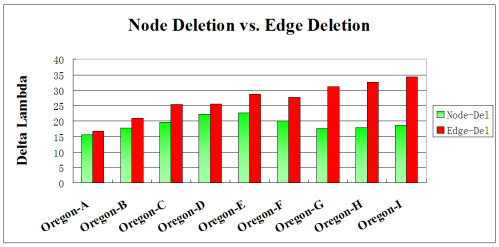
- *Experimental Evaluations.* Our evaluations on real large graphs show that (a) compared with alternative choices to optimize the link structure, our methods are much more effective to affect the outcome of the dissemination process; (b) compared with the node deletion strategy, our K-EDGEDELETION offers a more effective way by operating on the edge level; and (c) both K-EDGEDELETION and K-EDGEADDITION scale to large graphs.

8. ACKNOWLEDGEMENT

This material is based upon work supported by the Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053, the U.S. Defense Advanced Research Projects Agency (DARPA) under Agreement Number W911NF-12-C-0028, and the National Science Foundation under Grant No. IIS-1017415. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

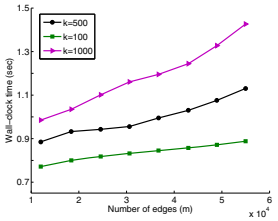


(a) $\tilde{k} = 1$

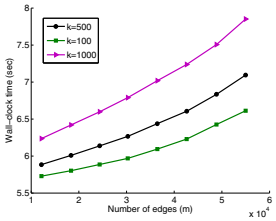


(b) $\tilde{k} = 10$

Figure 4: Comparison between node deletion vs. edge deletion. Larger is better. With the same amount of edges deleted, our proposed K-EDGEDELETION (red) leads to a bigger decrease in terms of the leading eigenvalue.



(a) K-EDGEDELETION



(b) K-EDGEADDITION

Figure 7: Scalability of proposed algorithms. Both K-EDGEDELETION and K-EDGEADDITION scale near-linearly wrt the size of the graph.

9. REFERENCES

[1] S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change in informational cascades. *Journal of Political Economy*, 100(5):992–1026, October 1992.

[2] A. N. Bishop and I. Shames. Link operations for slowing the spread of disease in complex networks. *EPL*, 95(1), 2011.

[3] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. *WORM 2003*, Oct. 27 2003.

[4] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt. Recommendations to boost content spread in social networks. In *WWW*, pages 529–538, 2012.

[5] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.

[6] S. Datta, A. Majumder, and N. Shrivastava. Viral marketing for multiple products. In *ICDM*, pages 118–127, 2010.

[7] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[8] A. Ganesh, E. Massouli, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM*, 2005.

[9] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 2001.

[10] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[11] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04*, 2004.

[12] Y. Hayashi, M. Minoura, and J. Matsukubo. Recoverable prevalence in growing scale-free networks and the effective immunization. *arXiv:cond-mat/0305549 v2*, Aug. 6 2003.

[13] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42:599–653, 2000.

[14] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Phys. Rev. E*, 2002.

[15] E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[16] U. Kang, S. Papadimitriou, J. Sun, and H. Tong. Centralities in large networks: Algorithms and observations. In *SDM*, pages 119–130, 2011.

[17] R. M. Karp. *Reducibility Among Combinatorial Problems*. New York, 1972.

[18] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[20] A. Kolla, Y. Makarychev, A. Saberi, and S.-H. Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. In *STOC*, pages 57–66, 2010.

[21] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW*, 2003.

[22] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *KDD*, pages 1059–1068, 2010.

[23] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *EC*, pages 228–237, New York, NY, USA, 2006. ACM Press.

[24] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs: Patterns and a model. In *Society of Applied and Industrial Mathematics: Data Mining (SDM07)*, 2007.

[25] W. Liang, X. Shen, and Q. Hu. Finding the most vital edge for graph minimization problems on meshes and hypercubes. *International Journal of Parallel and Distributed Systems and Networks*, 2000.

[26] J. Marcelino and M. Kaiser. Reducing influenza spreading over the airline network. *PLoS*, 2009.

[27] A. Milanese, J. Sun, and T. Nishikawa. Approximating spectral impact of structural perturbations in large networks. *Phys. Rev. E*, 81, 2010.

[28] J. Moody and D. R. White. Social cohesion and embeddedness: A hierarchical conception of social groups. *American Sociological Review*, pages 1–25, 2003.
[29] M. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.
[30] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).
[31] M. Papagelis, F. Bonchi, and A. Gionis. Suggesting ghost edges for a smaller world. In *CIKM*, pages 2305–2308, 2011.
[32] C. A. Phillips. The network inhibition problem. In *STOC*, pages 776–785, 1993.
[33] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*, 2011.
[34] 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 (3)*, pages 99–114, 2010.
[35] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. *SIGKDD*, 2002.
[36] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann. Restraining epidemics by improving immunization strategies. *CoRR*, abs/1102.1929, 2011.
[37] H. Shen. Finding the k most vital edges with respect to minimum spanning tree. *Acta Informatica*, 36:405–424, 1995.
[38] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.
[39] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *ICDM*, pages 1091–1096, 2010.
[40] N. Valler, B. A. Prakash, H. Tong, M. Faloutsos, and C. Faloutsos. Epidemic spread in mobile ad hoc networks: Determining the tipping point. In *Networking (1)*, pages 266–280, 2011.
[41] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. *SRDS*, 2003.
[42] R. K. Wood. Network interdiction problem. *Mathematical and Computer Modeling*, 17(2):1–18, 1993.

APPENDIX

Higher-Order *NetMelt*. From Lemma 3, it can be seen that the only place we introduce the approximation in Alg. 1 is to approximate the actual decrease of the leading eigenvalue by the first-order matrix perturbation theory. The readers might wonder if we can further improve the quality by using higher-order matrix perturbation theory, while maintaining the linear scalability of the algorithm.

We explored second-order matrix perturbation theory to approximate the actual decrease of the leading eigenvalue, and found that (1) it generates very similar results as the proposed K-EDGEDELETION algorithm and (2) it requires 5-10x more wall-clock time. The reason might be that for the *NetMelt* problem, the first-order perturbation already gives a very good approximation. Therefore, in practice, we recommend K-EDGEDELETION for simplicity.

Nonetheless, the new algorithm based on the second-order perturbation exhibits some interesting theoretic properties. It also helps understand the relationship between edge deletion and node deletion on the algorithmic level. We present it here for the completeness.

Let $c = \frac{1}{\mathbf{u}'\mathbf{v}}$, with second-order matrix perturbation, we can approximate[3] the impact of deleting a set of edges $\mathcal{S}$ in terms of the leading eigenvalue as:

$$\lambda - \hat{\lambda} \simeq \text{Impact}(\mathcal{S}) = c\Big(\sum_{e_x \in \mathcal{S}} \mathbf{u}(i_x)\mathbf{v}(j_x) - \frac{1}{2\lambda} \sum_{e_x \in \mathcal{S}, e_y \in \mathcal{S}, j_x = i_y} \mathbf{u}(i_x)\mathbf{v}(j_y)\Big) \quad (5)$$

Compared with the first-order perturbation (eq. (3)), we have an additional penalized term in eq. (5): $\mathbf{u}(i_x)\mathbf{v}(j_y)$ for any two adjacent edges e_x and e_y. The intuition is to encourage the edges in the set $\mathcal{S}$ to be far away (not adjacent) from each other.

By eq. (5), the impact of different edges in the set $\mathcal{S}$ is no longer independent with each other. At the first glance, this might complicate the algorithm since now we need to optimize at the set level, that is, to find a set of edges that *collectively* maximize eq. (5). However, by the following lemma, the impact defined in eq. (5) exhibits some nice diminishing return properties.

LEMMA 7. **Second-Order Approximation Properties**. *The Impact($\mathcal{S}$) defined in eq.* (5) *has the following properties:*

(1) Impact(Φ) = 0, where Φ is an empty set;
(2) Impact($\mathcal{S}$) is monotonically non-decreasing wrt the set $\mathcal{S}$;
(3) Impact($\mathcal{S}$) is sub-modular wrt the set $\mathcal{S}$.

PROOF. Omitted for brevity. □

Thanks to such diminishing return properties, it naturally leads to the following greedy algorithm (K-EDGEDELETION++) to find a *near-optimal* subset of edges to delete from the original graph $\mathbf{A}$. And it can be shown that the overall time complexity of K-EDGEDELETION++ remains linear wrt the size of the graph.

Algorithm 3 K-EDGEDELETION++

Input: the adjacency matrix $\mathbf{A}$ and the budget k
Output: k edges indexed by set $\mathcal{S}$

```
compute the first eigen-value λ of A; compute the corresponding left and right eigenvectors u and v (u, v ≥ 0), respectively;
initialize the set S to be empty;
score(e_x) = u(i_x)v(j_x) (e_x : ⟨i_x, j_x⟩, e_x = 1, ..., m);
for k_0 = 1, ..., k do
    find e_0 = argmax_{e_x, e_x ∉ S} score(e_x);
    add the new edge e_0 : (i_0, j_0) into S;
    for each edge e_y : ⟨i_y, j_y⟩ s.t. j_y = i_0 do
        score(e_y) ← score(e_y) − 1/(2λ)u(i_y)v(j_0);
    end for
    for each edge e_y : ⟨i_y, j_y⟩ s.t. i_y = j_0 do
        score(e_y) ← score(e_y) − 1/(2λ)u(i_0)v(j_y);
    end for
end for
```

An interesting property of Alg. 3 is that it builds the equivalence between edge deletion and node deletion on the algorithmic level:

LEMMA 8. **Equivalence of Alg. 3 to Node Immunization**. *Let $\mathcal{S}$ be the set of edges by running Alg. 3 on graph $\mathbf{A}$; $\mathcal{T}$ be the set of edges by running the node immunization algorithm [39] on the line graph $L(\mathbf{A})$; and $|\mathcal{S}| = |\mathcal{T}|$. We have $\mathcal{S} = \mathcal{T}$.*

PROOF. Omitted for brevity. □

[3]This formulas is similar as the one in [27]