Efficient Algorithms to Optimize Diffusion Processes under the Independent Cascade Model

Xiaojian Wu[1] **Daniel Sheldon**[1,2] **Shlomo Zilberstein**[1]

[1] College of Information and Computer Sciences, University of Massachusetts Amherst
[2] Department of Computer Science, Mount Holyoke College

Abstract

We study scalable algorithms to optimize diffusion processes under the Independent Cascade model. We consider a broad class of intervention actions, including selecting sources, raising the probability that the diffusion propagates from one node to another and changing the topology of networks to facilitate the diffusion. Optimizing the selection of such actions with a limited budget tends to be NP-hard and is neither submodular nor supermodular. We provide scalable algorithms for three different problem settings that range in terms of the strength of the assumptions we make about the model. The algorithms are very efficient (faster than a baseline greedy algorithm), producing high-quality solutions in several diffusion maximization problems in the area of computational sustainability and in some cases also have provable approximation guarantees. These techniques offer promising results that may be applied to diffusion optimization problems in social and information networks.

1 Introduction

Dynamic phenomena such as the spread of information, ideas, and opinions [1, 2], and infectious disease propagation among humans [3] can be described as a diffusion process over an underlying network. Abstractly, the diffusion process in a network can be described as follows. A set of nodes called *sources* are set to be *infected* or *active* at the beginning of the process. Recursively, currently infected nodes can infect their neighbors with some probability. After a certain number of such *cascading* cycles, a large number of nodes become infected in the network. An interesting question is how to shape a given diffusion process so as to accomplish some desired objectives over infected nodes by taking intervention actions . An example is the source selection problem for viral marketing. Viral marketing can be modeled as a diffusion process in which people who purchase a product are considered to be "infected" and can send information or recommendations about the product to their friends, who may purchase the product as well and continue the cascading process. To maximize the number of people who eventually purchase the product, a company could pick K people, for examples ones having a large number of friends, and provide them with free products with the hope of triggering a large number of purchases. Here, the intervention action is modeled as adding sources. The question is, if we are allowed to add at most K sources, which ones should be selected so as to maximize the number of purchases resulting from the diffusion process. This optimization problem has been shown to have submodular objective function and therefore a natural greedy algorithm can obtain a solution that is provably within 63% of the optimal value [4].

Besides source selection, other intervention actions may be used to facilitate diffusion processes in different applications. For example, to maximize the number of purchases in viral marketing, a company can offer a person a small amount of money, if her friends purchase the product because of her recommendation [2]. In this case, the offer, which can be considered a type of intervention action, is in fact an incentive to increase the chance that the person sends a recommendation of the product to her friends, particularly recommendations that increase the likelihood of infection. In this case, the problem is to determine whom to make such offers to and how much money to offer. Another example is a social media website that recommends to users additional information outlets to increase the spread of ideas and memes. A third example is a decision maker who may cut off communication between communities to slow down the spread of rumors. These examples raise several interesting questions: how to model these intervention actions and whether we can develop efficient algorithms to select actions so as to optimize the outcome of the diffusion processes subject to initial constraints.

Some recent works model and study algorithms for problems that go beyond source selection and consider different diffusion models. Khalil *et al.* [5] consider two types of actions, adding edges to or deleting edges from the existing network, under a well-known diffusion model called the Linear Threshold (LT) model. They show that this network structure modification problem under the LT model has a supermodular objective and therefore can be solved by algorithms with provable approximation guarantees. Under the Susceptible Infected Recovered (SIR) model, some positive network optimization results exist: Tong *et al.* [6] 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 [7, 8]. Instead of maximizing (minimizing) the spread of substances directly, some methods typically optimize a static property of the network, in the hope of optimizing diffusion. For instance, Schneider *et al.* [8] proposed *betweenness centrality* as a heuristic for immunizing nodes or removing edges under the SIR model, while *degree centrality* was adopted by Gao *et al.* [7] to protect against virus propagation in email networks.

We focus on a well-known diffusion model called the Independent Cascade (IC) model [4]. Besides edge addition, edge deletion and source selection, we also consider other intervention actions, such as increasing the probability that a node infects its neighbors. To do this, we define a general decision making problem under the IC model to capture a broad class of intervention actions, including source selection and network structure modification. It can be shown that our general decision making problem is NP-hard and is neither submodular nor supermodular. In the area of social computing, we are particularly interested in finding algorithms that can produce high-quality solutions for networks with thousands or millions of nodes. To the best of our knowledge, there are no such efficient general-purpose algorithms for this class of decision problems. Sheldon *et al.* [9] propose an algorithm to handle the node addition problem under the IC model using the Sample Average Approximation (SAA) technique and mixed integer programming (MIP). But, their MIP is based on the assumption that the network is a directed acyclic graph. Furthermore, their algorithm can only scale up to networks with thousands of nodes. The greedy algorithm is faster, but can guarantee high-quality solutions only when the objective function is submodular. Heuristic search is another option, but it is hard to apply to large-scale networks [10] or when the number of candidate actions is large. Domain knowledge based methods can be used but they don't provide theoretical guarantees on the solution quality. Besides, we are interested in generic algorithms rather than ones that work well only for a specific application.

In this paper, we give a brief overview of algorithms that we recently created to handle the above decision problems. These algorithms are very efficient and can produce high-quality solutions, supported by both theoretical justification and experimental evidence. They are created under three different settings that become gradually more general. The most restricted setting is based on the assumption that the underlying network is a directed rooted tree with directed edges spreading out from a unique root. For this case, we created a fully polynomial-time approximation schema (FPTAS) with quadratic runtime complexity. A more general and harder setting is when the underlying network is a general directed graph, for which we created a very fast algorithm that is in fact faster than the greedy algorithm. Applying the algorithm to optimize a diffusion process of a species over fragmented landscape, we compute high-quality solutions compared to existing techniques. For the previous two settings, we only care whether the node is infected or not and ignore the time at which a node becomes infected. In the last setting that is the most general and difficult one, we consider the infection time and use the so-called *Continuous-Time Independent Cascade* (CTIC) model. Under this model, the definition of intervention actions is slightly different. We found an algorithm and successfully applied it to minimize the travel time that ambulances take to reach patients in different locations. Although all experiments are on problems in the area of computational sustainability, the algorithms have nice scalability and are potentially applicable to solving diffusion optimization problems in social and information networks.

The rest of the paper is organized as follows. Section 2 introduces the IC model and the decision making problem. Section 3 introduces algorithms for the case that the underlying networks are directed rooted trees. Section 4 introduces the algorithm for the case that the underlying networks are general directed graphs. Section 5 briefly introduces the CTIC model, the decision making problem under the CTIC model and solution methods.

2 Decision Making Under The Independent Cascade Model

Independent Cascade Model: The input is a directed graph $G = \{V, E\}$ where each node is either infected or uninfected. Once a node becomes infected, it has one chance to infect each of its uninfected neighbors independently with some probability. For example, once a node u becomes infected, it can infect a node v along an existing directed edge (u, v) with probability p_{uv}. We call p_{uv} the *infection probability* of u to v. If v fails to be infected at this point, u cannot infect v in the future. Once a node becomes infected, it remains infected forever. The process starts from a set of initially infected nodes called *sources* and terminates when no more nodes become infected.

Without loss of generality, we assume that there is a unique source. Otherwise, a dummy source is created and a directed edge is added from it to each other source with infection probability 1.0. Currently, we make the assumption that the infection time of any pair of nodes remains a common constant. Then, we can view the infection time to be 1 so that the diffusion process unfolds in discrete steps. In section 5, we will relax this assumption.

Intervention Actions: For each edge (u, v), we define a candidate action a_{uv} that can change the infection probability of (u, v) from p_{uv} to p'_{uv}. We call p_{uv} the *initial infection probability* and p'_{uv} the *modified infection probability*. Each action is associated with a specified cost.

Decision Making Problem: A policy denoted by π is defined as a subset of actions, or equivalently a subset of edges, for which the associated actions are taken. A policy is *feasible* if the total cost of taken actions is no greater than a given budget limit $\mathcal{B}$. For a given directed graph G with a unique (dummy) source and a specified budget limit $\mathcal{B}$, the decision making problem is to select the best policy among all feasible policies, with which the diffusion infects the maximum (minimum) expected number of nodes. Mathematically, the decision making problem is written as:

$$\max(\min)_{\pi} \sum_{v \in V} \mathbb{E}\left[\text{v is infected}\right] \qquad s.t.\ \ Cost(\pi) \leq \mathcal{B} \tag{1}$$

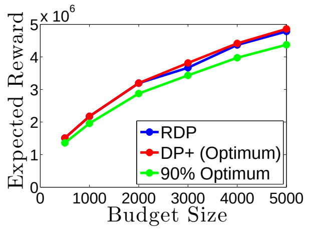
This problem is a complex stochastic optimization problem. Specifically, it can be shown that the problem is in general NP-hard and the objective function is neither submodular nor supermodular.

Extensions: The problem can be extended in two ways: 1) allowing one action to change the probabilities of multiple edges and 2) allowing multiple candidate actions with different strengths per edge. Algorithms that we will introduce later can be modified to handle one or both these extensions.

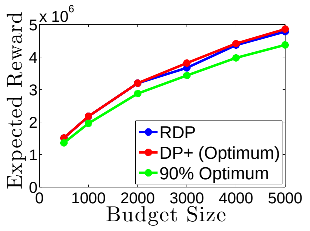
It is easy to see that the source selection problem is a special case of the above general decision problem. Given the input graph G, we add an extra node s as the dummy source and create a directed edge from it to each other node in G with the initial infection probability 0. For each such edge, we define an action that can raise the infection probability from 0 to 1, each with cost 1. Finally, we set $\mathcal{B} = K$. The objective is to maximize the expected number of nodes that are eventually infected. It can be shown that taking the action on the edge (s, v) is the same as setting v to be a source. Since the budget limit is K and each action takes one unit of cost, at most K sources can be selected. Therefore, the problem becomes a source selection problem.

3 Tree Structured Networks

In this section, we first consider the setting in which the underlying network is a directed rooted tree, a tree with a unique root (the source) and directed edges spreading out from the root. Let N_V be the number of nodes in the input graph. We make the following claim [11].



(a) Value Comparison



(b) Runtime

Figure 1: Solution quality and runtime for different budgets

Proposition 1. *When the underlying network is a directed rooted tree, the decision making problem is NP-hard and is neither submodular nor supermodular. For the maximization problem, we can find a FPTAS, which takes time $O(N_V^2/\epsilon^2)$ to find a policy with value at least $(1-\epsilon)$ times optimal value.*

The algorithm and proofs were presented at AAAI'14 [11]. The basic idea is to first build a dynamic programming algorithm that can calculate the optimal policy but is a pseudo-polynomial time algorithm and then use a rounding strategy to make the algorithm scalable. The resulted algorithm is called *rounded dynamic programming* (RDP). In the dynamic programming algorithm, each subtree is associated with a table, which is indexed by the values that are achievable by any feasible policy. The policy indexed by a value z gives the least cost among all policies that achieve z. Tables are computed recursively from small subtrees to large subtrees. The optimal policy is extracted from the table of the complete tree. However, this algorithm is not efficient because the number of achievable values or the size of a table can be as large as 2^{N_V}. To make the algorithm scalable, we use a rounding strategy. For each table, the range of all achievable values is discretized into a small number of non-overlapping intervals and all values in the same interval are considered to be the same. Therefore, the number of different values becomes the number of intervals which is much less than before. The granularity of the discretization is controlled by a paramter ϵ that affects both the runtime and the approximation quality.

We applied this algorithm to the barrier removal problem [12] where the goal is to maximize the spread of fish in a river network by selectively removing instream barriers (equivalently raising the infection probabilities). The results shown in Fig. 1 compare our algorithm (RDP) with the dynamic programming algorithm without the rounding strategy (DP), which produces the optimal solution, on a small network of 18550 nodes and 9354 candidate actions (some edges don't have actions). Our algorithm produces the near optimal solution and is much faster than "DP". Note that the larger budget size is, the more values are achievable and the larger the size of the table is. While "DP" takes 20 minutes for budget size 5000, "RDP" only takes 20 seconds. Moreover, we observe that the runtime of our algorithm in practice is much faster than the theoretical upper bound in Proposition 1.

In previous work [13], we have also presented a FPTAS to solve a more complex problem, for which each node sends out an influence that spreads separately in the network and the goal is to design the network to maximize the expected total number of influences received by all nodes.

4 General Directed Graph

In this section, we introduce a fast algorithm for general directed graphs. Although the algorithm doesn't have the nice approximation guarantee as RDP, it is very fast and is able to produce solutions with high qualities empirically. The basic idea is as follows. First, the *Sample Average Approximation* (SAA) technique is used to construct a network design problem that approximates the stochastic optimization problem (1). Then, a fast combinatorial algorithm is built based on the Lagrangian relaxation technique and the primal-dual schema to solve the constructed network design problem approximately. We describe below each part of the algorithm.

Sampling: We use a simple example shown in Fig. 2 to illustrate the idea of sampling. The original graph G has two nodes with u being the source. The initial infection probability is 0.2. An action a

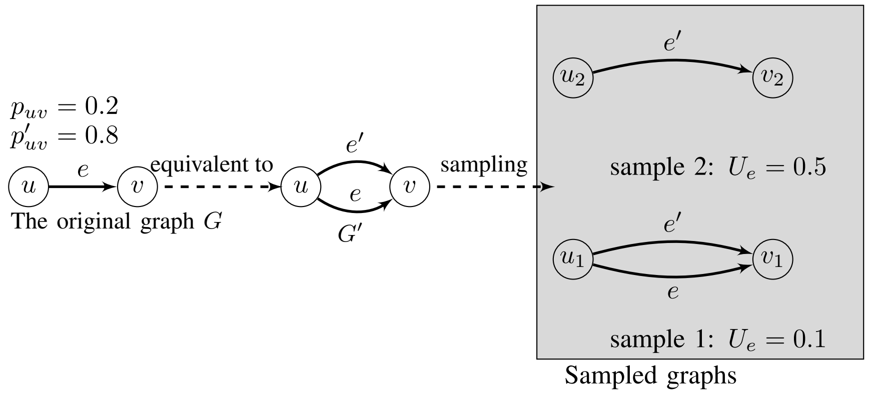


Figure 2: An example of sampling procedure.

can raise it into 0.8. We define another graph G' with two parallel edges e and e' where e' is present if and only if the action a is taken. The infection probabilities of e and e' are p_e^* and $p_{e'}^*$ respectively. We want to define p_e^* and $p_{e'}^*$ properly so that the probability that v is infected is the same for both graphs. The way we define them is as follows.

Define U_e to be a random variable uniformly distributed in range $[0, 1]$. u infects v via e' if $U_e \leq p'_{uv}$ and via e if $U_e \leq p_{uv}$. Therefore, for both graphs, if the action a is taken, the probability that v is infected is 0.8. If the action a is not taken, the probability is 0.2. For the general directed graphs, we make the following claim.

Lemma 1. *Given the original graph G and a constructed graph G' that has two parallel edges for each edge in G, we can define the infection probabilities for the parallel edges properly such that the probability that any node v is infected is the same for both graphs.*

For a given constructed graph G', we sample a sequence of graphs independently of any policy. In each sampled graph, we determine for each edge, by drawing a sample of U_e, whether the infection via that edge succeeds. If yes, the edge is present in that sampled graph. Otherwise, the edge is absent. For example, as shown in the last part of Fig. 2, the sample 1 has $U_e = 0.1$. Then, the infections via both edges succeed and both edges are present in the first sample.

Given N graphs sampled using the above procedure, we construct a network design problem, in which an action is associated with a set of edges and the goal is to purchase edge sets to maximize the average number of nodes that are connected to sources in sampled graphs. The ratio that a node is connected to the source is the approximation of the probability that the node is infected. The network design problem is

$$\max_{\pi} \frac{1}{N} \sum_{i=1:N} \text{\# of nodes connected to sources in } i\text{th sample under } \pi \quad s.t. \; Cost(\pi) \leq \mathcal{B} \tag{2}$$

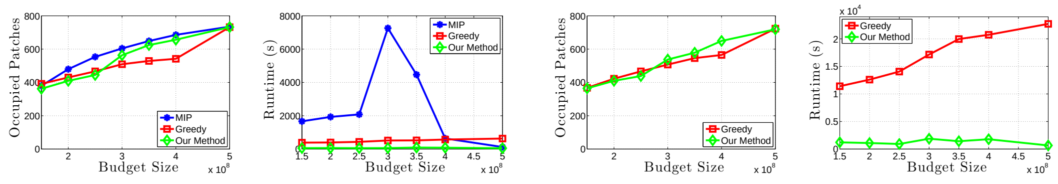
Solving the Network Design Problem: The problem (2) can be written as a Mixed Integer Program (MIP). It is NP-hard and its size increases linearly with the number of samples and the number of edges, so solving it by a standard MIP solver is inefficient for large-scale networks. In previous work [14], we provide a fast algorithm to solve it approximately. The idea is to use Lagrangian relaxation method to bring the budget constraint into the objective together with a Lagrangian multiplier β. The *relaxation problem* is parameterized by the β and can be solved efficiently by a primal-dual algorithm. A bisection procedure is used to find a β, for which the near optimal solution of the relaxation problem is also a near optimal solution of the problem (2).

Let N_V denote the number of nodes and N_E denote the number of edges. Let M be the number of iterations in bisection procedure and K be the number actions that are selected by the greedy algorithm. As mentioned in section 2, there are N_E candidate actions, one for each edge. Now, we make the following claim.

Proposition 2. *For the general directed graph, we can find an approximate algorithm whose performace depends on N, the number of samples. With N samples, the algorithm takes time $O\left(M(NN_VN_E + N_EN_E)\right)$ while a greedy algorithm takes time $O(KNN_E(N_E + N_V))$.*

For the greedy algorithm, there are K iterations. At each iteration, $O(N_E)$ candidate actions are tested and the one with the maximum marginal gain is selected. For each candidate action, we calculate the number of nodes that are reachable from the source, which takes time $N(N_E + N_V)$.

In experiments, we observe that M is usually very small (e.g. between 10 and 20) and a small number of samples (e.g. $N = 50$) are usually enough for convergence. If we assume that $NN_V > N_E > N_V$, the runtimes of our algorithm and the greedy algorithm are $O(MNN_VN_E)$ and $O(KNN_EN_E)$ respectively. Then, our algorithm is $\frac{KN_E}{MN_V}$ times faster.



(a) Value (b) Runtime (c) Value (d) Runtime

Figure 3: Value and runtime comparisons. "MIP" is the mixed integer solver (Gurobi) that produces the optimal solution for the discrete problem. The first two figures are for 10 samples. The last two figures are for 300 samples.

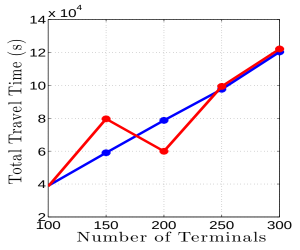
We applied our algorithm to the conservation planning problem [9], for which the goal is to maximize the spread of birds by selectively adding lands (nodes) into the networks. This problem is a special case of our general decision making problem. Similar to our method, Sheldon. et. al construct a network design problem using the SAA technique and solve the constructed problem using a MIP solver. Our sampling procedure degenerates into their sampling method in this node addition problem. But, we provide a much faster way to solve the constructed problem. The network contains 253 thousands nodes, about 600 millions edges and only 443 candidate actions where one action can change the probabilities of multiple edges simultaneously. Nodes in the graph are grouped into 100 layers and edges only connected nodes in adjacent layers. This special structure of the network enables us to implement our algorithm much faster than the theoretical bounds. The results of solving the constructed network design problems are shown in Fig. 3. More results are provided in [14, 9]. With 10 samples, our algorithm and the greedy algorithm are much faster than "MIP" and produce near optimal solutions. For 300 samples where "MIP" fails to finish within a reasonable amount of time, our algorithm is much faster than the greedy algorithm and produces slightly better solutions.

5 Continuous-Time Independent Cascade Model

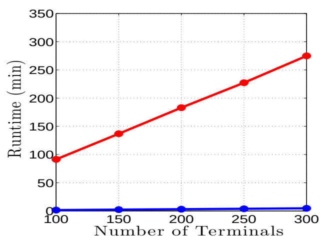
Now, we consider the last setting when the infection time is not a constant but randomly distributed. We use the *Continuous-Time Independent Cascade (CTIC)* model. In this case, the infection time is randomly distributed in range $[0, \infty]$ where ∞ means the infection never succeeds. Then, the process unfolds in the following way. Initially, a set of nodes (sources) are infected. Once a node becomes infected recently, it samples an infection time t randomly and independently for an uninfected neighbors v. After time t, v becomes infected. The process terminates when no more nodes can be infected within a finite amount of time.

Under this model, we define the decision making problem as follows. For an edge (u, v), we define two different probability distributions of the infection time, one for the action a_{uv} being taken and the other one for the action a_{uv} not taken, both over $[0, \infty]$. For example, if a_{uv} is not taken, the infection time follows the exponential distribution Exp(λ_1). If a_{uv} is taken, the infection time follows the distribution Exp(λ_2) with $\lambda_2 > \lambda_1$. In this case, we say that the action makes the infection time *stochastic shorter* in the sense that after taking the action, the probability that the infection time is less than t (any value in $[0, t]$) becomes larger. The objective is to minimize the expected average time for a node to become infected. However, for some node the infection time may be ∞, so we set a big penalty M_v, as the trade-off parameter, for the node v that has the infinite infection time. Then, the decision making problem is written as

$$\max_{\pi} \frac{1}{N_V} \mathbb{E}[\sum_{v \text{ infected}} \text{infection time of } v + \sum_{v \text{ uninfected}} M_v] \quad s.t.\ Cost(\pi) \leq \mathcal{B} \tag{3}$$



(a) Value



(b) Time

Figure 4: Performance on the pre-disaster preparation problem.

where the objective is the expected average infection time of all nodes. Other objectives, for example, maximizing the expected number of nodes being infected before certain time point, are interesting as well. It is our future work to consider those objectives [15, 16].

To solve the problem (3), we use the same basic idea as in the last section, but the sampling procedure and the algorithm to solve the constructed network design problem are different. We directly make the following claim. The details can be found in [17].

Proposition 3. *For CTIC model and the general directed graph, we can find an approximate algorithm whose performance depends on N, the number of samples. With N samples, the algorithm takes time $O(M(N^2N_E^2 + N^2N_V^2))$ while a greedy algorithm takes time $O(KNN_E^2 logNN_V)$.*

The meaning of N_V, N_E, M and K are the same as in the last section. For the greedy algorithm, there are K iterations and in each iteration, $O(N_E)$ candidate actions are tested, for each of which we run Dijkstra's algorithm once to calculate the single-source shortest paths by time $O(NN_E logNN_V)$. If $N_E > N_V$, our algorithm takes time $O(MN^2N_E^2)$. So, in terms of the worst case analysis, our algorithm is faster only if $MN < KlogNN_V$. But, in practice, we observe that our algorithm runs faster than the worst case time complexity.

We applied our algorithm to solve an instance of the pre-disaster preparation problem introduced by [18], for which the goal is to minimize the total travel time that ambulances need to reach patients in different locations during a flood. The travel of ambulances in the road network can be modeled by the CTIC model. A node represents a specific location in the road network and an edge represents a piece of road segment. The ambulance center is encoded as the source. The travel time on a road segment is modeled as the infection time along the correspondent edge. A node is infected by time t is equivalent that the node can be reached by time t. An action can reinforce one road segment to make its travel time stochastically shorter during the flood.

In this problem, only a portion of nodes are needed to be reached, so we only minimize the total travel time to those nodes. With this in mind, we can reduce the runtime of our algorithm dramatically. We first tested on a small network (10037 edges) and the results are shown in Fig. 4. The greedy algorithm performed similarly well as our algorithm in terms of quality, but was much slower. We also tested our algorithm on a large network (55687 edges) where one iteration of the greedy algorithm takes more than 10 hours. Our algorithm took about 6 hours and produced much better solution than other fast algorithms, such as the algorithm that randomly enumerates policies for 10 hours and picks the best one. Additional results can be found in our AAAI'16 paper [17].

6 Conclusion

In this paper, we formulate the problem of optimizing diffusion processes using a general decision making problem under the Independent Cascade model. Then, we introduce efficient algorithms for three different settings 1) the underlying network is a directed rooted tree, 2) the network is a general directed graph and 3) the infection time is continuous. The details of these algorithms are in our published papers in which we apply them to solve several problems in the area of computational sustainability. The results show that our algorithms are much faster than existing algorithms and produce near optimal solutions. The good scalability of these algorithms make them highly relevant to other diffusion optimization problems in social and information networks.

References

[1] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[2] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[3] Roy M Anderson, Robert M May, and B Anderson. *Infectious diseases of humans: dynamics and control*, volume 28. Wiley Online Library, 1992.

[4] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[5] Elias Boutros Khalil, Bistra Dilkina, and Le Song. Scalable diffusion-aware optimization of network topology. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1226–1235. ACM, 2014.

[6] Hanghang Tong, B Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 245–254. ACM, 2012.

[7] Chao Gao, Jiming Liu, and Ning Zhong. Network immunization and virus propagation in email networks: experimental evaluation and analysis. *Knowledge and information systems*, 27(2):253–279, 2011.

[8] Christian M Schneider, Tamara Mihaljev, Shlomo Havlin, and Hans J Herrmann. Suppressing epidemics with a limited amount of immunization units. *Physical Review E*, 84(6):061911, 2011.

[9] Daniel Sheldon, Bistra Dilkina, Adam Elmachtoub, Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla Gomes, David Shmoys, William Allen, Ole Amundsen, and William Vaughan. Maximizing the spread of cascades using network design. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 517–526, 2010.

[10] Hermann Schichl and Meinolf Sellmann. Predisaster preparation of transportation networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 709–715, 2015.

[11] Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Rounded dynamic programming for tree-structured stochastic network design. *Proc. of the 28th Conference on Artificial Intelligence (AAAI)*, 2014.

[12] Jesse Rush OHanley and David Tomberlin. Optimizing the removal of small fish passage barriers. *Environmental Modeling & Assessment*, 10(2):85–98, 2005.

[13] Xiaojian Wu, Daniel R Sheldon, and Shlomo Zilberstein. Stochastic network design in bidirected trees. In *Advances in Neural Information Processing Systems*, pages 882–890, 2014.

[14] Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Fast combinatorial algorithm for optimizing the spread of cascades. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2655–2661, 2015.

[15] Nan Du, Le Song, Manuel Gomez-Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3147–3155, 2013.

[16] Manuel Gomez Rodriguez and Bernhard Schölkopf. Influence maximization in continuous time diffusion networks. *arXiv preprint arXiv:1205.1682*, 2012.

[17] Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Optimizing resilience in large scale networks. *Accepted in the Thirtieth Conference on Artificial Intelligence*, 2016.

[18] Srinivas Peeta, F. Sibel Salman, Dilek Gunnec, and Kannan Viswanath. Pre-disaster investment decisions for strengthening a highway network. *Computers and Operations Research*, 37(10):1708–1719, 2010.