Scalable Diffusion-Aware Optimization of Network Topology

Elias Khalil
lyes@gatech.edu

Bistra Dilkina
bdilkina@cc.gatech.edu

Le Song
lsong@cc.gatech.edu

School of Computational Science & Engineering
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765

ABSTRACT

How can we optimize the topology of a networked system to bring a flu under control, propel a video to popularity, or stifle a network malware in its infancy? Previous work on information diffusion has focused on modeling the diffusion dynamics and selecting nodes to maximize/minimize influence. Only a paucity of recent studies have attempted to address the network modification problems, where the goal is to either facilitate desirable spreads or curtail undesirable ones by adding or deleting a small subset of network nodes or edges. In this paper, we focus on the widely studied *linear threshold diffusion model*, and prove, for the first time, that the network modification problems under this model have *supermodular* objective functions. This surprising property allows us to design efficient data structures and scalable algorithms with provable approximation guarantees, despite the hardness of the problems in question. Both the time and space complexities of our algorithms are *linear* in the size of the network, which allows us to experiment with millions of nodes and edges. We show that our algorithms outperform an array of heuristics in terms of their effectiveness in controlling diffusion processes, often beating the next best by a significant margin.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications — *Data Mining*
General Terms: Algorithms, Theory, Experimentation
Keywords: diffusion networks; network optimization; supermodularity; approximation

1. INTRODUCTION

The diffusion of physical, conceptual or digital substances over networks has been studied in many domains such as epidemiology [10], social media [16], and computer and mobile [19] networks. These studies have resulted in an array of models aiming to capture the diffusion dynamics, among which the most well-known are the linear threshold (LT) model, the independent cascade (IC) model, and the Susceptible Infected Recovered (SIR) model. Starting with the work of Domingos and Richardson on viral marketing [6], a lot of research has concentrated on how one can pick a small set of source nodes, whose initial adoption of a given substance would trigger maximal spread in the network. The seminal work of Kempe et al. [12] showed that *source node selection for influence maximization* is a submodular maximization problem under the IC and LT diffusion models, and therefore admits a simple greedy algorithm with approximation guarantees. Their result was followed by a whole line of research on source node selection in various related information diffusion contexts [4, 7, 9].

In contrast to these previous works where *the diffusion networks remain unchanged*, we are rather interested in problems of *modifying the topology of a diffusion network* to either facilitate the spread of desirable substances, or curtail the spread of undesirable ones. One can consider deleting edges or nodes to minimize possible undesirable spread, such as that of a virus, disease or rumor, or one can consider adding edges or nodes to facilitate, for example, the spread of information or dispersal of endangered species. For instance, in disease control, authorities may consider disallowing travel between certain pairs of cities to curb the spread of a flu epidemic. On the other hand, social media websites can recommend to users additional information outlets to follow to increase the spread of ideas and memes. The modification setting is particularly relevant when the agent optimizing the topology does not have control over the sources of the spread, but is able to change some subset of the edges or nodes that he has access to. Hence, our setting is most relevant when the agent in question is looking to *strategically design* the topology, as opposed to reacting to a particular event. Thus, we will later assume that the set of nodes that are likely to be sources of diffusion are known, but which ones among them are the sources of a particular diffusion event follows some probability distribution (e.g. uniform). For example, in an information network, we may know which popular news sites are typically the sources of viral news, but are uncertain as to which of these sites will be the source of a particular piece of news. Despite the broad practical relevance of the diffusion network modification problems, existing results are very limited, and lack in either formal optimality guarantees or in algorithmic efficiency.

Related work. Under the SIR model, some positive network optimization results exist: Tong et al. [22] address the edge deletion (addition) problem by approximately minimizing (maximizing) the eigenvalue of the adjacency matrix. In addition, methods have been designed to optimize surrogates for diffusion spread under SIR [8, 20]. Instead of maximizing/minimizing the spread of substances directly,

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623704 .

these methods typically optimize a static property of the network, in the hope of optimizing diffusion. For instance, Schneider et al. [20] proposed "betweenness centrality" as a heuristic for immunizing nodes or removing edges under the SIR model, while "degree centrality" was adopted in [8] to protect against virus propagation in email networks.

Under the IC model, existing results are negative: Sheldon et al.[21] study the problem of node addition to maximize spread, and provide a counter-example showing that the objective function is *not* submodular. Thus, they resort to a principled but expensive approach based on sample average approximation and mixed integer programming, which provides provable optimality guarantees but cannot scale to large networks. Bogunovic [1] addresses the node deletion problem. For the edge deletion problem under the IC model, Kimura et al.[14] apply the greedy algorithm used by Kempe et al. [12] for source node selection, but do not provide any approximation guarantees. We note that the edge deletion objective that we consider is not supermodular under IC; a simple counter-example is provided in the extended version of this paper.

Network optimization under the LT model is still largely unexplored, and this paper seeks to fill that gap. The greedy approach has been used in [13] to delete edges under the LT model, albeit without any analysis of the supermodularity of the objective, nor formal approximation guarantees. More recently, Kuhlman et al. [15] propose heuristic algorithms for edge removal under a simpler deterministic variant of the LT model. In contrast, we describe *principled algorithms* for edge deletion and addition by exploiting supermodularity of the objectives under the more general *stochastic* LT model. Most related are [3, 9], where node deletion under the "competitive" LT model (a variant of stochastic LT) is addressed using supermodularity. The node deletion problem can be considered a special case of the "influence blocking maximization" problem described in [3, 9]. However, the supermodularity results in those works are limited to node deletion, whereas our theoretical framework leads to supermodularity results for edge deletion and addition, node deletion and addition.

Our contributions. In this paper, we will address network topology optimization for diffusion under the *linear threshold model*. We will focus on two network modification problems involving the deletion of edges for minimizing spread and the addition of edges for maximizing spread [1] :

> **Edge deletion problem:** Given a set of source nodes X, find a set of k edges to remove s.t. the spread of a certain substance is minimized.
>
> **Edge addition problem:** Given a set of source nodes X, find a set of k edges to add s.t. the spread of a certain substance is maximized.

Proof techniques used in the source node selection literature are inadequate for our more complex network modification problems. Instead, we propose a novel theoretical framework (Section 3) that yields surprising results, namely that the objective function in both problems is *supermodular*, a property that has positive algorithmic implications. In particular, minimization of a supermodular function under cardinality constraints, although typically an NP-hard problem, admits a greedy algorithm with approximation guarantees [18]. Similarly, cardinality-constrained supermodular maximization has recently been shown to admit a simple modular approximation scheme [11]. Our finding, combined with these combinatorial optimization results allows, for the first time, the design of efficient diffusion-aware algorithms with approximation guarantees for the network modification problems under the LT model, filling a gap in the existence research literature on this topic.

We address several challenges in transforming the two general supermodular approximation algorithms into efficient and practical approaches for our setting. Directly implementing the supermodular optimization algorithms is impractical, since evaluating the objective function given a set of source nodes is #P-hard in general [4]. We exploit the correspondence between the LT model and the "live-edge graph" construction [12], and estimate the objective function using a sample of random live-edge graphs. Nevertheless, a naive application of the supermodular approximation schemes to the sample of random live-edge graphs will result in runtime quadratic in the network size, an approach that does not scale to modern problems with millions of nodes and edges. To tackle this issue, we design two data structures, the descendant-count trees for the edge deletion problem, and the neighbor-counting graphs for the edge addition problem, in order to support approximate evaluation of the objective function. These data structures can be *constructed in time linear in the network size*, and *queried in constant time*, allowing us to scale the supermodular optimization algorithms to networks with millions of nodes.

Finally, we evaluate our algorithms on both synthetic and real-world diffusion networks and compare the quality of the solutions to scalable alternative approaches, based on optimizing structural properties of the networks. Our algorithms can lead to as large as 10-20% additional efficacy for edge deletion, and up to 100% for edge addition, compared to the other approaches. In terms of scalability, our algorithms can scale to large networks with millions of nodes and edges.

2. LINEAR THRESHOLD MODEL

In this section, we will provide background on the linear threshold (LT) model for diffusion processes [12], which will be at the center of this study. This model is well-suited for representing threshold behavior, where entities in a network have a "tipping point" in terms of the fraction of neighboring nodes that have to adopt the diffusing substance, beyond which they would adopt it themselves. For instance, in a social setting, an individual may refrain from voicing his opinion, until a significant fraction (e.g. half) of his friends have voiced a similar opinion.

2.1 Cascade Generative Process

Underlying the LT model is a weighted directed graph $G = (V, E, w)$, called the *influence graph*, where V is a set of n nodes and E is a set of m directed edges, and $w : V \times V \to [0, 1]$ is a weight function. For edges $(u, v) \notin E$ we ignore the value of $w(u, v)$. We further require that $\sum_{u:(u,v)\in E} w(u, v) \leq 1$ for each node v. Starting from a source node (or an initially activated node) $S_0 = \{a\}$, a *cascade* then proceeds in discrete time steps $t = 0, 1, 2, \ldots$ as follows: (1) at $t = 0$, every node v first independently selects a threshold θ_v uniformly at random in the range $[0, 1]$, reflecting the uncertainty as to users' true thresholds; (2) subsequently, an inactive node v becomes activated at time $t + 1$ if $\sum_{u:u\in S_t,(u,v)\in E} w(u, v) \geq \theta_v$ where S_t is the set of nodes activated up to time t; (3) finally, the process terminates if no more activations are possible.

[1] One can also consider the analogous node deletion and addition problems, and our theoretical and algorithmic results on the finer-scale edge problems can be extended trivially.

Given an influence graph $G = (V, E, w)$, the *influence function* $\sigma(a, G)$ of a source node $a \in V$ is defined as the expected number of active nodes at the end of the diffusion process, $\sigma(a, G) = \mathbb{E}_{\theta_v}[|S_\infty|]$, where the expectation is taken with respect to the randomness of the node thresholds θ_v.

2.2 Live-Edge Graph Representation

Kempe et al. [12] showed that the influence function can be computed in an alternative way using what is referred to as "live-edge graphs", a construction that is more amenable to mathematical analysis. More specifically, a random live-edge graph X is generated as follows: Independently for each node $v \in V$, at most one of its incoming edges is selected with probability $w(u, v)$, and no edge is selected with probability $1 - \sum_{u:(u,v)\in E} w(u, v)$. Note that the set of nodes of X is equal to V, the set of "live" (or sampled) edges E_X of X is a subset of E, *i.e.*, $E_X \subseteq E$, and these edges are unweighted. Then, the influence function can be alternatively computed as

$$\sigma(a, G) = \mathbb{E}_X[r(a, X)] = \sum\nolimits_{X\in\mathcal{X}_G} \Pr[X|G] \cdot r(a, X), \quad (1)$$

where $\mathcal{X}_G$ is the space of all possible live-edge graphs based on G, $\Pr[X|G]$ is the probability of sampling a particular live-edge graph X, and $r(a, X)$ is the set of all reachable nodes in X from source a. If we define function

$$p(v, X, G) := \begin{cases} w(u, v), & \text{if } \exists u : (u, v) \in E_X \\ 1 - \sum_{u:(u,v)\in E} w(u, v), & \text{otherwise} \end{cases}$$

which is the probability of the configuration of incoming edges for node v in X; then, the probability of a particular live-edge graph X is

$$\Pr[X|G] = \prod\nolimits_{v\in V} p(v, X, G). \quad (2)$$

3. SPACE OF LIVE-EDGE GRAPHS

We will show that the objective functions of the edge deletion and addition problems are supermodular in the next section. However, the proofs require some properties beyond those used in [12]. Thus, we will first give some intuition as to why these properties are needed. We will denote the modification of an influence graph G by deleting or adding a set of edges S respectively by

$$G \setminus S := (V, E \setminus S, w), \quad \text{and} \quad G \cup S := (V, E \cup S, w).$$

Consider the monotonicity of the influence function as defined in Eq. (1), with respect to the set of source nodes A: $\sigma(A \cup a, G) - \sigma(A, G)$ is a sum over $\mathcal{X}_G$, and so proving that $\Pr[X|G] \cdot r(A \cup a, X) - \Pr[X|G] \cdot r(A, X) \geq 0$ for the single live-edge graph $X \in \mathcal{X}_G$ suffices to prove monotonicity, in this case. In contrast, consider the monotonicity of the influence function, with respect to the set of edges in the graph: $\sigma(a, G \setminus S) - \sigma(a, G \setminus (S \cup \{e\}))$. The issue here lies in that the former function sums over $\mathcal{X}_{G\setminus S}$, whereas the latter sums over $\mathcal{X}_{G\setminus(S\cup\{e\})}$: since the set of live-edge graphs and the associated probabilities involved in the computation of the influence function will change as edges are deleted, it is not obvious that this function is monotone at all. Similar difficulties apply to proving supermodularity.

In this section, we will prove four properties related to the space of live-edge graphs, which will form the basis of our later analysis. More specifically, we are concerned with

- a subset S of the edge set E in the original influence graph G, *i.e.*, $S \subseteq E$; and
- two distinct edges $e = (u, v) \in E \setminus S$ and $g = (u', v') \in E \setminus S$ outside S, where v may or may not equal v'.

Deleting a set S from E will result in a new influence graph $G \setminus S$, which will generate a new space of live-edge graphs, $\mathcal{X}_{G\setminus S}$, and the associated live-edge graph probabilities, $\Pr[X|G \setminus S]$. Furthermore, we will divide the space $\mathcal{X}_{G\setminus S}$, according to the edge e, into three disjoint partitions (see Fig. 1(left)):

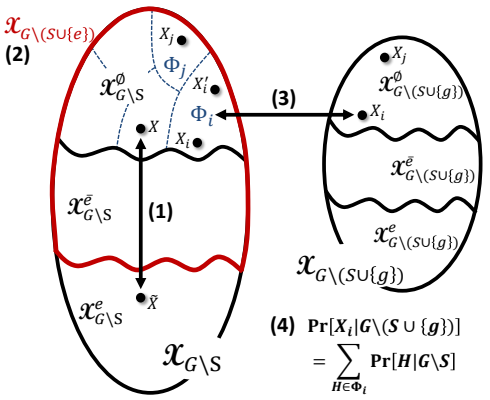


Figure 1: Four properties of the space of live-edge graphs: (1) within space mapping, (2) space inclusion, (3) across space mapping, and (4) across space probability mapping.

- $\mathcal{X}^{e}_{G\setminus S}$, the set of live-edge graphs where incoming edge $e = (u, v)$ is selected for node v;
- $\mathcal{X}^{\bar{e}}_{G\setminus S}$, the set of live-edge graphs where a different incoming edge $\bar{e} = (y, v)$ is selected for node v;
- $\mathcal{X}^{\emptyset}_{G\setminus S}$, the set of live-edge graphs where no incoming edge is selected for v.

Note that the probabilities of a live-edge graph X common to both $\mathcal{X}_G$ and $\mathcal{X}_{G\setminus\{e\}}$ may or may not change. Specifically, if node v has another incoming edge $g = (u', v)$ in X, then

$$\Pr[X|G] - \Pr[X|G \setminus \{e\}] = 0.$$

Otherwise, if node v has no incoming edge in X, then

$$\Pr[X|G] - \Pr[X|G \setminus \{e\}] = -w(u, v) \prod_{v'\neq v} p(v', X, G) \quad (3)$$

since all terms in Eq. (2) for $\Pr[X|G]$ and $\Pr[X|G \setminus \{e\}]$ concerning nodes $v' \neq v$ are exactly the same, and

$$p(v, X, G) = 1 - \sum_{u':(u',v)\in E\setminus\{e\}} w(u', v) - w(u, v)$$
$$= p(v, X, G \setminus \{e\}) - w(u, v).$$

We establish the following four relations between the spaces of live-edge graphs, and defer their proofs to the appendix. See Fig. 1 for illustrations.

3.1 Within Space Mapping

Our first result establishes a one-to-one mapping between the elements in partition $\mathcal{X}^{e}_{G\setminus S}$ and $\mathcal{X}^{\emptyset}_{G\setminus S}$.

PROPOSITION 1. *For every live-edge graph $X \in \mathcal{X}^{\emptyset}_{G\setminus S}$, there exists a corresponding live-edge graph $\widetilde{X} \in \mathcal{X}^{e}_{G\setminus S}$, and vice versa. If $X = (V, E_X)$, then $\widetilde{X} = (V, E_X \cup \{e\})$.*

3.2 Space Inclusion

Our second result relates partitions $\mathcal{X}^{\bar{e}}_{G\setminus S}$ and $\mathcal{X}^{\emptyset}_{G\setminus S}$ of the space $\mathcal{X}_{G\setminus S}$, to the space $\mathcal{X}_{G\setminus(S\cup\{e\})}$. We note that this second space of live-edge graphs, $\mathcal{X}_{G\setminus(S\cup\{e\})}$, is generated from the influence graph $G \setminus (S \cup \{e\})$ with an additional edge e deleted from $G \setminus S$. This result also shows that the space $\mathcal{X}_{G\setminus(S\cup\{e\})}$ is included in the space $\mathcal{X}_{G\setminus S}$.

PROPOSITION 2. *$\mathcal{X}_{G\setminus(S\cup\{e\})} \subseteq \mathcal{X}_{G\setminus S}$, and furthermore $\mathcal{X}_{G\setminus(S\cup\{e\})} = \mathcal{X}^{\bar{e}}_{G\setminus S} \cup \mathcal{X}^{\emptyset}_{G\setminus S}$.*

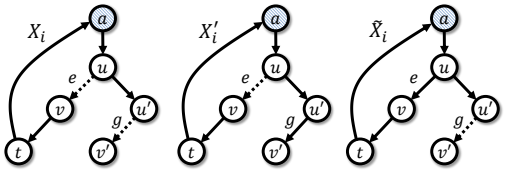


Figure 2: A dashed line means the edge has been deleted.

3.3 Across Space Mapping

Our third result further partitions $\mathcal{X}^{\emptyset}_{G\setminus S}$ into a collection of sets $\{\Phi_i\}$, and establishes a one-to-one mapping between Φ_i and element X_i in the space $\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$, where $g=(u',v')$ is an edge in $E\setminus S$ with $v'\neq v$.

PROPOSITION 3. *Let $t=|\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}|$, then $\mathcal{X}^{\emptyset}_{G\setminus S}$ can be partitioned into t sets $\{\Phi_i\}_{i=1}^{t}$ such that, for every Φ_i, there exists a corresponding $X_i\in\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$, and vice versa.*

3.4 Across Space Probability Mapping

Our fourth result relates the probability of the partition $\Phi_i\subseteq\mathcal{X}^{\emptyset}_{G\setminus S}$ to the probability of the corresponding live-edge graph $X_i\in\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$. This result is a sequel to the across space mapping property in the last section. Essentially, we show that the sum of the probabilities of the elements in Φ_i is equal to the probability of X_i.

PROPOSITION 4. *For every $\Phi_i\subseteq\mathcal{X}^{\emptyset}_{G\setminus S}$ and its associated $X_i\in\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$, $\Pr[X_i|G\setminus(S\cup\{g\})]=\sum_{H\in\Phi_i}\Pr[H|G\setminus S]$.*

4. NETWORK OPTIMIZATION

In this section, we will prove a set of results for the LT model, namely that the objective functions are *supermodular* in both the edge deletion and the edge addition problems. These results will form the basis for our algorithm design.

A set function $f:2^E\mapsto\mathbb{R}$ defined over the power set 2^E of a set E is called *supermodular* iff $\forall S\subseteq T\subset E,\forall e\in E\setminus T$

$$f(S\cup\{e\})-f(S)\leq f(T\cup\{e\})-f(T). \tag{4}$$

Intuitively, for a monotone increasing supermodular function f, the marginal gain of adding a new element e to a set T is greater than the gain of adding e to any subset S of T. This property is referred to as the *increasing differences* property, as opposed to *diminishing returns* in the case of a submodular function. If f is a monotone decreasing function (as will be the case when we consider deleting edges or nodes), then the marginal loss in adding e to T would be smaller than that of adding e to S.

We define the *susceptibility* of an influence graph G to a set of potential sources A as $\sum_{a\in A}\sigma(a,G)$, which is the sum of the influence function for each node a. Intuitively, one can think of each node $a\in A$ as having equal probability of being the source, and the susceptibility of G as the expected value of the influence function with respect to the randomness of picking an a from A. Our definition of susceptibility can also be generalized to the case where each node a has a different probability of being the source. In this case, all our subsequent theorems would still hold. Furthermore, we assume that the size of the source set is only poly-logarithmic in the total number of nodes in the network, *i.e.*, $|A|\ll|V|$.

4.1 Edge Deletion Problem

In this problem, given an influence graph $G=(V,E,w)$ and a set of sources A, we want to delete a set of edges S^* of size k from G such that the susceptibility of the resulting influence graph is minimized. That is

$$S^* := \operatorname*{argmin}_{S\subseteq E:|S|=k}\sum_{a\in A}\sigma(a,G\setminus S), \tag{5}$$

where the objective function is a set function over the edges S to be deleted. We will show that each $\sigma(a,G\setminus S)$ is a monotonically decreasing and supermodular function of S, and hence their positive sum, $\sum_{a\in A}\sigma(a,G\setminus S)$, also is.

Monotonicity. In this section, we prove that $\sigma(a,G\setminus S)$ is a monotonically decreasing function of S. We will use the within space mapping property in Prop. 1 and the space inclusion property in Prop. 2 to prove the following result:

THEOREM 5. *$\sigma(a,G\setminus S)$ is a monotonically decreasing function of the set of edges S to be deleted.*

PROOF. Given the influence graph $G=(V,E,w)$, we need to show that for any set $S\subseteq E$ and $e=(u,v)\in E\setminus S$:

$$\sigma(a,G\setminus S)-\sigma(a,G\setminus(S\cup\{e\}))\geqslant 0.$$

Using the fact that the space $\mathcal{X}_{G\setminus S}$ is partitioned into three sets, $\mathcal{X}^{e}_{G\setminus S}$, $\mathcal{X}^{\bar{e}}_{G\setminus S}$ and $\mathcal{X}^{\emptyset}_{G\setminus S}$, and the space $\mathcal{X}_{G\setminus(S\cup\{e\})}$ is partitioned into two sets $\mathcal{X}^{\bar{e}}_{G\setminus S}$ and $\mathcal{X}^{\emptyset}_{G\setminus S}$ (space inclusion property in Prop. 2), we can write the difference as

$$\begin{aligned}&\sigma(a,G\setminus S)-\sigma(a,G\setminus(S\cup\{e\})) \qquad (6)\\ =&\sum\nolimits_{X\in\mathcal{X}^{e}_{G\setminus S}}\Pr[X|G\setminus S]\cdot r(a,X)\\ +&\sum\nolimits_{X\in\mathcal{X}^{\emptyset}_{G\setminus S}}(\Pr[X|G\setminus S]-\Pr[X|G\setminus(S\cup\{e\})])\cdot r(a,X)\\ +&\sum\nolimits_{X\in\mathcal{X}^{\bar{e}}_{G\setminus S}}(\Pr[X|G\setminus S]-\Pr[X|G\setminus(S\cup\{e\})])\cdot r(a,X)\end{aligned}$$

Recall that $e=(u,v)$. We will simplify the last two summands in the above equation using the following two facts:

- For $X\in\mathcal{X}^{\emptyset}_{G\setminus S}$ based on Eq. (3): $\Pr[X|G\setminus S]-\Pr[X|G\setminus(S\cup\{e\})]=-w(u,v)\prod_{v'\neq v}p(v',X,G\setminus S)$.
- For $X\in\mathcal{X}^{\bar{e}}_{G\setminus S}$, the probability is the same, $p(v,X,G\setminus S)=p(v,X,G\setminus(S\cup\{e\}))=w(\bar{e})$.

Then Eq. (6) simplifies to $\sum_{X\in\mathcal{X}^{e}_{G\setminus S}}\Pr[X|G\setminus S]\cdot r(a,X)$ $+\sum_{X\in\mathcal{X}^{\emptyset}_{G\setminus S}}-w(u,v)\prod_{v'\neq v}p(v',X,G\setminus S)\cdot r(a,X)$. Since any $\widetilde{X}\in\mathcal{X}^{e}_{G\setminus S}$ has probability $\Pr[\widetilde{X}|G\setminus S]=w(u,v)\cdot\prod_{v'\neq v}p(v',\widetilde{X},G\setminus S)$, then using Prop. 1 to match $\widetilde{X}\in\mathcal{X}^{e}_{G\setminus S}$ to $X\in\mathcal{X}^{\emptyset}_{G\setminus S}$, we have that Eq. (6) is equal to

$$\sum\nolimits_{X\in\mathcal{X}^{\emptyset}_{G\setminus S}}\Pr[\widetilde{X}|G\setminus S]\cdot\Big(r(a,\widetilde{X})-r(a,X)\Big). \tag{7}$$

Since the live-edge graph $\widetilde{X}$ has one more edge than X, clearly $r(a,\widetilde{X})-r(a,X)\geqslant 0$, which completes the proof. □

Supermodularity. We will use the across space mapping property in Prop. 3 (also Prop. 4) and the probability mapping property in Prop. 1 to prove the following result:

THEOREM 6. *The function $\sigma(a,G\setminus S)$ is a supermodular function of the set of edges S to be deleted.*

PROOF. Let $t=|\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}|$, then using the across space mapping property in Prop. 3, we can partition $\mathcal{X}^{\emptyset}_{G\setminus S}$ into t sets $\{\Phi_i\}_{i=1}^{t}$ and rewrite Eq. (7) in the proof of Thm. 5 as:

$$\sum\nolimits_{i=1}^{t}\sum\nolimits_{X\in\Phi_i}\Pr[\widetilde{X}|G\setminus S]\cdot\Big(r(a,\widetilde{X})-r(a,X)\Big) \tag{8}$$

Using similar reasoning to that in Eq. (7) in the proof of Thm. 5 for $G\setminus(S\cup\{g\})$, we have

$$\begin{aligned}&\sigma(a,G\setminus(S\cup\{g\}))-\sigma(a,G\setminus(S\cup\{g,e\}) \qquad (9)\\ =&\sum\nolimits_{X\in\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}}\Pr[\widetilde{X}|G\setminus(S\cup g)]\cdot\Big(r(a,\widetilde{X})-r(a,X)\Big)\end{aligned}$$

Then we need only compare Eq. (9) and (8) term by term for each $X_i \in \mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}, i = 1,\ldots,t$. Clearly, when $\Phi_i = \{X_i\}$, the terms from the two equations are equal. When $\Phi_i = \{X_i, X_i'\}$, we need to show

$$\Pr[\widetilde{X}_i|G\setminus S]\cdot\Big(r(a,\widetilde{X}_i)-r(a,X_i)\Big) + \Pr[\widetilde{X}_i'|G\setminus S]\cdot\Big(r(a,\widetilde{X}_i')-r(a,X_i')\Big) \geqslant \Pr[\widetilde{X}_i|G\setminus(S\cup\{g\})]\cdot\Big(r(a,\widetilde{X}_i)-r(a,X_i)\Big) \quad (10)$$

Based on the probability mapping property in Prop. 4, we have $\Pr[\widetilde{X}_i|G\setminus(S\cup\{g\})] = \Pr[\widetilde{X}_i|G\setminus S] + \Pr[\widetilde{X}_i'|G\setminus S]$. Then to establish Eq. (10), it suffices to show that

$$r(a,\widetilde{X}_i')-r(a,X_i') \geqslant r(a,\widetilde{X}_i)-r(a,X_i).$$

Recall that $X_i' = (V, E_{X_i}\cup\{g\})$. Since live-edge graphs are constructed in a way that each node has at most one incoming edge, each reachable node y has a unique path from the source node a to node y. Furthermore: (1) a reachability path in $\widetilde{X}_i$ is clearly also present in $\widetilde{X}_i'$, hence if removing edge $e=(u,v)$ from $\widetilde{X}_i$ results in unreachability of some nodes in X_i then those same nodes become unreachable when removing e from $\widetilde{X}_i'$; (2) removing edge e from $\widetilde{X}_i'$ may disconnect some additional nodes whose paths from the source a include edge g. Therefore the reduction in reachable nodes when removing edge e from $\widetilde{X}_i'$ is the same or larger than the reduction when removing edge e from $\widetilde{X}_i$. This completes the proof. □

4.2 Edge Addition Problem

In this section, given a *partial* influence graph $G'(V,E',w)$ and a larger *potential* influence graph $G=(V,E,w)$ with $E'\subseteq E$, we want to add to G' a set of edges S^* of size k from $E\setminus E'$ such that the resulting susceptibility is maximized:

$$S^* := \operatorname*{argmax}_{S'\subseteq E\setminus E':|S'|=k} \sum_{a\in A}\sigma(a,G'\cup S'), \quad (11)$$

where the objective function is a set function over the edges S' to be added. We will show that each $\sigma(a,G'\cup S')$ is monotone and supermodular, and hence their positive combination, $\sigma(A,G'\cup S')$, is also monotone and supermodular.

THEOREM 7. *The function $\sigma(a,G'\cup S')$ is a monotone and supermodular function of the set of S' edges to be added.*

PROOF. We will prove the results by relating the objective function to that in the edge deletion problem and then apply the results from the edge deletion problem. More specifically, let $S'\subseteq T'$ and $e\in E\setminus T'$. If we define $S = E\setminus(S'\cup E'\cup\{e\})$, then

$$\sigma(a,G'\cup(S'\cup\{e\})) = \sigma(a,G\setminus S)$$
$$\sigma(a,G'\cup S') = \sigma(a,G\setminus(S\cup\{e\})),$$

since $G'\cup(S'\cup\{e\}) = G\setminus S$ and $G'\cup S' = G\setminus(S\cup\{e\})$. Similarly, if we define $T = E\setminus(T'\cup E'\cup\{e\})$, then

$$\sigma(a,G'\cup(T'\cup\{e\})) = \sigma(a,G\setminus T)$$
$$\sigma(a,G'\cup T') = \sigma(a,G\setminus(T\cup\{e\})).$$

Note that $S'\subseteq T'$ implies that $T\subseteq S$. Then we apply the supermodularity of $\sigma(a,G\setminus S)$ as a function of the edges S to be deleted in Thm. 6, and obtain

$$\sigma(a,G'\cup(S'\cup\{e\})) - \sigma(a,G'\cup S') \leqslant \sigma(a,G'\cup(T'\cup\{e\})) - \sigma(a,G'\cup T'),$$

which completes the proof. □

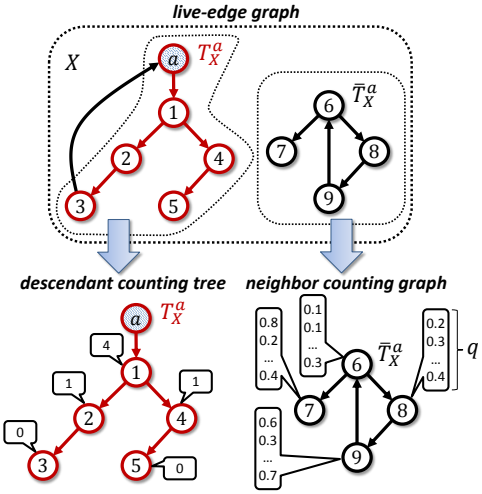


Figure 3: Illustration of a live-edge graph X, the tree T_X^a induced by BFS rooted at a, the part of the graph $\overline{T}_X^a$ which is the complement to T_X^a. For the edge deletion problem, we build a descendant counting tree data structure for each live-edge tree T_X^a where each node stores its number of descendants. For the edge addition problem, we build a neighbor-counting graph data structure for each complement $\overline{T}_X^a$, where each node stores a list of q least labels $\{l_i^*(v)\}$ obtained over q random labelings.

5. SCALABLE ALGORITHMS

Given that the edge deletion and addition problems are supermodular, we can, in principle, solve the network topology optimization problems using the state-of-the-art supermodular optimization algorithms. However, there remain great challenges in scaling these algorithms up to diffusion networks with millions of nodes. First, supermodular optimization requires evaluating the influence function many times. The problem of computing the influence function $\sigma(a,G)$ exactly has been shown to be #P-Hard [4]. Thus there is a need to design methods to approximately compute the influence function in near-linear time. To tackle this problem, we will estimate $\sigma(a,G)$ using *empirical averaging* (EA) over a *fixed* set of live-edge graphs, pre-sampled using the LT live-edge graph generation process described in section 2.2. That is

$$\sigma(a,G) \approx \widehat{\sigma}(a,G) := \frac{1}{|\mathcal{L}|}\cdot\sum_{X_i\in\mathcal{L}} r(a,T_{X_i}^a) \quad (12)$$

where $\mathcal{L} = \{X_i\}_{1\leq i\leq l}$ is the set of sampled *live-edge graphs* from G. Recall $T_{X_i}^a$ is the tree rooted at a induced from X_i.

Second, typically, the marginal change of the influence function for each candidate edge needs to be computed. This imposes the additional requirement that each marginal change computation has to be nearly constant-time to handle the large number of candidate edges. To address these challenges, we will design an efficient descendant-counting tree data structure for the edge deletion problem, and use it as part of a greedy algorithm to minimize the supermodular objective function. For the edge addition problem, we will employ an efficient randomized neighbor-counting graph data structure, and use it inside a modular approximation algorithm to maximize the supermodular objective function. Fig. 3 illustrates the data structures. In both cases, time and space complexities are linear in the network size.

5.1 Edge Deletion

It is easy to see that the empirical average influence function $\widehat{\sigma}(a,G\setminus S)$ under edge deletion is also supermodular and monotonically decreasing. To solve the problem at hand, we

adopt a simple greedy approach: at each iteration, given the current solution S_t, add to the solution the element e with the largest *marginal loss* $\Delta(e|S_t)$ defined using Eq. (12)

$$\frac{1}{\mathcal{L}} \sum_{a \in A} \sum_{X_i \in \mathcal{L}} r(a, T^a_{X_i} \setminus S_t) - r(a, T^a_{X_i} \setminus (S_t \cup \{e\})) \quad (13)$$

where $T^a_{X_i} \setminus S_t$ means deleting edges S_t from tree $T^a_{X_i}$. Based on this expansion, we notice that the edge with largest marginal loss is the edge whose deletion results in the largest decrease in the average number of descendants over all source nodes and the set of induced live-edge trees T^a_X. Note that we use the terminology "marginal loss" rather than "marginal gain" because our objective function is monotone *decreasing*, hence $\Delta(e|S_t)$ measures the marginal loss resulting from removing e after edges S_t have been removed.

Naïvely applying the greedy algorithm is computationally intensive and will not scale to networks with millions of nodes. Basically, at iteration t, for every edge $e \in E \setminus S_t$ and every T^a_X, we need to compute $\Delta(e|S_t)$ by performing a Breadth-First-Search (BFS) traversal from the source a, and count the number of nodes reachable in $T^a_{X_i} \setminus S_t$ and $T^a_{X_i} \setminus (S_t \cup \{e\})$. It is easy to see that such an approach will lead to an $O(|V|^2 + |V||E|)$ complexity algorithm: BFS is $O(|V| + |E|)$ and we need to check $O(|E|)$ edges in $E \setminus S_t$. Such quadratic dependence on the network size motivates us to design a more efficient solution.

Scaling Up. Can we avoid the many BFS traversals? To answer this question, we first make the following observation

OBSERVATION 1. *Given an edge $e = (u, v)$ to be deleted where v is reachable from the source a, the marginal loss $\Delta(e|S_t)$ can be computed as*

$$r(a, T^a_X \setminus S_t) - r(a, T^a_X \setminus (S_t \cup \{e\})) = r(v, T^a_X \setminus S_t) + 1 \quad (14)$$

This observation implies that, if we can compute $r(v, T^a_X)$ for all $v \in V$ in the original live-edge tree T^a_X, we can then compute the marginal gain of each edge e efficiently.

Can we compute the number of descendant, $r(v, T^a_X)$, efficiently, for *all* $v \in V$? Fortunately, since we are dealing with trees of at most $|V|$ edges each, this can be done in time $O(|V|)$ using a single BFS traversal. More specifically, after initializing $r(v, T^a_X) = 0,\ \forall v \in V$,

1. Perform a BFS starting from the source a of T^a_X, adding each traversed edge e to a stack H; at the end of the BFS, the top of the queue is the last edge traversed.
2. While stack H is not empty, pop edge $e = (u, v)$ and increment $r(u, T^a_X)$ by $r(v, T^a_X) + 1$.

The correctness of the above procedure is easy to verify: the number of descendants of a node is equal to the sum of the number of descendants of its children, plus the number of children it has, which is exactly what we are computing.

Suppose we have already maintained the descendant counts $r(v, T^a_X \setminus S_t)$ for all node $v \in V$. Then after deleting edge $e = (u, v)$, there are two types of nodes for which need to update the descendant counts: the ancestors u' of node v, and the nodes that have become unreachable. For the former, we update their descendant counts by subtracting out the number of descendants of node v plus 1. Similar to Eq. (14),

$$r(u', T^a_X \setminus (S_t \cup \{e\})) = r(u', T^a_X \setminus S_t) - r(v, T^a_X \setminus S_t) - 1.$$

For the latter, we simply set their descendant counts to zero, *i.e.*, $r(u', T^a_X \setminus (S_t \cup \{e\})) = 0$. Last, the marginal loss of each edge can also be updated according to Eq. (14).

Overall Algorithm: GREEDYCUTTING is summarized in Algorithm 1. It first samples live-edge graphs and obtains the corresponding live-edge trees for the input sources A. Lines 4–12 compute the initial descendant counts variables $r(u, T^a_X)$ for each node u and each T^a_X, and the edge marginal loss variables $\Delta(e)$ for all edges in E. For each iteration, line 15 adds to the solution set the edge with largest marginal loss, and lines 16–27 locally update the descendant count variables for nodes, and marginal loss variables for edges. Finally, the solution set S^* is returned.

If we assume the number of source nodes to be poly-logarithmic in $|V|$, then Algorithm 1 has computational complexity $O(k|\mathcal{L}||V|)$, which is linear (up to poly-logarithmic factors) in the size of the network. As for space complexity, our main data structures store the node descendant counts for each induced live-edge tree on one hand, and the marginal losses of the edges on the other, requiring space of complexity $O(|E| + |V||\mathcal{L}|)$, linear in the network size.

LEMMA 8. *Let $S^* \in \operatorname{argmin}_{S \subseteq E : |S| = k} \sum_{a \in A} \sigma(a, G \setminus S)$, and $h(S) = \sum_{a \in A} \sigma(a, G) - \sigma(a, G \setminus S)$, the reduction in influence when S is deleted from E. Let α be the approximation factor for influence estimation by EA. The solution $\widehat{S}$ returned by* GREEDYCUTTING *satisfies*

$$h(\widehat{S}) \geqslant (1 - 1/e - \alpha) h(S^*)$$

PROOF. Straightforward based on [18]. □

Algorithm 1: GREEDYCUTTING

Input: Influence Graph $G(V, E, w)$, Sources A, k
Output: Edges S^*

1 Sample a set of live-edge graphs $\mathcal{L} = \{X\}$ from G
2 Obtain the set of induced live-edge trees $\{T^a_X\}$ from $\mathcal{L}$
3 Initialize $\Delta(e) = 0$ for all $e \in E$, $r(u, T^a_X) = 0$ for all $u \in V$ and T^a_X
4 **for** *each* T^a_X **do**
5 Initialize queue Q, stack H, $Q.enqueue(a)$, $visited = \{a\}$
6 **while** Q *is not empty* **do**
7 $s = Q.dequeue()$
8 **for** $u \in V$ *and* (s, u) *is an edge in* T^a_X **do**
9 **if** $u \notin visited$ **then**
10 $visited = visited \cup \{u\}$, $Q.enqueue(u)$, $H.push((s, u))$
11 **while** H *is not empty* **do**
12 $(u, v) = H.pop()$, $r(u, T^a_X)$ += $r(v, T^a_X) + 1$, $\Delta((u, v))$ += $r(v, T^a_X) + 1$
13 $S^* = \emptyset$
14 **for** *t=1 to k* **do**
15 $e_t = (u_t, v_t) = \operatorname{argmax}_{e \in E \setminus S^*} \Delta(e)$, $S^* = S^* \cup \{e_t\}$
16 **for** *each* T^a_X **do**
17 **if** e_t *is an edge in* T^a_X **then**
18 $s = u_t$
19 **while** s *is not the source* a **do**
20 $r(s, T^a_X)$ −= $r(v_t, T^a_X) + 1$, $\Delta((parent(s), s))$ −= $r(v_t, T^a_X) + 1$, $s = parent(s)$
21 Initialize queue Q, $Q.enqueue(u_t)$, $visited = \{u_t\}$
22 **while** Q *is not empty* **do**
23 $s = Q.dequeue()$
24 **for** $u \in V$ *and* (s, u) *is an edge in* T^a_X **do**
25 **if** $u \notin visited$ **then**
26 $visited = visited \cup \{u\}$, $Q.enqueue(u)$, $\Delta((s, u))$ −= $r(u, T^a_X) + 1$, $r(s, T^a_X) = 0, r(u, T^a_X) = 0$
27 **return** S^*

5.2 Edge Addition

We now turn to our algorithmic framework for solving the problem of adding edges. Recently, Iyer et al. [11] proposed

a simple approach for constrained submodular minimization with approximation guarantees, which we will adapt for our (analog) supermodular maximization problem. Recall that given a *partial* influence graph $G'(V, E', w)$ and a larger *potential* influence graph $G = (V, E, w)$ with $E' \subseteq E$, we want to add to G' a set of edges S^* of size k from $E \setminus E'$ such that the resulting susceptibility is maximized (Eq. (11)). The algorithm constructs a *modular lower bound* (MLB) of the objective function, and then adds edges that maximize this lower bound, instead of the original objective. That is,

$$\frac{1}{|\mathcal{L}|}\sum_{a\in A}\sum_{X_i\in\mathcal{L}} r(a, T^a_{X_i} \cup S) \geqslant \frac{1}{|\mathcal{L}|}\sum_{a\in A}\sum_{X_i\in\mathcal{L}}\sum_{e\in S} r(a, T^a_{X_i} \cup \{e\}),$$

where $\mathcal{L} = \{X_i\}_{1\leq i\leq l}$ is the set of sampled *live-edge graphs* from G', and $T^a_{X_i}$ is the tree rooted at a induced from X_i. Then, $T^a_{X_i} \cup S$ refers to the live-edge tree resulting from adding new edges S to $T^a_{X_i}$. Note that the resulting tree may allow the source a to reach some nodes originally not reachable in $T^a_{X_i}$.

The MLB approach has several nice properties. First, the modular lower bound function is simple, essentially requiring us to compute the reachability score $r(a, T^a_{X_i} \cup \{e\})$ for each candidate edge to be added. Second, maximizing the MLB for a budget k reduces to simply finding the top k edges which lead to the largest function value

$$\sum_{a\in A}\widehat{\sigma}(a, G' \cup \{e\}) = \frac{1}{|\mathcal{L}|}\sum_{a\in A}\sum_{X_i\in\mathcal{L}} r(a, T^a_{X_i} \cup \{e\}). \quad (15)$$

However, naïvely applying the MLB algorithm is computationally intensive and can not be scaled up to networks with millions of nodes. Basically, for every candidate edge e to be added and for every T^a_X, we need to compute the reachability $r(a, T^a_{X_i} \cup \{e\})$ by performing a BFS traversal from the source a, and count the number of nodes reachable in $T^a_{X_i} \cup \{e\}$. It is easy to see that such an approach will lead to an $O(|V||E|)$ complexity algorithm: BFS is $O(|V|)$ (since trees T^a_X have at most $|V|$ edges), and we need to check $O(|E|)$ edges in $E \setminus E'$. Again, we are motivated to design a more efficient solution.

Scaling Up. Can we avoid the many BFS traversals? To answer this question, we first make the following observation

OBSERVATION 2. *Given an edge $e = (u, v)$ to be added, where node v is originally not reachable from the source a, but becomes reachable with the addition of e, the reachability of a can be updated as*

$$r(a, T^a_X \cup \{e\}) = r(a, T^a_X) + w(e) \cdot (r(v, \overline{T}^a_X) + 1) \quad (16)$$

where $\overline{T}^a_X$ is the complement of T^a_X containing those nodes and edges not reachable from a.

We note that the term $r(v, \overline{T}^a_X) + 1$ is multiplied by the weight $w(e)$ of edge e to account for the *probability* of that edge being actually picked by node v in the live-edge generation process of the new influence graph $G' \cup \{e\}$. Furthermore, note that $\overline{T}^a_X$ may contain cycles.

Can we compute the number of reachable nodes, $r(v, \overline{T}^a_X)$, efficiently, for all v in $\overline{T}^a_X$? Fortunately, this problem has been extensively studied in the theoretical computer science literature as the neighborhood size-estimation problem [5], and was recently applied in the context of influence estimation for a continuous-time diffusion model [7]. We will adapt a linear-time algorithm by Cohen [5] for our problem.

We apply the algorithm to $\overline{T}^a_X$ as follows: first, we assign to each node u a label $l(u)$ drawn from the exponential distribution with parameter (mean) 1. Then, we exploit the fact that the minimum $l^*(v)$ of the set of exponential random labels $\{l(u)\}$ for nodes u reachable from v will itself be an exponential random variable, with its parameter equal to $r(v, \overline{T}^a_X)$. If we repeat the random labeling q times and obtain q such *least labels* $\{l^*_i(v)\}_{i=1}^q$, then the neighborhood size is estimated as

$$r(v, \overline{T}^a_X) \approx \frac{q-1}{\sum_{i=1}^q l^*_i(v)}. \quad (17)$$

Can we find the least labels efficiently for all nodes v given each random labeling? In fact, this can be done using a modified BFS traversal which requires time only linear in the network size. More specifically, For a given labeling, we start from the node v with the smallest label, and perform a BFS traversal in the reverse direction of the graph edges. For each node u encountered in the BFS, we set $l^*(u) = l(v)$. Once a node has been encountered in a BFS and its least label has been set, it is marked as *visited*, not only for this particular BFS, but across all subsequent BFS runs. After the BFS traversal from node v is complete, we move to the *unvisited* node with the next smallest label, and repeat the procedure iteratively until all nodes have been marked as visited. It is easy to see why this algorithm correctly assigns the appropriate least label $l^*(v)$ to each node v: since we order the BFS runs by minimum labels, and traverse the edges in reversed direction, then once a node u has been visited, we are guaranteed that the $l^*(u)$ we assign to it is the smallest, and any subsequent BFS that can reach u will have a label larger than $l^*(u)$.

Overall Algorithm: MODULARADDING is summarized in Algorithm 2: we first generate the live-edge graphs, induce the live-edge trees, and draw q labels for each node $v \in V$ from the exponential distribution with mean 1 (lines 1-5). Then, for each source node $a \in A$, we iterate over the induced live-edge trees T^a_X, collecting the estimated neighborhood size of each node v in the complement $\overline{T}^a_X$ of each such tree, by applying Cohen's algorithm (lines 7-20). After iterating over the live-edge trees, we compute the final score for each edge $e \in \mathcal{C}$ in the candidate set $\mathcal{C}$ as the sum over v's neighborhood size estimates, weighted by the edge's diffusion probability $w(e)$ (lines 21-22). Finally we sort the scores vector in descending order, and return the top k edges.

We assume the number of sources $|A|$ is poly-logarithmic in $|V|$ and hence ignored in the complexity analysis. Then Algorithm 2 has computational complexity $O(q|\mathcal{L}||V|)$ and space complexity $O(q|V|)$. This is because the Cohen's algorithm has complexity $O(q|V|)$, and is invoked $O(|\mathcal{L}|)$ times. The final sorting of the scores can be done in $O(|E \setminus E'|)$. As for space, we only require data structures of sizes linear in the number of nodes $O(q|V|)$ to hold the least labels.

LEMMA 9. *Let $S^* \in \operatorname{argmax}_{S'\subseteq E\setminus E':|S'|=k} \sum_{a\in A}\sigma(a, G' \cup S')$, and $g(S) = \sum_{a\in A}\sigma(a, G) - \sigma(a, G' \cup S')$, the difference between the influence in the potential graph G and the influence when S is added to E'. Let κ_g be the* curvature *[11] of g, and β be the approximation factor of our two estimation subroutines (1) EA and (2) Cohen's algorithm. The solution $\widehat{S}$ returned by* MODULARADDING *satisfies*

$$g(\widehat{S}) \leqslant \beta/(1-\kappa_g) \cdot g(S^*)$$

PROOF. Straightforward based on Thm. 5.4 in [11]. □

6. EXPERIMENTS AND RESULTS

We now present our experimental setting and results.

6.1 Setting

Synthetic networks. We generate three types of networks using the Kronecker graph model[2] [17], known to

[2] http://snap.stanford.edu/data/

Algorithm 2: ModularAdding

Input: $G'(V, E', w), k$, Sources A, Candidates $\mathcal{C}$
Output: Edges S^*

Sample a set of live-edge graphs $\mathcal{L} = \{X\}$ from G'
Obtain the set of induced live-edge trees $\{T_X^a\}$ from $\mathcal{L}$
for *each* $v \in V$ **do**
  for *each* $i = 1, \ldots, q$ **do**
    $l_i(v) \sim \exp(-x)$
for *each* $a \in A$ **do**
  for *each* $\overline{T}_X^a$ **do**
    for *each* $i = 1, \ldots, q$ **do**
      $visited = \emptyset$
      for *nodes* v *in* $\overline{T}_X^a$ *ordered according to* $argsort(\{l_i(v)\})$ **do**
        if $v \notin visited$ **then**
          $visited = visited \cup \{v\}$
          Initialize Q, $Q.enqueue(v)$
          while Q *is not empty* **do**
            $u = Q.dequeue()$, $l_i^*(u) = l(v)$
            $visited = visited \cup \{u\}$
            for *each parent* s *of* u *in* $\overline{T}_X^a$ **do**
              $Q.enqueue(s)$
    for *each node* v *in* $\overline{T}_X^a$ **do**
      $r(v, \overline{T}_X^a) = \dfrac{q-1}{\sum_{i=1}^{q} l_i^*(v)}$
  for *each* $e = (u, v) \in \mathcal{C}$ **do**
    $\texttt{score}(e)$ += $w(e) \cdot \sum_{X_i \in \mathcal{L}} (r(v, \overline{T}_X^a) + 1)$
$S^* = argsort(\texttt{score}, k, \text{descending})$
return S^*

Table 1: Datasets summary. Numbers outside (inside) bracket are for edge deletion (addition) experiments. Last column is for Kronecker parameter matrices.

Dataset	#Nodes	#Edges	Kronecker
CorePeriphery			[.9 .5; .5 .3]
ErdosRenyi	1M(65K)	2M (131K)	[.5 .5; .5 .5]
Hierarchical			[.9 .1; .1 .9]
HepPH	35K	420K	
Epinions	75K	509K	—
MemeTracker	1.8K	5K	

generalize a number of realistic graph models: (1) CorePeriphery, (2) ErdosRenyi and (3) Hierarchical. These three graph models have very different structural properties, allowing us to test for sensitivity to network structure.

Real-world networks. We choose three publicly available real-world datasets[2] that are amenable to diffusion processes and hence suitable for our problems: (1) HepPH: a whom-cites-whom citation network based on the Arxiv *High-energy Physics* papers over the period 1993-2003; (2) Epinions: a who-trusts-whom online social network of the consumer review site Epinions.com; (3) MemeTracker: a who-copies-from-whom network of news media sites and blogs. The statistics of the datasets are summarized in Table 1.

Assigning probabilities. Given a network $G(V, E)$, we populate the weight vector representing the probabilities on the edges E according to the LT model, as follows: for a given node $v \in V$, we draw a probability value $\widetilde{w}(u, v)$ for each edge $e = (u, v) \in E$ that is incoming into v, uniformly at random from the interval $[0, 1]$. In addition, we draw from the same interval a probability value $\overline{w}_v$ representing no infection, i.e the probability that v's infected parents fail to activate it. Since the probabilities on the edges plus the probability of no infection must sum to 1, we then normalize each probability over the sum of all the probabilities, *i.e.*, we obtain $w(u, v) = \widetilde{w}(u, v)/(\sum_{u \in V} \widetilde{w}(u, v) + \overline{w}_v)$. We apply this method for all datasets except for MemeTracker.

Table 2: Parameter values used in the experiments of Fig. 4 (1st row), and Fig. 5 (2nd row): A is the set of sources, $\mathcal{L}_{\texttt{opt}}$ is the set of live-edge graphs used by our algorithm, $\mathcal{L}_{\texttt{eval}}$ is the set of live-edge graphs used for evaluation of all algorithms and heuristics, t refers to the budget of edges deleted for which diffusion stops completely, q is the number of random labelings used in Algo. 2.

	Parameters for Experiments				
Problem	$\lvert A\rvert$	$\lvert\mathcal{L}_{\texttt{opt}}\rvert$	$\lvert\mathcal{L}_{\texttt{eval}}\rvert$	k	q
Edge Deletion	100	1,000	5,000	$[0, t]$	–
Edge Addition				$[0, 2000]$	20

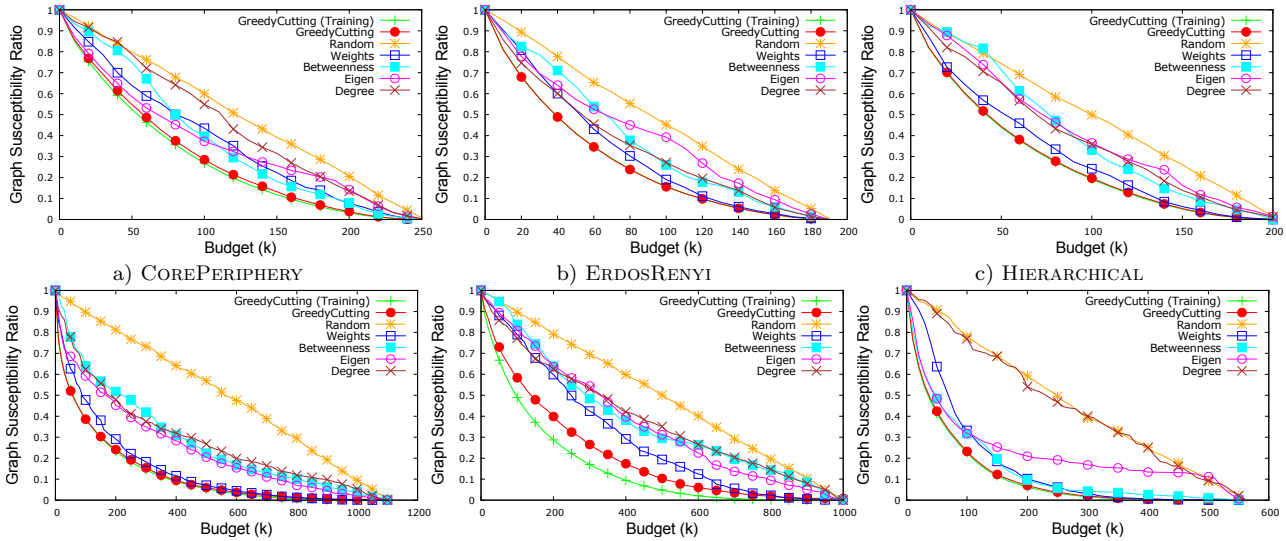
For MemeTracker, we make use of the median transmission time, also provided as part of the dataset. Let $\widetilde{t}(u, v)$ be the median transmission time between two nodes u and v, then we set $w(u, v) \propto \widetilde{t}(u, v)^{-1}$, rewarding smaller transmission times with higher diffusion probabilities, and vice versa. We assign a probability of $\overline{w}_v = 0.2, \forall v \in V$, and normalize the weights for all nodes v such that $\sum_{u \in V} w(u, v) + \overline{w}_v = 1$.

Competing heuristics. To evaluate the efficacy of the solutions provided by our algorithms, we compare against other heuristic measures that are not based on the dynamics entailed by the LT diffusion model. These heuristic strategies can be described as follows: (1) `Random`: select k edges uniformly at random from the input set of edges, (2) `Weights`: select the k edges with highest diffusion probability (weight) $w(u, v)$, (3) `Betweenness`: select k edges with highest *edge betweenness centrality* [2, 20], (4) `Eigen`: select the k edges that cause the maximum decrease (increase) in the leading eigenvalue of the network when removed from it, or added to it [22], (5) `Degree`: select the k edges whose destination nodes have the highest out-degrees [8].

6.2 Deleting Edges

We carry out each experiment as follows: given an influence graph $G(V, E, w)$, a set of source nodes A chosen uniformly at random from V, and a budget k of edges to delete, we run GreedyCutting and the five heuristics and obtain a set of edges from each. Then, for each algorithm or heuristic, we simulate the LT diffusion process by generating a set of live-edge graphs $\mathcal{L}_{\texttt{eval}}$ based on G, and then deleting the proposed set of edges S^* from all live-edge graphs in $\mathcal{L}_{\texttt{eval}}$. The efficacy of each proposed set of edges is measured by I_k, the average number of infected nodes over $\mathcal{L}_{\texttt{eval}}$. These parameters are summarized in Table 2. The budget k is increased until diffusion is no longer possible, *i.e.*, the source nodes are completely isolated.

Synthetic networks. The results are shown in Fig. 4 (a-c). First, we observe that our algorithm clearly outperforms *all five* other heuristics: for *any* budget k of edges to delete, our algorithm minimizes the graph susceptibility ratio (I_k/I_0) better than any of the heuristics for all three synthetic network types, implying that it produces good solutions *independently* of the structural properties of the input network. On the other hand, the considered heuristics perform arbitrarily good or bad, as we vary the type of synthetic network. At last, we observe that even for $|\mathcal{L}_{opt}| = 1,000$, a quantity much smaller than the typical $10,000$ used in the literature, the green and red lines are almost indistinguishable, meaning our solution generalizes well to the larger evaluation set of $5,000$ live-edge graphs.



a) CorePeriphery b) ErdosRenyi c) Hierarchical

d) HepPH e) Epinions f) MemeTracker

Figure 4: Efficacy of the edge deletion solutions provided by different algorithms. Lower is better. The x-axis refers to the budget k; the y-axis refers to the *graph susceptibility ratio*, defined in the range $[0, 1]$ as: I_k/I_0.

Real-world networks. We observe similar performance for real-world networks (Fig. 4 (d-f)). For instance, for the Epinions dataset (Fig. 4(e)), our method has decreased the graph susceptibility to 40% of its original value at $k = 200$, whereas the best performing heuristic at the same k is `Weights` with 60% (here, lower is better).

6.3 Adding Edges

The experimental procedure for evaluating our ModularAdding algorithm and other heuristics is analogous to that described in 6.2 for edge deletion. We compare our algorithm to all previously described heuristics, for the exception of the `Betweenness` heuristic, as it is not obvious how meaningful it would be to compute this metric for edges that do not initially exist in the network. Results are in Fig. 5.

Synthetic networks. Our algorithm is almost always *twice as effective* as the next best heuristic, be it `Weights` or `Degree`. This efficacy gap is consistent across all three types of networks, confirming yet again the robustness of the solutions we find to varying structural properties of networks.

Real-world networks. Similarly to the synthetic setting, our algorithm significantly outperforms all four heuristics in the real-world setting, for all three datasets. For instance, for the HepPH dataset in Fig. 5(d), the `Degree` heuristic requires adding $2,000$ edges to the set A of 100 sources in order to increase the graph susceptibility by twice its initial value (*i.e.*, at $k = 0$), whereas our algorithm increases the graph susceptibility by the same amount for $k = 200$ edges, a small fraction of 2000. This superior performance implies that our algorithm is more amenable to real-world applications, where the budget is typically very small relative to the number of nodes, possibly representing humans in a social network, blogs on the web, etc.

6.4 Scalability

Scalability is a major concern in the industrial setting. We experimentally verify the scalability of both our edge deletion and addition algorithms. All experiments were executed on a laptop with a 2.7GHz quad-core i7 CPU and 16Gb RAM. The results presented in Fig. 6 measure the runtime of our algorithms on synthetic CorePeriphery networks of increasing number of nodes, and fixed average degree of 2.

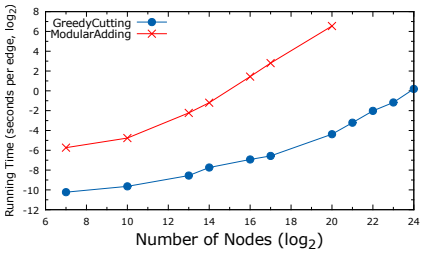


Figure 6: Runtime on synthetic core-periphery networks: for both lines, each point represents the average time in seconds per edge deleted or added. The number of sources is poly-logarithmic in the number of nodes and the number of live-edge graphs used is 100, for both problems. For ModularAdding, $q = 5$.

We vary the number of nodes, starting at $2^7 = 128$ nodes, and up to $2^{23} = 8,388,608$ nodes ($16,777,216$ edges). The experimental results show that our algorithms scale linearly in the size of the network. As expected, ModularAdding, while also having a linear scaling, is more time-consuming than GreedyCutting, due to the repeated linear-time algorithm for building the neighbor-counting graph.

7. ACKNOWLEDGEMENTS

This research was supported in part by NSF/NIH BIGDATA 1R01GM108341-01, NSF IIS1116886, NSF CAREER IIS1350983 and a Raytheon faculty fellowship to L. Song.

References

[1] I. Bogunovic. Robust protection of networks against cascading phenomena. Master's thesis, ETHZ, 2012.
[2] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2), 2001.
[3] W. Chen, L. V. Lakshmanan, and C. Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 2013.
[4] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *IEEE ICDM*, pages 88–97, 2010.
[5] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.
[6] P. Domingos and M. Richardson. Mining the network value of customers. In *ACM KDD*, pages 57–66, 2001.

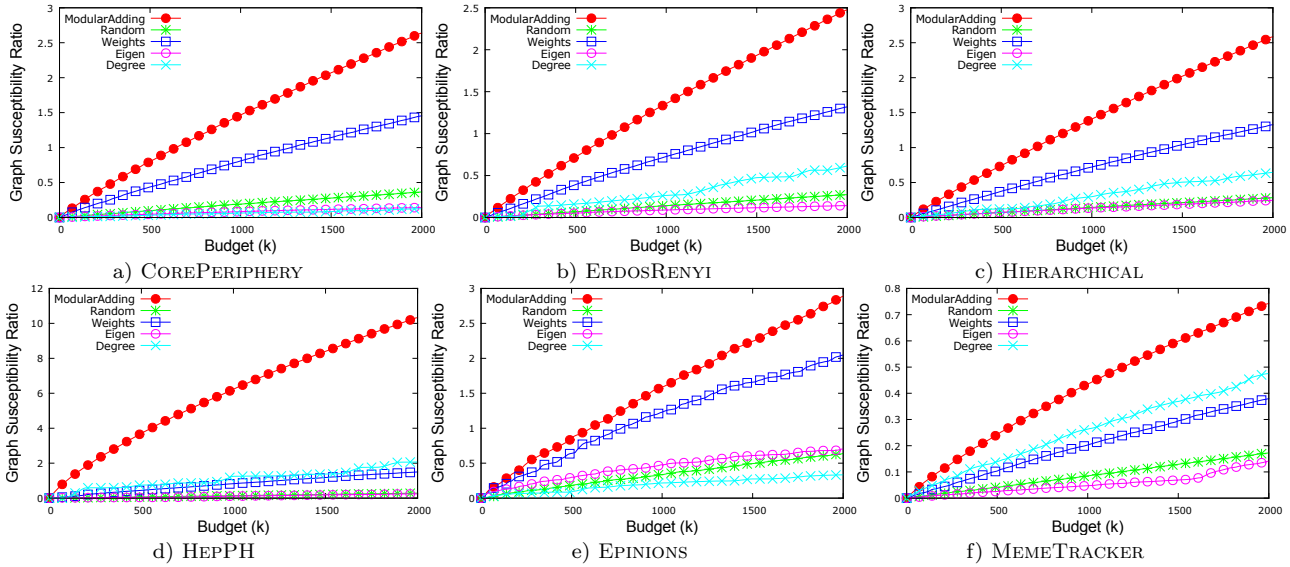


Figure 5: Efficacy of the edge addtion solutions provided by different algorithms. Higher is better. The x-axis refers to the budget of edges to add; the y-axis refers to the *graph susceptibility ratio*, defined in the range $[0, n]$ as: $(I_k - I_0)/I_0$.

[7] N. Du, L. Song, H. Zha, and M. Gomez Rodriguez. Scalable influence estimation in continuous time diffusion networks. In *NIPS*, 2013.

[8] C. Gao, J. Liu, and N. Zhong. Network immunization and virus propagation in email networks: experimental evaluation and analysis. *KAIS*, 27(2):253–279, 2011.

[9] X. He, G. Song, W. Chen, and Q. Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *SDM*, 2012.

[10] H. W. Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

[11] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, 2013.

[12] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *ACM KDD*, pages 137–146. ACM, 2003.

[13] M. Kimura, K. Saito, and H. Motoda. Solving the contamination minimization problem on networks for the linear threshold model. In *PRICAI*. 2008.

[14] M. Kimura, K. Saito, and H. Motoda. Blocking links to minimize contamination spread in a social network. *ACM TKDD*, 3(2):9, 2009.

[15] C. J. Kuhlman, G. Tuli, S. Swarup, M. V. Marathe, and S. Ravi. Blocking simple and complex contagion by edge removal. In *IEEE ICDM*, 2013.

[16] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *ACM KDD*, pages 497–506. ACM, 2009.

[17] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. 11(Feb):985–1042, 2010.

[18] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

[19] S. Peng, S. Yu, and A. Yang. Smartphone malware and its propagation modeling: A survey. *IEEE Communications Surveys Tutorials*, PP(99):1–17, 2013.

[20] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann. Suppressing epidemics with a limited amount of immunization units. *Physical Review E*, 84(6), 2011.

[21] D. Sheldon, B. Dilkina, A. N. Elmachtoub, R. Finseth, et al. Maximizing the spread of cascades using network design. *UAI*, 2010.

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

APPENDIX

Proof of Prop. 1. Since edge $e \notin S$, we can always find two live-edge graphs within $\mathcal{X}_{G\setminus S}$ which differ by the edge e. The first live-edge graph X does not contain e, and the second live-edge graph X' contains the additional edge e. □

Proof of Prop. 2. Since $S \subset S \cup \{e\}$, the influence graph $G \setminus (S \cup \{e\})$ has one less edge than $G \setminus S$, while other parameters of the two graphs remain the same. This implies that any live-edge graph X generated from the former influence graph can always be generated from the latter one, which establishes the first part of the proposition. Furthermore, $\mathcal{X}_{G\setminus(S\cup\{e\})}$ contains those live-edge graphs without edge e, which is essentially the union of $\mathcal{X}^{\bar{e}}_{G\setminus S}$ and $\mathcal{X}^{\emptyset}_{G\setminus S}$ by definition. □

Proof of Prop. 3. We will explicitly construct a set $\Phi_i \subseteq \mathcal{X}^{\emptyset}_{G\setminus S}$ for each element $X_i \in \mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$. There are two types of elements in $\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})}$, and we will construct Φ_i respectively as follows: (1) If node v' has an incoming edge in X_i, then $\Phi_i = \{X_i\}$. Φ_i is contained in $\mathcal{X}^{\emptyset}_{G\setminus S}$ since $\mathcal{X}^{\emptyset}_{G\setminus(S\cup\{g\})} \subseteq \mathcal{X}^{\emptyset}_{G\setminus S}$ using a similar argument as in the space inclusion property. (2) Otherwise, $\Phi_i = \{X_i, X_i'\}$, where $X_i' = (V, E_{X_i} \cup \{g\})$ is obtained by extending X_i with edge g. Φ_i is also contained in $\mathcal{X}^{\emptyset}_{G\setminus S}$ since $g \notin S$ and hence X_i' is a valid live-edge graph in $\mathcal{X}^{\emptyset}_{G\setminus S}$. It is easy to see that the Φ_is are pairwise disjoint and form a partition of the space $\mathcal{X}^{\emptyset}_{G\setminus S}$. □

Proof of Prop. 4. We will consider two cases. When $\Phi_i = \{X_i\}$ is a singleton, $\Pr[X_i|G\setminus(S\cup\{g\})] = \Pr[X_i|G\setminus S]$ and hence the statement holds true trivially. When $\Phi_i = \{X_i, X_i'\}$, the difference $\Pr[X_i|G\setminus(S\cup\{g\})] - \sum_{H\in\Phi_i} \Pr[H|G\setminus S]$ is proportional to $p(v', X_i, G\setminus(S\cup\{g\})) - p(v', X_i, G\setminus S) - p(v', X_i', G\setminus S)$, where we only need to consider the contribution of the terminal node v' of edge $g = (u', v')$, since all other nodes contribute the same amount to the probability of each live-edge graph involved. Based on Eq. (3), the difference between the first two terms, $w(u', v')$, cancels out with the third term, $w(u', v')$, which shows that the difference $\Pr[X_i|G\setminus(S\cup\{g\})] - \sum_{H\in\Phi_i} \Pr[H|G\setminus S]$ is zero, and completes the proof. □