Comput Optim Appl (2018) 69:563–595
https://doi.org/10.1007/s10589-017-9958-x

A two-stage stochastic programming approach for influence maximization in social networks

Hao-Hsiang Wu[1] · **Simge Küçükyavuz**[1]

Received: 17 August 2016 / Published online: 23 October 2017

Abstract We consider stochastic influence maximization problems arising in social networks. In contrast to existing studies that involve greedy approximation algorithms with a 63% performance guarantee, our work focuses on solving the problem optimally. To this end, we introduce a new class of problems that we refer to as two-stage stochastic submodular optimization models. We propose a delayed constraint generation algorithm to find the optimal solution to this class of problems with a finite number of samples. The influence maximization problems of interest are special cases of this general problem class. We show that the submodularity of the influence function can be exploited to develop strong optimality cuts that are more effective than the standard optimality cuts available in the literature. Finally, we report our computational experiments with large-scale real-world datasets for two fundamental influence maximization problems, independent cascade and linear threshold, and show that our proposed algorithm outperforms the basic greedy algorithm of Kempe et al. (Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, KDD'03, New York, NY, USA, ACM, pp 137–146, 2003).

Keywords Social networks · Independent cascade · Linear threshold · Influence maximization · Stochastic programming · Submodularity

✉ Simge Küçükyavuz
simge@uw.edu

Hao-Hsiang Wu
hhwu2@uw.edu

[1] Industrial and Systems Engineering, University of Washington, Seattle, WA, USA

1 Introduction

The exploding popularity of social networking services, such as Facebook, LinkedIn, Google+ and Twitter, has led to an increasing interest in the effective use of word-of-mouth to market products or brands to consumers. A few individuals, seen as influencers, are targeted with free merchandise, exclusive deals or new information on a product or brand. Marketers hope that these key influencers promote the product to others in their social network through status updates, blog posts or online reviews and that this information propagates throughout the social network from peers to peers of peers until the product "goes viral." Therefore, a key question for marketers with limited budgets and resources is to identify a small number of individuals whom to target with promotions and relevant information so as to instigate a cascade of peer influence, taking into account the network effects.

1.1 Literature review

Domingos and Richardson [15] first introduce the problem of finding which customers to target to maximize the spread of their influence in the social network. The authors propose a Markov random-field-model of the social network, where the probability that a customer is influenced takes into account whether her connections are influenced. After building this network, the authors propose several heuristics to identify which k individuals to target in a viral marketing campaign, where k is a user-defined positive integer. Kempe et al. [21] formalize the optimization problem and introduce two fundamental models to maximize the influence spread in a social network: the *independent cascade* model and the *linear threshold* model. The authors show that the optimization problems are NP-hard, assuming that there is an efficient oracle to compute the influence spread function. This seminal work spurred a flurry of research on social networks with over 4600 citations recorded by Google Scholar in August 2017. Wang et al. [53] show that calculating the influence spread function is #P-hard under the probabilistic assumptions of Kempe et al. [21]. Therefore, the independent cascade problem is #P-hard, and there are two sources of difficulty. First, the calculation of the influence spread function is hard because there is an exponential number of scenarios. This difficulty is overcome by using sampling. Second, the seed selection is combinatorial in nature, and requires the evaluation of an exponential number of choices. This difficulty is overcome by seeking heuristic solutions in the literature. We describe the results of the seminal paper by Kempe et al. [21] and the subsequent developments in Sect. 2.

The majority of the existing work on optimization-based methods for social network analysis focus on various aspects other than influence maximization (see the review by Xanthopoulos et al. [55]) with the exception of some recent work [18,42], which develop mathematical programming approaches to solve *deterministic* weighted target set selection problems so that the total cost of influencing *all* nodes in a social network is minimized. Outside the influence maximization realm, the first class of problems studied is that of identifying the influential nodes of a network with respect to the nodes' centrality and connectivity. As an example of this class of problems, Arulselvan et al.

[3] propose an integer programming formulation for the problem of identifying k nodes whose removal from a *deterministic* social network causes maximum fragmentation (disconnected components). The second class is that of clustering the nodes of a *deterministic* social network to identify the cohesive subgroups of the network. For example, Balasundaram et al. [4] and Ertem et al. [16] utilize optimization models to identify clique relaxations. Third, game-theoretic approaches are used to study various aspects of social networks, such as modeling competitive marketing strategies of two firms to maximize their market shares (see, e.g., [7]).

In contrast to these models, we focus on the stochastic influence maximization problems and propose a two-stage stochastic programming method. In addition, by utilizing the submodularity of the second stage value (objective) function, we develop effective decomposition algorithms. Two-stage stochastic programming is a versatile modeling tool for decision-making under uncertainty. In the first stage, a set of decision needs to be made when some parameters are random. In the second stage, after the uncertain parameters are revealed, a second set of (recourse) decisions are made so that the expected total cost is minimized. We refer the reader to Birge and Louveaux [8] and Shapiro et al. [47] for an overview of stochastic (linear) programming. To the best of our knowledge, Song and Dinh [49] provide the only study besides ours that uses a stochastic programming approach to solve a problem in social networks. In this paper, the authors consider the problem of protecting some arcs of a social network (subject to a limited budget) so that the damage caused by the spread of rumors from their sources to a set of targeted nodes is minimized.

1.2 Our contributions

Despite the ubiquity of social networks, there has been a paucity of research in finding provably optimal solutions to the two fundamental problems of maximizing influence in social networks (independent cascade and general threshold). The algorithms studied to date are approximation algorithms with a worst-case guarantee within 63% optimal ([22], and references therein). The proposed heuristics are tested on real social networks and compared to other simple heuristics. However, their practical performance has not been tested against the optimal solution due to the hardness of the problem and the unavailability of an algorithm that can find the optimal solution for large-scale instances of the problem. To fill this gap, we introduce a new class of problems that we refer to as two-stage stochastic submodular optimization models. We propose a delayed constraint generation algorithm to find the optimal solution to this class of problems with a finite number of samples. The proposed delayed constraint generation algorithm exploits the submodularity of the second-stage value function. The influence maximization problems of interest are special cases of this general problem class. Utilizing the special structure of the influence function, we give an explicit characterization of the cut coefficients of the submodular inequalities, and identify conditions under which they are facet-defining for the full master problem that is solved by delayed constraint generation. This leads to a more efficient implementation of the proposed algorithm than is available from a textbook implementation of available algorithms for this class of problems [5,35,51]. In addition, we give the

complete linear description of the master problem for $k = 1$, where k is the number of nodes targeted in a social network. We illustrate our proposed algorithm on the classical *independent cascade* and *linear threshold* problems [21]. In our computational study, we show that our algorithm outperforms a basic implementation of the greedy heuristic in most of the large-scale real-world instances.

We note that while we demonstrate our algorithms on the independent cascade and linear threshold models, our approach is more generally applicable to many other variants of the influence maximization problem studied previously in the literature. Furthermore, beyond social networks, there are other applications of identifying a few key nodes in complex networks for which our models are applicable. For example, Ostfeld and Salomons [40] consider the problem of locating costly sensors on the crucial junctures of the water distribution network to ensure water quality and safety by the early detection and prevention of outbreaks. The models could also be useful in the development of immunization strategies in epidemic models (see, e.g. [31]), and prevention of cascading failures in power systems (see, e.g., [19]). Furthermore, it also applies to more general stochastic optimization problems that have submodular second-stage value functions. For example, recently Contreras and Fernández [13] consider a *deterministic* hub location problem, and prove that the routing costs in the objective function are submodular. Using this observation, the authors employ the delayed constraint generation algorithm of Nemhauser and Wolsey [35] to solve the optimization problem more effectively than the existing models for this problem. Our proposed algorithm can be used to solve a *stochastic* extension of the hub location problem, where in the first stage, the hub locations are determined, and in the second stage, after the revelation of uncertain demand of multiple commodities, the optimal routing decisions are made. Hence, the general two-stage stochastic submodular optimization model and method that we introduce in Sect. 3 has a potential broader impact beyond social networks.

1.3 Outline

In Sect. 2, we formally introduce the influence maximization problem and review the greedy algorithm of Kempe et al. [21]. In Sect. 3, we define a general two-stage stochastic submodular optimization model, and describe a delayed constraint generation algorithm that exploits the submodularity of the second-stage value function. We show that for $k = 1$, solving a linear program with a simple set of submodular optimality cuts and the cardinality restriction on the seed set guarantees an integer optimal solution. In Sect. 4, we consider the two fundamental influence maximization problems as defined by Kempe et al. [21], namely *independent cascade* and *linear threshold*. We show that for these special cases of the two-stage stochastic submodular optimization problems, we can obtain an explicit form of the submodular optimality cuts and identify conditions under which they are facet defining. In Sect. 5, we report our computational experience with large-scale real-world datasets, which show the efficacy of the proposed approach in finding optimal solutions as compared to the greedy algorithm. We share our conclusions and future work in Sect. 6.

2 Greedy algorithm of Kempe et al. [21]

In this section, we describe the modeling assumptions of Kempe et al. [21], and overview the greedy hill-climbing algorithm proposed by these authors. Suppose that we are given a social network $G = (V, A)$, where $|V| = n, |A| = m$. The vertices represent the individuals, and an arc $(i, j) \in A$ represents a potential influence relationship between individuals i and j. Our goal is to select a subset of *seed* nodes, $X \subset V$, with $|X| \leq k < n$ to activate initially, so that the expected number of people influenced by X (denoted by $\sigma(X)$) is maximized, where k is a given integer. (Note that the original problem statement is to select exactly k nodes to activate. However, for the relaxation that seeks $|X| \leq k$ seed nodes that maximize influence, there exists a solution for which the inequality holds at equality.) The influence propagation is assumed to be *progressive*, in other words, once a node is activated it remains active.

Kempe et al. [21] show that for various influence maximization problems, the influence function $\sigma(X)$ is nonnegative, monotone and submodular. Therefore, the influence maximization problem involves the maximization of a submodular function. The authors show that this problem is NP-hard even if there is an efficient oracle to compute the influence spread function. However, using the results of Cornuéjols et al. [14] and Nemhauser et al. [36] that the greedy method gives a $(1 - \frac{1}{e})$-approximation algorithm for maximizing a nonnegative monotone submodular function, where e is the base of the natural logarithm, Kempe et al. [21] establish that the greedy hill-climbing algorithm solves the influence maximization problem with a constant (0.63) guarantee, assuming that the function $\sigma(X)$ can be calculated efficiently. Recognizing the computational difficulty of calculating $\sigma(X)$ exactly, which involves taking the expectation of the influence function with respect to a finite (but exponential) number of scenarios, Kempe et al. [21] propose Monte-Carlo sampling, which provides a subset of equiprobable scenarios, Λ, of moderate size. Letting σ_ω denote the influence function for scenario $\omega \in \Lambda$, we get $\sigma(X) = \frac{1}{|\Lambda|} \sum_{\omega \in \Lambda} \sigma_\omega(X)$. The basic greedy approximation algorithm of Kempe et al. [21] is given in Algorithm 1.

Subsequently, Wang et al. [53] formally show that calculating $\sigma(X)$ is #P-hard under the assumption of independent arc probabilities π_{ij}, $(i, j) \in A$. Therefore, Kempe et al. [22] propose a modification where an arbitrarily good approximation of $\sigma(X)$ is obtained in polynomial time by sampling from the true distribution. In particular, Kempe et al. [22] show that for a sample size of $\Omega\left(\frac{n^2}{\varepsilon^2} \ln(1/\alpha)\right)$, the average number of activated nodes over the sample is a $(1 \pm \varepsilon)$-approximation to $\sigma(X)$, with probability at least $1 - \alpha$.

Algorithm 1: Greedy Approximation Algorithm of Kempe et al. [21].

```
Start with X = ∅ and a sample set of scenarios Λ;
while |X| ≤ k do
    For each node i ∈ V\X, use the sample Λ to approximate σ(X ∪ {i});
    Add node i with the largest estimate for σ(X ∪ {i}) to X;
end
Output the set of seed nodes, X.
```

Further algorithmic improvements to the greedy heuristic are given in the literature (see [10,22], for an overview). Most notably, Borgs et al. [9] give a randomized algorithm for finding a $(1 - 1/e - \epsilon)$-approximate seed sets in $O((m + n)\epsilon^{-3} \log n)$ time for any precision parameter $\epsilon > 0$. Note that this run time is independent of the number of seeds k. The authors show that the running time is close to the lower bound of $\Omega(m+n)$ on the time required to obtain a constant factor randomized approximation algorithm. The proposed randomized algorithm has a success probability of 0.6, and failure is detectable. Therefore, the authors suggest repeated runs if failure is detected to improve the probability of success.

3 A general two-stage stochastic submodular optimization model and method

In this section, we define a general two-stage stochastic submodular optimization model and outline a delayed constraint generation algorithm for its solution. Then, in Sect. 4, we describe how this general model and method is applicable to the influence maximization problems of interest.

Let $(\Lambda, \mathcal{F}, \mathbb{P})$ be a finite probability space, where the probability of an elementary event $\omega \in \Lambda$ is $p_\omega := \mathbb{P}(\omega)$. Consider a general two-stage stochastic binary program

$$\max \quad c^\top x + \sum_{\omega \in \Lambda} p_\omega \sigma_\omega(x) \tag{1a}$$

$$\text{s.t.} \quad x \in \mathcal{X} \tag{1b}$$

$$x \in \{0, 1\}^n, \tag{1c}$$

where $c \in \mathbb{R}^n$ is a given objective vector, the set $\mathcal{X}$ represents the constraints on the first-stage variables x and $\sigma_\omega(x)$ is the objective function of the second-stage problem for scenario $\omega \in \Lambda$ solved as a function of first-stage decisions given by

$$\sigma_\omega(x) := \max \quad q^\top y \tag{2a}$$

$$\text{s.t.} \quad y \in \mathcal{Y}(x, \omega). \tag{2b}$$

Here q is an objective vector of conformable dimension, y is the vector of second-stage decisions, and $\mathcal{Y}(x, \omega)$ defines the set of feasible second-stage decisions for a given first-stage vector x, and the realization of the uncertain outcomes given by the scenario $\omega \in \Lambda$. We assume that $\sigma_\omega(x) : \{0, 1\}^n \to \mathbb{R}$ is known to be a submodular function for each $\omega \in \Lambda$, and refer to the optimization problem (1) as a *two-stage stochastic submodular optimization model*. It is well-known from the property of submodular functions that if $\sigma_\omega(x)$, $\omega \in \Omega$ is submodular, then so is the second-stage value function $\sigma(x) = \sum_{\omega \in \Lambda} p_\omega \sigma_\omega(x)$, which is a nonnegative (convex) combination of submodular functions. Furthermore, we assume that $\mathcal{Y}(x, \omega)$ is a non-empty set for each $x \in \mathcal{X}$, $\omega \in \Lambda$, a property known as *relatively complete recourse* in stochastic programming.

Next we overview a delayed constraint generation approach to solve the two-stage program (1). The generic master problem at an iteration is formulated as

$$\max \quad c^\top x + \sum_{\omega \in \Lambda} p_\omega \theta_\omega \tag{3a}$$

$$\text{s.t.} \quad x \in \mathcal{X} \tag{3b}$$

$$(x, \theta) \in \mathcal{C}, \tag{3c}$$

where θ is a $|\Lambda|$-dimensional vector of variables θ_ω representing the second-stage objective function approximation for scenario ω, constraints (3c) represent the so-called *optimality cuts* generated until this iteration. The set of inequalities in $\mathcal{C}$ provides a piecewise linear approximation of the second stage value function, which is iteratively refined through the addition of the optimality cuts. (We will describe different forms of these inequalities in the following discussion.) Let $(\bar{x}, \bar{\theta})$ be the optimal solution to the master problem at the current iteration. Then for all $\omega \in \Lambda$ we solve the subproblems (2) to obtain $\sigma_\omega(\bar{x})$. We add valid optimality cuts to $\mathcal{C}$ if $\bar{\theta}_\omega > \sigma_\omega(\bar{x})$ for any $\omega \in \Lambda$, otherwise we deduce that the current solution $\bar{x}$ is optimal. The generic version of the delayed constraint generation algorithm is given in Algorithm 2. In this algorithm, ε is a user-defined optimality tolerance. The particular implementation of Algorithm 2 depends on the method with which subproblems are solved to obtain $\sigma_\omega(\bar{x})$ (in line 5 of Algorithm 2), and the form of the optimality cuts added to the master problem (in line 7 of Algorithm 2). In this section, we explore the possibility of utilizing the submodularity of the second-stage value function in a two-stage stochastic programming problem. We discuss a natural alternative in "Appendix A", which we use as a benchmark.

Algorithm 2: Delayed Constraint Generation Algorithm.

```
1  Start with C = {0 ≤ θ_ω ≤ n, ω ∈ Λ}. Let LB = −∞ and UB = ∞;
2  while UB − LB > ε do
3      Solve the master problem (3) and obtain (x̄, θ̄). Let UB be the upper bound obtained from the
       optimal objective value of the master problem;
4      for ω ∈ Λ do
5          Solve Subproblem (2) to obtain σ_ω(x̄) ;
6          if θ̄_ω > σ_ω(x̄) then
7              Add an optimality cut to C;
8          end
9      end
10     Let σ(x̄) = Σ_{ω∈Λ} p_ω σ_ω(x̄). if LB < σ(x̄) then
11         Let LB ← σ(x̄), and let x̂ ← x̄ be the incumbent solution
12     end
13 end
14 Output the set of seed nodes X = {i ∈ V : x̂_i = 1}.
```

Nemhauser and Wolsey [35] give submodular inequalities to describe the maximum of a submodular set function (see also [37]). Consider the polyhedra $\mathcal{S}_\omega = \{(\theta_\omega, x) \in \mathbb{R} \times \{0,1\}^n : \theta_\omega \leq \sigma_\omega(S) + \sum_{j \in V \setminus S} \rho_j^\omega(S) x_j, \forall S \subseteq V\}$, and $\mathcal{S}'_\omega = \{(\theta_\omega, x) \in \mathbb{R} \times \{0,1\}^n : \theta_\omega \leq \sigma_\omega(S) - \sum_{j \in S} \rho_j^\omega(V \setminus \{j\})(1 - x_j) + \sum_{j \in V \setminus S} \rho_j^\omega(S) x_j, \forall S \subseteq V\}$ for $\omega \in \Lambda$, where $\rho_j^\omega(S) = \sigma_\omega(S \cup \{j\}) - \sigma_\omega(S)$ is the marginal contribution of adding $j \in V \setminus S$ to the set S.

Theorem 3.1 (cf. [35]) *For a submodular and nondecreasing set function* $\sigma_\omega : 2^n \to \mathbb{R}$, $\bar{X}$, *with a characteristic vector* $\bar{x}$, *is an optimal solution to* $\max_{S\subseteq V:|S|\leq k}\{\sigma_\omega(S)\}$, *if and only if* $(\theta_\omega, \bar{x})$ *is an optimal solution to* $\{\max\ \theta_\omega : \sum_{j\in V} x_j \leq k, (\theta_\omega, x) \in \mathcal{S}_\omega\}$. *Similarly for a submodular and nonmonotone set function* $\sigma_\omega : 2^n \to \mathbb{R}$, $\bar{X}$, *with a characteristic vector* $\bar{x}$, *is an optimal solution to* $\max_{S\subseteq V:|S|\leq k}\{\sigma_\omega(S)\}$, *if and only if* $(\theta_\omega, \bar{x})$ *is an optimal solution to* $\{\max\ \theta_\omega : \sum_{j\in V} x_j \leq k, (\theta_\omega, x) \in \mathcal{S}'_\omega\}$.

Therefore, we can adapt the algorithm of Nemhauser and Wolsey [37] given for deterministic submodular maximization problems to two-stage stochastic submodular optimization problems. Note that because there are exponentially many submodular inequalities, we cannot add all of them a priori to the formulation. Instead, we use a delayed constraint generation approach that adds the violated inequalities as needed and solves the resulting mixed-integer program by branch-and-cut (see, e.g., [6] for an overview of the general form of a delayed constraint generation algorithm for linear programs). The proposed method takes the form of Algorithm 2. For a given first stage solution, $\bar{x}$, which is a characteristic vector of the set $\bar{X}$, and scenario $\omega \in \Lambda$, we use the optimality cut

$$\theta_\omega \leq \sigma_\omega(\bar{X}) + \sum_{j\in V\setminus\bar{X}} \rho_j^\omega(\bar{X}) x_j, \tag{4}$$

if the second-stage value function $\sigma_\omega(x)$ is nondecreasing and submodular. If the second-stage value function $\sigma_\omega(x)$ is nonmonotone and submodular, then we use the optimality cut given by the inequality

$$\theta_\omega \leq \sigma_\omega(\bar{X}) - \sum_{j\in\bar{X}} \rho_j^\omega(V\setminus\{j\})(1 - x_j) + \sum_{j\in V\setminus\bar{X}} \rho_j^\omega(\bar{X}) x_j. \tag{5}$$

We refer the reader to Nemhauser and Wolsey [35] for validity of inequalities (4)-(5). Ahmed and Atamtürk [1] and Yu and Ahmed [56] strengthen the submodular inequalities by lifting, under the condition that the submodular utility function is strictly concave, increasing, and differentiable. These assumptions do not apply to our problem.

Corollary 3.1 *Algorithm 2 with optimality cuts* (4) *and* (5) *converges to an optimal solution in finitely many iterations for a two-stage stochastic program with binary first-stage decisions,* $x \in \{0, 1\}^{|V|}$ *for which the second-stage value function,* $\sigma_\omega(x)$, $\omega \in \Lambda$, *(*$|\Lambda|$ *finite) is submodular nondecreasing and submodular nonmonotone, respectively.*

Proof The result follows from the fact that the number of feasible first stage solutions is finite, and from Theorem 3.1. □

Note that Algorithm 2 is generally applicable to two-stage stochastic programs with binary first-stage decisions, $x \in \{0, 1\}^n$, where the second-stage value function, $\sigma_\omega(x)$ is submodular for all $\omega \in \Lambda$. There is very limited reporting on the computational performance of this algorithm even for deterministic submodular maximization problems for which the method was originally developed (see [13,26], for computational results

on quadratic cost partition and hub location problems, respectively). Kawahara et al. [20] utilize the algorithm of Nemhauser and Wolsey [35] along with convexity cuts derived from Lovász extension of submodular functions to solve deterministic submodular maximization problems. The authors derive inequalities that are not globally valid as they cut off solutions that do not strictly improve upon the current incumbent solution. To the best of our knowledge, our work is the first adaptation and testing of this algorithm for two-stage stochastic optimization. While the submodular inequalities (4)-(5) are implicit in that they require the calculation of $\rho_j^\omega(\cdot)$ terms, in Sect. 4, we give an explicit form of the submodular optimality cuts for influence maximization problems of interest. This allows us to characterize conditions under which the optimality cuts are strong, and to improve the performance of a textbook implementation of the algorithm of Nemhauser and Wolsey [35].

Next, we consider the special case of cardinality-constrained first-stage problem (1), i.e., $\mathcal{X} := \{x \in \{0,1\}^n : \sum_{j\in V} x_j \leq k\}$, when $k = 1$. It is easy to see that in this case, the greedy algorithm is optimal. Note also that for fixed k, the problem is polynomially solvable (with respect to the input size of number of nodes, arcs and scenarios), because it involves evaluating $O(n^k)$ possible functions $\sigma_\omega(X)$, $\omega \in \Lambda$. Observe that, without loss of generality, we can assume that $p_\omega > 0$ for all $\omega \in \Lambda$ (otherwise, we can ignore scenario ω), and that $\sigma_\omega(\emptyset) = 0$ (otherwise, we can add a constant to the influence function). Furthermore, because $\sigma_\omega(\cdot)$ is submodular $\rho_j^\omega(\emptyset) \geq \rho_j^\omega(S)$ for any $S \subseteq V$, $S \neq \emptyset$ and $j \in V \backslash S$. As a result, if $\rho_j^\omega(\emptyset) < 0$, then $x_j = 0$ in any optimal solution. Therefore, without loss of generality, we can assume that $\rho_j^\omega(\emptyset) \geq 0$ for all $j \in V, \omega \in \Lambda$.

Proposition 3.1 *For submodular functions $\sigma_\omega(x), \omega \in \Lambda$ with $\rho_j^\omega(\emptyset) > 0$ for all $j \in V, \omega \in \Lambda$, and $\mathcal{X} := \{x \in \{0,1\}^n : \sum_{j\in V} x_j \leq 1\}$, adding the submodular optimality cut (4) with $\bar{X} = \emptyset$ to the linear programming (LP) relaxation of the master problem (3) for each $\omega \in \Lambda$ is sufficient to give the (integer) optimal solution x^*.*

Proof First, note that for $\bar{X} = \emptyset$, inequalities (4) and (5) are equivalent. Under the given assumptions, in an optimal solution $x \neq 0$, the right-hand side of (4) is positive for each $\omega \in \Lambda$. Therefore, the decision variables $\theta_\omega > 0$, $\omega \in \Lambda$, are basic variables at an extreme point optimal solution of the LP relaxation of the master problem (3). This gives us $|\Lambda|$ basic variables, and the number of constraints is $|\Lambda| + 1$. Hence, only one decision variable x_j for some $j \in V$ can be basic, and it is equal to 1 [due to constraint (6)], and $\theta_\omega = \rho_j^\omega(\emptyset) = \sigma_\omega(\{j\})$. Furthermore, this is the optimal solution to the master problem for the case $k = 1$. □

4 Application to the stochastic influence maximization problem

In this section, we specify how the general algorithm we propose for two-stage stochastic programs with submodular second-stage value functions applies to the influence maximization problems of interest. Kempe et al. [21] observe that even though the stochastic diffusion process of influence spread is dynamic, because the decisions of whom to activate do not influence the probability of an individual influencing another,

we may envision the process to be static and ignore the time aspect. In other words, we can generate sample paths (scenarios) of likely events for each arc, a priori. As a result, the decision-making process considered by Kempe et al. [21] may be viewed as a two-stage stochastic program. In the first stage, the nodes to be activated are determined. The uncertainty, represented by a finite collection of scenarios, Λ, is revealed with respect to how the influence spreads in the network. For each scenario $\omega \in \Lambda$, with associated probability p_ω, let the influence spread given the initial seed set X be given by $\sigma_\omega(X) := |\{j \in V : \exists \text{ a path from } i \text{ to } j \text{ in } G_\omega, i \in X\}|$, i.e., $\sigma_\omega(X)$ is the number of vertices reachable from X in G_ω. As a result, the expected total influence spread of the initial seed set X is given by $\sigma(X) = \sum_{\omega\in\Lambda} p_\omega \sigma_\omega(X)$. Let $x \in \{0, 1\}^n$ be the characteristic vector of $X \subset V$. Where appropriate, we use $\sigma(x)$ interchangeably with $\sigma(X)$.

As observed by Kempe et al. [21], the influence function $\sigma_\omega(X)$ is submodular and monotone (nondecreasing) for various influence maximization problems. Then the two-stage stochastic programming formulation of the classical influence maximization problem is given by (1) where $c_j = 0$ for al $j \in V$ and the set $\mathcal{X}$ defines the cardinality constraint on the number of seed nodes given by

$$\sum_{j\in V} x_j \leq k, \tag{6}$$

for a given $0 < k < |V|$. Therefore, Algorithm 2 can be used to solve the influence maximization problem. Furthermore, note that the influence functions of interest in this paper satisfy the assumption $\rho_j^\omega(\emptyset) > 0$ for all $j \in V, \omega \in \Lambda$, because influencing only node j contributes at least one node (itself) to the influence function. In addition, the first-stage problem is cardinality-constrained. Hence Proposition 3.1 applies to the influence functions considered in this paper.

To model the stochastic diffusion process and calculate the influence spread function, Kempe et al. [21] introduce a technique that generates a finite set, Λ, of sample paths (scenarios) by tossing biased coins. The coin tosses reveal, a priori, which influence arcs are active (live). A live-arc (i, j) indicates that if node i is influenced during the influence propagation process, then node j is influenced by it. For each scenario $\omega \in \Lambda$, with a probability of occurrence p_ω, a so-called *live-arc graph* $G_\omega = (V, A_\omega)$ is constructed, where A_ω is the set of *live* arcs under scenario ω. Then the influence spread under scenario $\omega \in \Lambda$ is calculated to obtain $\sigma_\omega(X)$. Hence, the expected influence spread function is given by $\sigma(X) = \sum_{\omega\in\Lambda} p_\omega \sigma_\omega(X)$. This is referred to as the "triggering model" or the "triggering set technique" by Kempe et al. [22]. The authors show the equivalence of the stochastic diffusion process of two fundamental influence maximization problems to the live-arc graph model with respect to the final active set. In addition, Kempe et al. [21] show that the influence spread in a live-arc graph representable problem is monotone and submodular under the given assumptions. As a result, our stochastic programming method applies to such problems. Next we describe the two fundamental influence maximization problems that are live-arc representable.

Independent Cascade Model In the independent cascade model of Kempe et al. [21], it is assumed that each arc $(i, j) \in A$ of the social network $G = (V, A)$ has an associated probability of success, π_{ij}. In other words, with probability π_{ij} individual i will be successful at influencing individual j. We say that an arc (i, j) is *active* or *live* in this case. We generate a sample path (scenario) by tossing biased coins (with probability of π_{ij} for each arc $(i, j) \in A$) to determine whether the arc is active/live to construct the live-arc graph. Because each arc influence probability is independent, and does not depend on which nodes are influenced, Kempe et al. [21] show that the influence maximization problem is equivalent to maximizing the expected influence function in the live-arc graph model.

Linear Threshold Model In the linear threshold model of Kempe et al. [21], each arc (i, j) in the social network $G = (V, A)$ has deterministic weight $0 \leq w_{ij} \leq 1$, such that for all nodes $j \in V$, $\sum_{i:(i,j)\in A} w_{ij} \leq 1$. In addition, each node $j \in V$ selects a threshold v_j *uniformly at random* between 0 and 1. A node is activated if sum of the weights of its *active* neighbors is above the thresholds, i.e., $\sum_{i:(i,j)\in A} w_{ij} x_i \geq v_j$. Given the set of initial seed nodes, $\bar{X}$, the activated nodes in the set U at time t influence their unactivated neighbor j at time $t + 1$ if $\sum_{u\in U} w_{uj} \geq v_j$. Kempe et al. [21] show that the linear threshold model also has an equivalent live-arc graph representation, where every node has at most one incoming live arc. Each node $j \in V$ selects at most one incoming live arc (i, j) with probability w_{ij}, or it selects no arc with probability $1 - \sum_{i:(i,j)\in A} w_{ij}$. Given the seed set $\bar{X}$, Kempe et al. [21] prove the following two are equivalent:

(i) The distribution of active nodes computed by executing the linear threshold model with starting seed set $\bar{X}$, and

(ii) the distribution of nodes reachable from $\bar{X}$ in the live-arc graph representation of the linear threshold model defined above.

Next, we demonstrate how the proposed algorithm (Algorithm 2) can be applied to influence maximization problems that have a live-arc graph representation. Subsequently, we give extensions where the proposed algorithm applies to models which are not live-arc graph representable. In such models, the form of the cuts change, but as long as the influence spread function is submodular, the proposed algorithm applies.

4.1 Exploiting the submodularity of the second-stage value function for live-arc graph models

Utilizing Theorem 3.1, we give an explicit description of the submodular inequalities for the influence maximization problems that have live-arc graph representations. We say that a node j is reachable from a set of nodes S, in scenario $\omega \in \Lambda$, if there exists a node $i \in S$ such that there is a directed path from i to j in the graph $G_\omega = (V, A_\omega)$. It is well known that reachability can be checked in linear time with respect to the number of arcs using depth- or breadth-first search. For $S \subseteq V$ and $\omega \in \Lambda$, let $R(S)$ be the set of nodes reachable from the nodes in S not including the nodes in S, and let $\bar{R}(S)$ be the set of nodes not reachable from the nodes in S in the graph $G_\omega = (V, A_\omega)$.

Proposition 4.1 *For $S \subseteq V$ and $\omega \in \Lambda$ the inequality*

$$\theta_\omega \leq \sigma_\omega(S) + \sum_{j \in \bar{R}(S)} r_j^\omega(S) x_j, \tag{7}$$

is a valid optimality cut for the master problem (3), where $r_j^\omega(S)$ is the number of nodes reachable from $j \in \bar{R}(S)$ (including j) that are not reachable from any node in S in G_ω.

Proof From Theorem 3.1, we know that $\theta_\omega \leq \sigma_\omega(S) + \sum_{j \in V \setminus S} \rho_j^\omega(S) x_j$ is a valid inequality. Note that $\bar{R}(S) \subseteq V \setminus S$ and for $j \in \bar{R}(S)$, we have $\rho_j^\omega(S) = r_j^\omega(S)$, in other words, the marginal contribution of adding $j \in \bar{R}(S)$ to S is precisely $r_j^\omega(S)$. Furthermore, for any node $j \in R(S)$, the marginal contribution of adding j to S is zero, because j is already reachable from at least one node in S. This completes the proof. □

We refer to the cuts in the form of (7) as *submodular optimality cuts*. Note that to obtain an inequality (7) for a given G_ω and S, we need to solve multiple reachability problems on the same graph, where time complexity of a single reachability problem by depth- or breadth-first search is $\mathcal{O}(|A_\omega|)$. We first compute $\sigma_\omega(S)$ by solving a reachability problem on G_ω and mark all nodes reachable from S. Then, for each $j \in \bar{R}(S) \setminus S$, we compute $r_j^\omega(S)$ by solving another reachability problem, where we count the number of unmarked nodes reachable from j. Hence the overall complexity of generating an inequality (7) is $\mathcal{O}(|A_\omega| \times |\bar{R}(S) \setminus S|)$.

Next we give conditions under which inequalities (7) are facet defining for $\text{conv}(\mathcal{S}_\omega)$.

4.2 Strength of the submodular inequalities

For $i \in V$, let $\text{indeg}(i)$ and $\text{outdeg}(i)$ denote the in-degree and out-degree of node i, respectively. Let $T := \{i \in V : \text{indeg}(i) = 0\}$, we refer to the nodes in T as *root nodes*. For $i \in V \setminus T$, let P_i be the set of root nodes such that i is reachable from the nodes in this set, i.e., $P_i := \{j \in T : i \in R(\{j\})\}$. Finally, let $L := \{i \in V : \text{indeg}(i) > 0, \text{outdeg}(i) = 0\}$ denote the set of *leaf nodes* that have no outgoing arcs.

First, note that the submodular inequality (7) for a set S is equivalent to that for the set $S \cup R(S) =: \hat{R}(S)$, because $\sigma_\omega(S) = \sigma_\omega(\hat{R}(S))$, $\bar{R}(S) = \bar{R}(\hat{R}(S))$, $r_j^\omega(S) = r_j^\omega(\hat{R}(S))$ for all $j \in \bar{R}(S)$ and $\rho_j^\omega(S) = 0$ for $j \in R(S)$. Therefore, in what follows, without loss of generality, we assume that for all non-leaf nodes $i \in S \setminus L$, we have $R(\{i\}) \subseteq S$ (Assumption **A1**).

Proposition 4.2 *For $S \subseteq V$ and $\omega \in \Lambda$ the submodular inequality (7) is facet defining for $\text{conv}(\mathcal{S}_\omega)$ only if the following conditions hold*

(i) if $i \in S$, then $i \notin T$,
(ii) there exists $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$.

These conditions are also sufficient

(i) if $S = \emptyset$ (for any $k \geq 1$), or
(ii) if $|S| = 1$ for $k \geq 2$.

Proof Necessity

(i) Suppose, for contradiction, that there exists $i \in S \cap T$. Now consider the submodular inequality (7) for the set $S' = S \backslash \{i\}$ given by

$$\theta_\omega \leq \sigma_\omega(S') + \sum_{j \in \bar{R}(S')} r_j^\omega(S') x_j = \sigma_\omega(S) - 1 + x_i + \sum_{j \in \bar{R}(S)} r_j^\omega(S) x_j, \tag{8}$$

which follows because the set of all descendants of i, $R(\{i\})$ is contained in S by Assumption **A1**, so removing i reduces the influence function by exactly 1 (recall that, by the contradictory assumption $i \in T$, hence its in-degree is 0 and it is not influenced by any other node in the graph), and the set of nodes not reachable from S' is given by $\bar{R}(S') = \bar{R}(S) \cup \{i\}$, and hence the coefficients $r_j^\omega(S') = r_j^\omega(S)$ for $j \in \bar{R}(S)$, and $r_i^\omega(S') = 1$. Because $x_i \leq 1$, inequality (8) dominates the submodular inequality (7) for this choice of S. Hence, the submodular inequality for a set S such that there exists $i \in S \cap T$ is not facet defining for $\text{conv}(\mathcal{S}_\omega)$.

(ii) Suppose, for contradiction, that there does not exist $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$. In other words, the minimum cardinality of root nodes $T' \subseteq T$ such that $S \subseteq R(T')$ is greater than or equal to k. In this case, consider the set $\hat{S} := \{i \in S : \nexists j \in S \text{ with } i \in R(\{j\})\}$, in other words, $\hat{S}$ is the set of nodes in the graph induced by S that have no incoming arcs from other nodes in S. Note that from condition (i), we know that $\hat{S} \cap T = \emptyset$. Then, by the contradictory assumption, there exist at least k nodes, say nodes $1, \ldots, k \in \hat{S}$ such that $P_i \cap P_j = \emptyset$ for all pairs $i, j \in \{1, \ldots, k\}, i \neq j$. Now consider the submodular inequality (7) for the set $S' = S \backslash \{1, \ldots, k\}$ given by

$$\theta_\omega \leq \sigma_\omega(S') + \sum_{j \in \bar{R}(S')} r_j^\omega(S') x_j = \sigma_\omega(S) - k + \sum_{j \in \bar{R}(S)} r_j^\omega(S) x_j + \sum_{i=1}^{k} \sum_{j \in \hat{R}(P_i) \backslash R(\{i\})} x_j, \tag{9}$$

which follows because the set of all descendants of $i \in \{1, \ldots, k\}$, $R(\{i\})$, is contained in S by Assumption **A1**, so removing nodes $i = 1, \ldots, k$ reduces the influence function by exactly k, and the set of nodes not reachable from S' is given by $\bar{R}(S') = \bar{R}(S) \cup \{1, \ldots, k\}$. In addition, the coefficients $r_j^\omega(S') = r_j^\omega(S)$ for $j \in \bar{R}(S)$ such that $j \notin \cup_{i=1}^k \left(\hat{R}(P_i) \backslash R(\{i\})\right)$, $r_j^\omega(S') = r_j^\omega(S) + 1$ for $j \in \bar{R}(S)$ such that $j \in \cup_{i=1}^k \left(\hat{R}(P_i) \backslash R(\{i\})\right)$, and $r_i^\omega(S') = 1$ for $i = 1, \ldots, k$. Because $\sum_{i=1}^k \sum_{j \in \hat{R}(P_i) \backslash R(\{i\})} x_j \leq \sum_{j \in V} x_j \leq k$, inequality (9) dominates the submodular inequality (7) for this choice of S. Hence, there must exist $T' \subseteq T$ with $|T'| < k$ such that $S \subseteq R(T')$ for the submodular inequality (7) to be facet defining for $\text{conv}(\mathcal{S}_\omega)$.

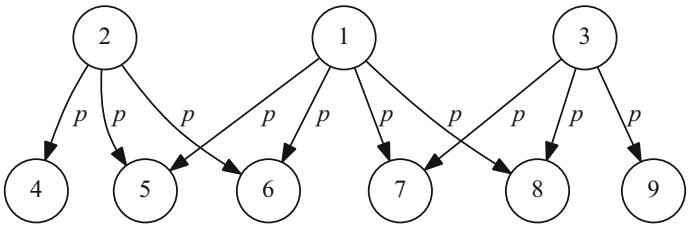

Fig. 1 Network with 9 nodes and 10 arcs with equal influence probabilities p

Sufficiency First, note that for $\omega \in \Lambda$, $\dim(\mathcal{S}_\omega) = n + 1$. Let $\mathbf{e}_i$ be a unit vector of dimension n whose ith component is 1, and other components are zero.

(i) Note that when $S = \emptyset$, the necessity conditions are trivially satisfied. Consider the $n + 1$ affinely independent points: $(\theta_\omega, x)^0 = \mathbf{0}$, and $(\theta_\omega, x)^i = (\sigma_\omega(\{i\}), \mathbf{e}_i)$, for $i \in V$. These points are on the face defined by the inequality (7) for $S = \emptyset$. Hence inequality (7) for $S = \emptyset$ is facet-defining for $\text{conv}(\mathcal{S}_\omega)$.

(ii) Note that for $|S| = 1$, the necessity conditions imply that $S := \{j\}$ for some $j \in L$. Consider the $n + 1$ affinely independent points: $(\theta_\omega, x)^0 = (\sigma_\omega(\{j\}), \mathbf{0})$; $(\theta_\omega, x)^j = (\sigma_\omega(\{j\}), \mathbf{e}_j$ and $(\theta_\omega, x)^i = (\sigma_\omega(\{i, j\}), \mathbf{e}_j + \mathbf{e}_i)$, for $i \in V \backslash \{j\}$. The last set of points is feasible because we have $k \geq 2$ in this case. These points are on the face defined by the inequality (7) for $S = \{j\}$. Hence inequality (7) for $S = \{j\}$ is facet-defining for $\text{conv}(\mathcal{S}_\omega)$. □

Note that during the course of the algorithm, if a submodular inequality (7) corresponding to the seed set S does not satisfy the necessary conditions given in Proposition 4.2, then a stronger inequality can be constructed using the arguments in the proof of the proposition.

From Proposition 4.2 we see that inequalities (7) with $S = \emptyset$ are facets of $\text{conv}(\mathcal{S}_\omega)$ for any $k \geq 1$. We will also see their importance in our computational study. Similarly, inequalities (7) with $|S| = 1$ are facets of $\text{conv}(\mathcal{S}_\omega)$ for any $k \geq 2$. We note that more conditions are necessary for the inequalities (7) with $|S| = 2$ to be facets of $\text{conv}(\mathcal{S}_\omega)$. We illustrate this in the next example.

Example 4.1 Consider the network in Fig. 1 for a given scenario $\omega \in \Lambda$ and let $k = 2$. From Proposition 4.2, inequalities (7) with $S = \emptyset$, and inequalities (7) with $S = \{j\}$, for $j = 4, \ldots, 9$ are facet-defining for $\text{conv}(\mathcal{S}_\omega)$. Inequalities (7) with $S = \{7, 8\}$ or $S = \{5, 6\}$ are facets of $\text{conv}(\mathcal{S}_\omega)$; each of these sets satisfies the necessary facet conditions in Proposition 4.2, which for these choices of S also turn out to be sufficient. However, the sets $S = \{7, 9\}$ or $S = \{4, 5\}$ satisfy the necessary facet conditions in Proposition 4.2, but they do not lead to facet-defining inequalities for $k = 2$. Finally, $S = \{4, 7\}$ violates the necessity condition (ii) of Proposition 4.2 (the minimum number of root nodes that can influence 4 and 7 is $2 = k$) and is not a facet.

It is important to note that in the direct adaptation of the delayed constraint generation algorithm proposed by Nemhauser and Wolsey [37] to our problem, for a given solution $\bar{x}$ to the current master problem, one would use the submodular inequalities (7), where we let $S = \{i \in V : \bar{x}_i = 1\} =: \bar{X}$. From Proposition 3.1, we have that

Algorithm 2 with optimality cuts (7) with $S = \bar{X}$ converges to an optimal solution in finitely many iterations for two-stage stochastic submodular maximization problems. However, note that at any iteration, the solution to the master problem, $\bar{x}$ will be such that $\sum_{j \in V} \bar{x}_j = k$. Therefore, all submodular optimality cuts (7) added will have $|S| = k$, and no facets with $S = \emptyset$ will be added for $k \geq 1$. This may lead to slow convergence of the delayed constraint generation algorithm with submodular optimality cuts for $S = \bar{X}$, which we illustrate next.

Example 4.1 (Continued.) Consider the network in Fig. 1, and suppose that $|\Lambda| = 1$, hence we consider a deterministic problem. Adding the submodular optimality cut (7) for $S = \emptyset$: $\theta_1 \leq 5x_1 + 4x_2 + 4x_3 + \sum_{j=4}^{9} x_j$ to the cardinality constraint, and solving the linear programming relaxation of the master problem yields the integer optimal solution, $\bar{x}_1 = 1, \theta_1 = 5$ at the first iteration (from Proposition 3.1). In contrast, solving the master problem without any optimality cuts (i.e., with just the cardinality constraint) may lead to an initial solution of $\bar{x}_2 = 1$. Then the following set of submodular cuts are added in that order during the course of the algorithm of Nemhauser and Wolsey [37]:

$$\theta_1 \leq 4 + 3x_1 + 4x_3 + \textstyle\sum_{j=7}^{9} x_j \quad (S = \bar{X} = \{2\}), \tag{10}$$

$$\theta_1 \leq 4 + 3x_1 + 4x_2 + \textstyle\sum_{j=4}^{6} x_j \quad (S = \bar{X} = \{3\}), \tag{11}$$

$$\theta_1 \leq 5 + 2x_2 + 2x_3 + x_4 + x_9 \quad (S = \bar{X} = \{1\}). \tag{12}$$

Furthermore, none of the inequalities (10)-(12) are facet-defining. Solving the LP relaxation of the master problem with the optimality cuts (10)-(12) leads to a fractional solution: $\bar{x}_2 = \bar{x}_3 = 0.5$. This small example highlights that a textbook implementation of the algorithm by Nemhauser and Wolsey [37] may lead to slow convergence because the algorithm (1) may explore, in the worst case, $O\binom{n}{k}$ many locally optimal solutions before finding an optimal solution, and (2) may require long solution times for each master problem, because the optimality cuts given by $S = \bar{X}$ may not be facet-defining and hence lead to weak LP relaxations.

These observations motivate us to devise a more efficient implementation of Algorithm 2, which we report in Sect. 5.

4.3 Extensions

In this section, we give various extensions of the influence maximization problems that can be solved using our proposed methods.

4.3.1 Extensions to live-arc graph models

Observe that while we demonstrate our general algorithm on the independent cascade and linear threshold models, our proposed model and method is applicable to many extensions of the social network problems studied in the literature. For example, an extension considered in the literature is to replace the cardinality constraint on the

number of nodes selected with a knapsack constraint representing a marketing budget where each node has a different cost to market. This model also admits an adapted and more involved 0.63-factor greedy approximation algorithm (see, [23,50]). In fact, our model is flexible enough to allow any constraints in $\mathcal{X}$ so long as the master problem can be solved with an optimization solver, while the greedy approximation algorithm needs careful adjustment and analysis for each additional constraint. Similarly, the time-constrained influence spread problem studied in Chen et al. [11] and Liu et al. [30] can also be solved using our method. In this problem, there is an additional constraint that the number of time periods it takes to influence a node should be no more than a given parameter τ. The resulting influence spread function is monotone and submodular, hence we can use inequalities (4) as the submodular optimality cuts. Furthermore, we can efficiently calculate the coefficients $\rho_j^{\omega}(\bar{X})$ by solving, with breadth-first search, a modified reachability problem limiting the number of hops from the seed set $\bar{X}$ to any other node by τ.

4.3.2 General cascade and general threshold models

In the *general cascade model*, every node $j \in V$ has an *activation function* $p'_j(i, S) \in [0, 1]$ for $S \subseteq \{(k, j) \in A\} =: N^{in}(j)$ and $i \in N^{in}(j) \backslash S$. The activation function represents the probability that node j is influenced by node i given that the nodes in S failed to activate node i. The independent cascade model is a special case, where $p'_j(i, S) = \pi_{ij}$, independent of S.

In the *general threshold model*, every node $j \in V$ has an *threshold function* $f_j(S)$ for $S \subseteq N^{in}(j)$, where $f_j(\cdot)$ is monotone and $f_j(\emptyset) = 0$. As before, every node j selects a threshold ν_j uniformly at random in the range $[0, 1]$. Then, a node j is activated if for a given active set S, $f_j(S \cap N^{in}(j)) \geq \nu_j$. The linear threshold model is a special case, where $f_j(S) = \sum_{i \in S} w_{ij}$.

Kempe et al. [21] show that general cascade model is equivalent to the general threshold model with an appropriate selection of activation and threshold functions. This is not true for the independent cascade and the linear threshold models (see Example 2.14 in [10]). Furthermore, the influence spread function is no longer submodular. However, if $f_j(S)$ is submodular for all $j \in V$, then the influence spread is submodular (first conjectured by Kempe et al. [21] and later proven by Mossel and Roch [33,34]). Therefore, the greedy hill climbing algorithm is a 0.63-approximation algorithm for this case as well. Algorithm 2 is applicable in the submodular threshold functions case, where the optimality cuts take the more general form (4) or (5) depending on the monotonicity of the function f.

5 Computational experiments

In this section we summarize our experience with solving the influence maximization problem using the delayed constraint generation method (DCG) with various optimality cuts as given in Algorithm 2, and the greedy hill-climbing algorithm (Greedy) of Kempe et al. [21] as given in Algorithm 1.

The algorithms are implemented in C++ with IBM ILOG CPLEX 12.6 Optimizer. All experiments were executed on a Windows Server 2012 R2 with an Intel Xeon E5-2630 2.40 GHz CPU, 32 GB DRAM and x64 based processor. In our implementation of Algorithm 2, we set the parameter $\varepsilon = 0$. For the master problem of the decomposition algorithm, the relative MIP gap tolerance of CPLEX was set to 1%, so a feasible solution which has an optimality gap of 1% is considered optimal.

5.1 Small-scale network

First, we study the quality of the solutions produced by DCG and Greedy on a small-scale network for which we can enumerate all possible outcomes of the random process. In these experiments, we are able to capture the random process precisely, and no information is lost through sampling from the true distribution. An illustrative network is given in Fig. 1 with 9 nodes, 10 directed arcs and independent influence probability $\pi_{ij} = p$ for all $(i, j) \in A$. Our goal is to select $k = 2$ seed nodes, so that the objective value, which is the expected number of nodes influenced by the seed nodes, is maximized. We generate all possible influence scenarios (a total of $2^{10} = 1024$ scenarios). Note that under the assumption that each influence is independent of the others, the probability of scenario ω, which has $\ell \leq 10$ live arcs, is given by $p_\omega = (1 - p)^{10-\ell} p^\ell$.

The solution of DCG and Greedy methods on 1024 scenarios with various values of $p = 0.1, 0.2, \ldots, 1$ is shown in Table 1. When $p \leq 0.5$, both algorithms have the same objective value. For $0.6 \leq p \leq 1$, Greedy selects node 1 as the seed in the first iteration of Algorithm 1 (line 4 of Algotihm 1) and selects either node 2 or 3 as the seed in the second iteration. However, DCG selects nodes 2 and 3 as the seed nodes, and provides a better objective value than Greedy (up to 12.5% improvement). So while Greedy does better than its worst-case bound (63%), it is within 12.5% of optimality.

Next, instead of generating all 1024 scenarios, we employed Monte-Carlo sampling, and independently sampled different number of scenarios $|\Lambda| = 10, 50$ and 100 according to different p values, and let $p_\omega = 1/|\Lambda|$. We summarize the results of this experiment in Table 2. For eight out of 15 cases, DCG has a higher objective value than Greedy, and in all other cases Greedy attains the optimal objective value (mostly for small influence probabilities $p = 0.1, 0.3$). We also observe that the objective value for the instances with a larger number of scenarios is generally closer to the objective value with all 1024 scenarios (except for $p = 0.1$ and $|\Lambda| = 100$). Note that Greedy is a 0.63-approximation algorithm even for the sampled problem, which assumes that the true distribution is given by the scenarios in Λ, whereas DCG provides the optimal solution to the sampled problem.

5.2 Large-scale network with real-world datasets

To evaluate the efficiency of DCG and Greedy on large networks, we conduct computational experiments on four real-world datasets with different categories and scales. These datasets are summarized below and in Table 3:

Table 1 Expected influence obtained from two algorithms for the small-scale network with 1024 scenarios

Algorithm	Objective values with different p									
	$p = 1.0$	$p = 0.9$	$p = 0.8$	$p = 0.7$	$p = 0.6$	$p = 0.5$	$p = 0.4$	$p = 0.3$	$p = 0.2$	$p = 0.1$
DCG	8	7.4	6.8	6.2	5.6	5	4.48	3.92	3.32	2.68
Greedy	7	6.68	6.32	5.92	5.48	5	4.48	3.92	3.32	2.68

Table 2 Expected influence obtained from two algorithms for the small-scale network with $|\Lambda|$ scenarios

$\|\Lambda\|$	Algorithm	Objective values for different p				
		$p = 0.9$	$p = 0.6$	$p = 0.5$	$p = 0.3$	$p = 0.1$
10	DCG	7.1	5.2	4.7	3.4	2.6
10	Greedy	6.8	5.1	4.6	3.4	2.6
50	DCG	7.4	5.84	5.18	3.98	2.68
50	Greedy	6.82	5.66	5.06	3.98	2.68
100	DCG	7.38	5.61	5.04	3.96	2.76
100	Greedy	6.69	5.52	5.04	3.96	2.76

Table 3 The summary of real world datasets

	Dataset			
	UCI-message	P2P02	Phy-HEP	Email-Enron
Network category	Online-message	File-shearing	Collaboration	Communication
Nodes	1899	10,876	15,233	36,692
Edges	59,835	39,994	58,891	183,831
Format	Directed	Undirected	Undirected	Directed

UCI-message is the dataset for the online student community network at the University of California, Irvine [39]. The 1,899 nodes represent the students, and the 59,835 directed arcs between two nodes indicate that one student sent a message to the other.

P2P02 is the dataset for the Gnutella peer-to-peer file sharing network from August 2002 [27,43]. The 10,876 nodes represent the hosts, and the 39,994 undirected edges denote the connections between two hosts.

Phy-HEP is the dataset for the academic collaboration network in the "high energy physics theory" (HEPT) section of the e-print arXiv (www.arxiv.org) [12,21,22]. The 15,233 nodes represent the authors, and the 58,891 undirected edges represent the co-authorship between each pair of authors in the "high energy physics theory" papers from 1991 to 2003. Note that this is the original dataset considered in Kempe et al. [21], and it is commonly used as a benchmark in comparing various algorithms for maximizing influence in social networks.

Email-Enron is the dataset for the email communication network of the Enron Corporation [24,29]. It is posted to the public by the Federal Energy Regulatory Commission during the investigation. The 36,692 nodes represent the different email addresses, and the 183,831 directed arcs denote one address sent a mail to the other.

Note that if the graph in the original datasets contain undirected edges between i and j, then we construct a directed graph with two directed arcs from i to j and j to i. We follow the data generation scheme of Kempe et al. [21] in constructing live-arc graphs for these instances (i.e., the probabilities of influence on each arc for the independent cascade model, and the arc weights and node thresholds for the linear threshold model follow from Kempe et al. [21]).

In our experiments in this subsection, we compare three algorithms: *Greedy* is the greedy hill-climbing algorithm (Algorithm 1); *DCG-SubIneqs* is Algorithm 2 using submodular optimality cuts (7) for $S = \bar{X}$, where $\bar{X}$ is the optimal solution given by the master problem in the current iteration; and *DCG-SubWarmup* adds the submodular inequalities (7) for $S = \emptyset$ for each scenario, before the execution of DCG-SubIneqs as a warm-start. Proposition 3.1 shows that the submodular optimality cut (7) with $S = \emptyset$, which is referred to as EmptySetCut, is sufficient to find the optimal solution for $k = 1$ (note that $\rho_j^{\omega}(\emptyset) \geq 1$ for all $j \in V, \omega \in \Lambda$, hence the assumptions of the proposition are satisfied). Since $k = 1$ is an easy case for DCG-SubWarmup due to Proposition 3.1, we include DCG-SubWarmup to test if EmptySetCut is also useful for $k > 1$. To verify this, we add EmptySetCuts for all scenarios to the master problem before executing the DCG algorithm, and solving the initial master problem. Note that the total computation time in our experiments includes the generation time of all EmptySetCuts, and the total number of user cuts also includes the number of EmptySetCuts. We also implemented Algorithm 2 using alternative optimality cuts (adapted and strengthened versions of integer-L-shaped cuts of [25], referred to as Benders-LC, and described in "Appendix A"); however, the running time of Benders-LC is extremely long. Therefore, we only report our results with DCG-SubIneqs, DCG-SubWarmup and Greedy, and discuss the inefficiency of Benders-LC in "Appendix A.1".

5.2.1 Independent cascade model

For the independent cascade model, we assign uniform influence probability $\pi_{ij} = p = 0.1$ independently to each arc (i, j) in the network as was done in Kempe et al. [21]. Note that Kempe et al. [21] consider the dataset **Phy-HEP** with influence probabilities $\pi_{ij} \in \{0.01, 0.1\}$ for each arc (i, j) in the network. However, we observe that for $\pi_{ij} = 0.01$, the total number of live arcs is very small, resulting in sparse live-arc graphs with a large number of singletons. For example, the expected number of live arcs in our largest dataset **Email-Enron** is $183,831 \times 0.01 = 1,838$ with $p = 0.01$. However, the number of nodes of **Email-Enron** is 36,692, resulting in over 30,000 singletons in the network. Therefore, we focus on the more interesting case of $\pi_{ij} = 0.1$ in our experiments in this section.

We generate $|\Lambda| = 100, 200, 300$ and 500 scenarios to find $k = 2$ to 5 seed nodes that maximize influence. Tables 4 and 5 summarize our experiments with

Table 4 Independent cascade model for UCI-message and P2P02

Datasets	k	$\|\Lambda\|$	DCG-SubIneqs		DCG-SubWarmup		Greedy
			Time (s)	Cuts (#)	Time (s)	Cuts (#)	Time (s)
UCI-message	2	100	75	200	72	200	40
	3	100	82	200	78	201	50
	4	100	76	200	71	200	59
	5	100	81	200	73	200	69
	2	200	90	399	85	400	77
	3	200	91	400	88	400	96
	4	200	95	400	84	400	118
	5	200	89	400	87	400	138
	2	300	356	596	260	600	133
	3	300	274	600	264	600	168
	4	300	268	600	282	600	201
	5	300	273	600	272	600	232
	2	500	237	994	201	1000	202
	3	500	221	999	207	1000	252
	4	500	219	1000	203	1000	299
	5	500	211	1000	200	1000	347
	Average		171.1	549.3	157.9	550.1	155.1
P2P02	2	100	466	200	505	212	364
	3	100	463	216	514	216	520
	4	100	721	245	591	244	682
	5	100	572	232	547	232	823
	2	200	526	400	514	400	558
	3	200	553	412	552	411	794
	4	200	569	433	552	428	1014
	5	200	785	451	590	451	1227
	2	300	240	600	246	600	1852
	3	300	246	600	251	600	2652
	4	300	324	658	295	643	3452
	5	300	310	678	302	658	4369
	2	500	522	1000	515	1000	4796
	3	500	489	1000	522	1000	6922
	4	500	609	1075	721	1193	9018
	5	500	608	1131	620	1129	11,043
	Average		500.2	583.2	489.8	588.6	3130.4

the algorithms DCG-SubIneqs, DCG-SubWarmup and Greedy for the independent cascade model. Column "k" denotes the number of seed nodes to be selected. Column "Cuts(#)" reports the total number of submodular inequalities (7) added to the master problem of DCG-SubIneqs, and column "Time(s)" reports the solution time in seconds.

Table 5 Independent cascade model for Phy-HEP and Email-Enron

Datasets	k	$\|\Lambda\|$	DCG-SubIneqs		DCG-SubWarmup		Greedy
			Time (s)	Cuts (#)	Time (s)	Cuts (#)	Time (s)
Phy-HEP	2	100	650	208	623	200	1141
	3	100	777	301	771	300	1654
	4	100	1064	385	1054	385	2107
	5	100	375	491	366	491	2530
	2	200	261	400	257	401	1593
	3	200	524	599	455	598	2185
	4	200	2208	770	1983	771	2770
	5	200	845	989	733	988	3344
	2	300	414	609	504	600	1123
	3	300	648	902	567	900	1561
	4	300	829	1040	863	1038	2004
	5	300	911	1190	737	1190	2465
	2	500	603	1000	614	1005	6250
	3	500	1266	1498	1353	1500	8992
	4	500	1544	1985	1434	1985	11,545
	5	500	2128	2220	2315	2323	13,808
	Average		940.4	911.7	914.9	917.2	4067
Email-Enron	2	100	1656	200	2004	200	7332
	3	100	1618	200	1721	200	9860
	4	100	1715	200	1618	200	12,465
	5	100	1622	200	1628	200	15,137
	2	200	4279	400	4262	400	11,623
	3	200	3372	400	3420	400	15,576
	4	200	3273	400	3668	400	19,414
	5	200	3265	400	3529	400	23,124
	2	300	5256	600	5037	600	17,675
	3	300	4890	600	5003	600	24,357
	4	300	4921	600	6116	726	31,387
	5	300	4900	600	5000	600	38,385
	2	500	9176	1000	8756	1000	25,970
	3	500	9726	1000	9305	1000	34,775
	4	500	9187	1000	9507	1000	43,588
	5	500	11,056	1000	9390	1000	52,548
	Average		4994.5	550	4997.8	557.9	23,951

We do not report the objective values in these experiments, because we are able to prove that despite its worst-case performance guarantee of 63%, Greedy is within the optimality tolerance for these instances. In Kempe et al. [21] Greedy is tested

empirically against other heuristics such as choosing the nodes with k highest degrees in the graph G, because it is said that an optimal solution is not available. Therefore, our computational experiments also provide an empirical test on the performance of the greedy heuristic when an optimal solution is available (to the sampled problem) due to our proposed method.

From column Cuts(#) in Tables 4 and 5, we observe that the number of cuts added to the master problem generally increases with the number of seed nodes k. In other words, more iterations are needed to prove optimality if we have more seed nodes to select. Columns DCG-SubIneqs Time(s) and Cuts(#) show that the overall running time does not necessarily increase with the number of user cuts, as more cuts may help the master problem converge to an optimal solution faster. Recall that the running time of DCG-SubIneqs includes the solution time of the master problem (a mixed-integer program) and the cut generation time of submodular inequalities, which decomposes by each scenario.

From column Greedy Time(s) we see that, for the same size of scenarios, the running time of Greedy increases linearly as the number of seed nodes increases. Recall that DCG solves a mixed integer master problem after the cut generation phase, which makes its running time nonlinear as we observe from the columns DCG-SubIneqs Time(s) and DCG-SubWarmup Time(s). The majority of the overall time for DCG is spent on cut generation for most instances (for example, the cut generation takes, on average, 80% of the time over all four problem instances with $|\Lambda| = 300$). Because we observe that the major bottleneck is the cut generation time and most of the remaining time is spent on the solution of the mixed-integer master problem, we did not implement other enhancements known to improve convergence of related Benders methods, such as the trust region method or heuristics (cf. [38,44]).

Considering the average solution time, we observe that there is not much difference between DCG-SubWarmup and DCG-SubIneq, and DCG outperforms Greedy for large networks with more than 10,000 nodes. For example, for the instance Email-Enron in Table 5, the average solution time of Greedy is five times that of DCG-SubIneqs. Only for the smallest instance, UCI-message, Greedy is the fastest algorithm (see Table 4).

5.2.2 Linear threshold model

In this section, we summarize our experiments with the linear threshold model. Recall that in the live-arc graph representation of linear threshold models, at most one incoming arc is chosen for each node in the live-arc graph construction for each scenario. As in Kempe et al. [21] we let the deterministic weight on each arc $(i, j) \in A$ be $w_{ij} = 1/\text{indeg}(j)$. We generate $|\Lambda| \in \{100, 200, 300, 500\}$ for the four real-world datasets described earlier.

The results are shown in Tables 6 and 7. Similar to the independent cascade model, the running time of Greedy increases linearly in k. As in the previous experiments for the independent cascade model, DCG-SubWarmup is slower than Greedy only for the smallest dataset with fewer than 2,000 nodes (UCI-message) (see Table 6). For the large-scale datasets with over 10,000 nodes (P2P02, Phy-HEP, and Email-Enron) reported in Tables 6 and 7, we observe that the warm-up strategy is highly effective.

Table 6 Linear threshold model for UCI-message and P2P02

Datasets	k	$\|\Lambda\|$	DCG-SubIneqs		DCG-SubWarmup		Greedy
			Time (s)	Cuts (#)	Time (s)	Cuts (#)	Time (s)
UCI-message	2	100	72	299	43	120	1
	3	100	94	378	70	169	2
	4	100	130	455	93	257	3
	5	100	162	606	89	303	4
	2	200	220	722	91	259	3
	3	200	221	735	80	296	4
	4	200	243	892	161	502	6
	5	200	469	1435	289	893	7
	2	300	188	590	200	513	5
	3	300	192	594	159	449	8
	4	300	573	1507	249	637	11
	5	300	543	1554	387	1011	13
	2	500	462	988	84	559	6
	3	500	747	1403	181	740	9
	4	500	665	1415	130	723	12
	5	500	1282	2909	462	1933	15
	Average		391.6	1030.1	173	585.3	6.8
P2P02	2	100	48	200	26	103	222
	3	100	55	200	28	108	337
	4	100	136	338	28	120	449
	5	100	230	580	31	133	565
	2	200	387	400	202	202	211
	3	200	583	596	225	214	311
	4	200	590	596	204	213	414
	5	200	1260	1156	306	290	512
	2	300	1564	600	584	308	1231
	3	300	1853	885	684	343	1833
	4	300	3529	1687	855	410	2531
	5	300	9557	4000	1656	765	3118
	2	500	1721	1000	576	503	1028
	3	500	1224	1000	534	512	1507
	4	500	4799	3308	600	552	1969
	5	500	10,505	5861	1047	887	2415
	Average		2377.6	1403.6	474.1	353.9	1165.8

It provides the best solution times, and fewer iterations and cuts. For example, DCG-SubWarmup outperforms Greedy by a factor of 2.46 in P2P02 in Table 6, a factor of 4.12 in Phy-HEP in Table 7, and a factor of 27 in Email-Enron in Table 7, the largest dataset considered.

Table 7 Linear threshold model for Phy-HEP and Email-Enron

Datasets	k	$\|\Lambda\|$	DCG-SubIneqs		DCG-SubWarmup		Greedy
			Time (s)	Cuts (#)	Time (s)	Cuts (#)	Time (s)
Phy-HEP	2	100	528	243	175	101	1116
	3	100	1117	428	333	182	1682
	4	100	1351	474	383	185	2235
	5	100	786	387	63	209	2740
	2	200	1141	586	477	359	1176
	3	200	1841	964	540	362	1791
	4	200	1448	779	602	363	2444
	5	200	1190	581	722	377	3229
	2	300	399	600	445	303	692
	3	300	489	600	516	312	1017
	4	300	943	871	1089	548	1329
	5	300	6425	3413	824	619	1642
	2	500	1910	1218	1229	887	4002
	3	500	2440	1421	1312	891	6342
	4	500	4340	2283	1635	988	8637
	5	500	9291	3845	2051	1040	10,993
	Average		2227.4	1168.3	774.8	482.9	3191.7
Email-Enron	2	100	206	200	97	100	3972
	3	100	196	200	131	106	5814
	4	100	414	379	158	137	7605
	5	100	644	535	299	230	9366
	2	200	1240	400	608	200	4732
	3	200	419	400	223	215	7001
	4	200	642	593	285	265	9192
	5	200	902	760	447	378	11,330
	2	300	740	600	335	300	9283
	3	300	666	600	375	326	13,602
	4	300	1082	886	436	365	17,833
	5	300	1975	1441	982	710	22,354
	2	500	1199	1000	613	500	12,113
	3	500	1192	1000	675	523	17,858
	4	500	2446	1929	845	669	23,443
	5	500	3490	2373	1044	844	28,876
	Average		1090.8	831	472.1	366.8	12,773.4

Some comments are in order for both the independent cascade and linear threshold models. First, we make some observations on increasing k. As can be seen from our experiments, the running time of Greedy increases linearly with k. However, the increase in the running time of DCG is nonlinear, as can be expected. Hence, as we

increase k, we need to set some time limits for both DCG and Greedy (currently, we impose no time limits). In this case, with DCG, we are still able to obtain an incumbent solution with k seed nodes and an estimate on optimality gap provided by the bound from the DCG master. However, with a time limit, we will have to stop Greedy prematurely, before it identifies all k seed nodes. For example, for the independent cascade model for the medium-sized instance P2P02, setting a time limit of one hour, for $k = 30$ and $|\Lambda| = 300$, Greedy stops at time limit with a solution that has $k = 4$ seed nodes (see Table 4). On the other hand, DCG stops with an incumbent solution that has $k = 30$ seed nodes and an optimality gap of 3.7%. For the largest instance Enron, from Table 5, we observe that Greedy cannot even find the first seed node in over three hours. Similarly, as we increase $|\Lambda|$, the running time of both algorithms increase greatly, and as in the case of increasing k we will have to impose time limits and stop Greedy prematurely to be able to compare the performance of the algorithms.

Note that, throughout the paper, we present a so-called multicut version of the DCG algorithm and its variants, where we add an optimality cut for each scenario at each iteration. We have also tested a single cut implementation, in which multiple cuts across all scenarios are aggregated into a single cut at each iteration. We observe a degraded performance of the single cut version for our problem instances; therefore, we present our results for the multicut approach. In particular, for the independent cascade model, the total computational time of the single cut version of DCG-SubWarmup is between 20 to 100% higher than the multicut version in all four datasets with 300 scenarios for $k > 1$ (See page 167 of [8], for a discussion on the problem-dependent nature of the performance of the single vs. multicut approach).

We remark that we implemented the basic greedy algorithm as proposed by Kempe et al. [21]. This implementation needs to perform the influence spread function evaluation for each node for each scenario at each iteration, resulting in a heavy running time of $\mathcal{O}(kn|\Lambda|m)$. Several improvements to this implementation using different data structures and algorithmic enhancements have been proposed (see, e.g., [12,28]). In particular, [12] propose the so-called Discount Greedy method and report that it solves the Phy-HEP instances within a second. In this paper, our main goal is not to compete with these efficient heuristics. Instead, we focus on solving the problem on large-scale datasets optimally in a reasonable time. Our computational experiments demonstrate an overlooked opportunity to use optimization methods to solve stochastic influence maximization problems to provable optimality.

6 Conclusion

In this paper, we propose a delayed constraint generation algorithm to solve influence maximization problems arising in social networks. We show that exploiting the submodularity of the influence function leads to strong optimality cuts. Furthermore, our computational experiments with large-scale real-world test instances indicate that the algorithm performs favorably against a popular greedy heuristic for this problem. In most instances, our algorithm finds a solution with provable optimality guarantees more quickly than a basic implementation of the greedy heuristic, which can only pro-

vide a 0.63 performance guarantee. Our algorithm is applicable to many other variants of the influence maximization problem for which the influence function is submodular. Furthermore, we generalize the proposed algorithm to solve any two-stage stochastic program, where the second-stage value function is submodular.

Our results on optimization-based methods for the fundamental influence maximization problems provide a foundation to build algorithms for more advanced models, such as the adaptive model of Seeman and Singer [45], where a subset of additional seed nodes is selected in the second stage based on the realization of some of the uncertain parameters and the seed nodes selected in the first stage. The decomposition methods of Sen [46]; Gade et al. [17] and Zhang and Küçükyavuz [57] can be employed in this case to convexify the second stage problems that involve binary decisions. Another possible future research direction is to develop optimization-based methods for the problem of marketing to nodes [21,22] to increase their probabilities of getting activated.

Acknowledgements We thank the coordinating editor and the two referees for their valuable comments that improved the presentation. This work is supported, in part, by the National Science Foundation Grants #1732364 and #1733001.

Appendix A: Alternative Benders optimality cuts for live-arc graph models

In this section, we present optimality cuts that can be obtained by traditional methods. First, we give an explicit linear programming (LP) formulation for the subproblems (2) used to calculate $\sigma_\omega(x)$ for live-arc graph models such as independent cascade or linear threshold. Observe that the maximum number of nodes reachable from nodes X (corresponding to the decision vector x) in graph G_ω can be formulated as a maximum flow problem an a modified graph $G'_\omega = (V \cup \{s, t\}, A'_\omega)$, where s is the source node, t is the sink node, and A'_ω includes the arcs A_ω and arcs (s, i) and (i, t) for all $i \in V$. Let the capacity of the arcs $(i, t), i \in V$ be one, and the capacity of arcs $(i, j) \in A_\omega$ be n (the maximum flow possible on any arc). In addition, we would like the arcs $(s, i), i \in V$ to have a capacity of n if $x_i = 1$ and 0 otherwise. Therefore, we let the capacity of arc (s, i) be nx_i. The reader might wonder why we create an arc (s, i) if a node i is not activated. To see why, note that in a two-stage stochastic programming framework, we need to build a second-stage model that is correct for any first-stage decision x. It is easy to see that the maximum flow on this graph is equal to the maximum number of vertices reachable from the seeded nodes X. The LP formulation of the second-stage problem for scenario $\omega \in \Lambda$ is

$$\sigma_\omega(x) = \max \sum_{i \in V} y_{si} \tag{13a}$$

$$\text{s.t.} \sum_{j:(j,i)\in A'_\omega} y_{ji} - \sum_{j:(i,j)\in A'_\omega} y_{ij} = 0, \quad i \in V \qquad (u_i^\omega) \tag{13b}$$

$$y_{si} \le nx_i, \quad i \in V \qquad (v_{si}^\omega) \tag{13c}$$

$$y_{ij} \le n, \quad (i, j) \in A_\omega \qquad (v_{ij}^\omega) \tag{13d}$$

$$y_{it} \leq 1, \quad i \in V \qquad (v_{it}^{\omega}) \qquad (13e)$$
$$y_{ij} \geq 0, \quad (i,j) \in A'_{\omega}, \qquad (13f)$$

where y_{ij} represents the flow on arc $(i,j) \in A'_{\omega}$, and the dual variables associated with each constraint are defined in parentheses. Note that the subproblems are feasible for any $\omega \in \Lambda$ and $x \in \{0,1\}^n$ (we can always send zero flows), therefore this problem is said to have *complete recourse*. The dual of the second-stage problem (13) is

$$\sigma_{\omega}(x) = \min \sum_{i \in V} (n x_i v_{si}^{\omega} + v_{it}^{\omega}) + \sum_{(i,j) \in A_{\omega}} n v_{ij}^{\omega} \qquad (14a)$$
$$\text{s.t. } u_i^{\omega} + v_{si}^{\omega} \geq 1, \quad i \in V \qquad (14b)$$
$$u_j^{\omega} - u_i^{\omega} + v_{ij}^{\omega} \geq 0, \quad (i,j) \in A_{\omega} \qquad (14c)$$
$$-u_i^{\omega} + v_{it}^{\omega} \geq 0, \quad i \in V \qquad (14d)$$
$$v_{ij}^{\omega} \geq 0, \quad (i,j) \in A'_{\omega}. \qquad (14e)$$

Note that we can write a large-scale mixed-integer program, known as the deterministic equivalent program (DEP), to solve the independent cascade problem. To do this, we create copies of the second-stage variables y_{ij}^{ω} for all $\omega \in \Lambda$, where y_{ij}^{ω} represents the flow on arc $(i,j) \in A'_{\omega}$ under scenario $\omega \in \Lambda$. The DEP is formulated as

$$\max \sum_{\omega \in \Lambda} p_{\omega} \sum_{i \in V} y_{si}^{\omega} \qquad (15a)$$
$$\text{s.t. } \sum_{j \in V} x_j \leq k \qquad (15b)$$
$$\sum_{j:(j,i) \in A'_{\omega}} y_{ji} - \sum_{j:(i,j) \in A'_{\omega}} y_{ij} = 0, \quad i \in V \qquad (15c)$$
$$y_{si}^{\omega} \leq n x_i, \quad \omega \in \Lambda, i \in V \qquad (15d)$$
$$y_{ij}^{\omega} \leq n, \quad \omega \in \Lambda, (i,j) \in A_{\omega} \qquad (15e)$$
$$y_{it}^{\omega} \leq 1, \quad \omega \in \Lambda, i \in V \qquad (15f)$$
$$x \in \{0,1\}^n, y_{ij}^{\omega} \geq 0, \omega \in \Lambda, (i,j) \in A'_{\omega}. \qquad (15g)$$

It is well-established in the stochastic programming field that due to its large size, it is not practical to solve DEP directly. Instead, as is commonly done, we consider the use of Benders decomposition method [5,51] utilizing the structure of this large-scale MIP.

A naive way of generating the optimality cuts is to solve the subproblem (13) for each $\omega \in \Lambda$ as an LP (in line 5 of Algorithm 2) to obtain $\sigma_{\omega}(\bar{x})$, and the corresponding dual vector $(\bar{u}^{\omega}, \bar{v}^{\omega})$. Then the optimality cut is

$$\theta_{\omega} \leq \sum_{i \in V} (n x_i \bar{v}_{si}^{\omega} + \bar{v}_{it}^{\omega}) + \sum_{(i,j) \in A_{\omega}} n \bar{v}_{ij}^{\omega}. \qquad (16)$$

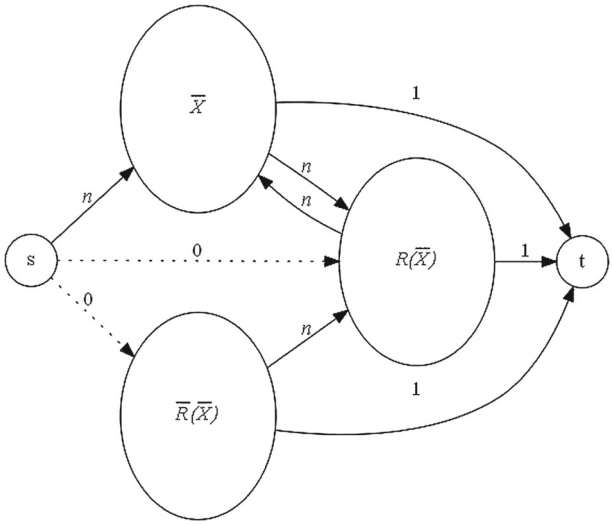

Fig. 2 Maximum flow formulation of the influence function

We refer to the optimality cuts (16) obtained by solving the subproblems as an LP as the *LP-based optimality cuts*.

Next, we discuss a more efficient way of obtaining the optimality cuts by utilizing the fact that the subproblems are maximum flow problems, which can be solved in polynomial time using specialized algorithms. In particular, for our problem, one only needs to solve a reachability problem to obtain the corresponding maximum flow. Reachability problem in a graph can be solved in linear time in the number of arcs using breadth- or depth-first search. We describe the equivalence of the maximum flow problem defining the evaluation of the influence spread to the graph reachability problem next.

For a given first-stage solution $\bar{x}$ and the corresponding seed set $\bar{X}$, let $\hat{R}(\bar{X}) \subseteq V$ be the set of nodes in V reachable from s, $R(\bar{X}) = \hat{R}(\bar{X}) \backslash \bar{X}$ be the set of nodes reachable from s not including the seed nodes $\bar{X}$, and $\bar{R}(\bar{X}) = V \backslash \hat{R}(\bar{X})$ be the set of nodes in V *not* reachable from s in G'_ω. From maximum flow minimum cut theorem (see, e.g., [2])) we can show that a minimum cut is given by $(\hat{R}(\bar{X}) \cup \{s\}, \bar{R}(\bar{X}) \cup \{t\})$. (See the maximum flow formulation of this problem for a given $\bar{X}$ and scenario $\omega \in \Lambda$ in Fig. 2.) Let $u_i^\omega = 1$ if $i \in \hat{R}(\bar{X})$, and $u_i^\omega = 0$, if $i \in \bar{R}(\bar{X})$. In addition, for $(i, j) \in A'_\omega$, let $v_{ij}^\omega = 1$ if $i \in \hat{R}(\bar{X}) \cup \{s\}$ and $j \in \bar{R}(\bar{X}) \cup \{t\}$, otherwise let $v_{ij}^\omega = 0$. It is easy to check that this choice of the dual variables is feasible. Furthermore, this choice is optimal. To see this, note that the objective value of the dual is

$$\sum_{i \in V} (n x_i \bar{v}_{si}^\omega + \bar{v}_{it}^\omega) + \sum_{(i,j) \in A_\omega} n \bar{v}_{ij}^\omega = \sum_{i \in \bar{R}(\bar{X})} n x_i + \sum_{i \in \hat{R}(\bar{X})} 1 + \sum_{(i,j) \in (\hat{R}(\bar{X}), \bar{R}(\bar{X}))} n$$
$$= |\hat{R}(\bar{X})|,$$

because $x_i = 0$ for $i \in \bar{R}(\bar{X})$ and there can be no arc $(i, j) \in A_\omega$ with $i \in \hat{R}(\bar{X})$, $j \in \bar{R}(\bar{X})$ (otherwise j would be reachable from s and hence it will be in $\hat{R}(\bar{X})$).

Because the optimal objective value of the primal subproblem is $\sigma_\omega(\bar{x}) = |\hat{R}(\bar{X})|$, this dual solution must be optimal. With this choice of the optimal dual vector, we obtain the Benders optimality cut

$$\theta_\omega \leq \sigma_\omega(\bar{x}) + \sum_{i \in \bar{R}(\bar{X})} n x_i . \tag{17}$$

We refer to the optimality cuts (17) obtained by solving the subproblems as reachability problems as *combinatorial optimality cuts*. Wallace [52] and Wollmer [54] use the same type of optimality cuts for a problem of investing in arc capacities of a network to maximize flow under stochastic demands. This problem is a relaxation of our problem in that the first stage variables are continuous. Hence, submodular inequalities cannot be used in their problem context.

Note that inequality (17) can also be seen as a big-M type inequality. For $x = \bar{x}$, with the associated seed set $\bar{X}$, we get a correct upper bound on θ_ω as $\sigma_\omega(x)$. For any other $x \neq \bar{x}$, if $x_i = 1$ for some $i \in \bar{R}(\bar{X})$, then the upper bound on θ_ω given by inequality (17) is trivially valid, because $\sigma_\omega(x) \leq n$ for any $x \in \{0, 1\}^n$. Finally, for any $x \neq \bar{x}$, if $x_i = 0$ for all $i \in \bar{R}(\bar{X})$, then we must have $x_j = 0$ for some $j \in \bar{X}$ and $x_\ell = 1$ from some $\ell \in R(\bar{X})$. However, because ℓ is reachable from $\bar{X}$, replacing j with ℓ will not increase the number of reachable nodes, i.e., $\sigma_\omega(x) \leq \sigma_\omega(\bar{x})$. Therefore, inequality (17) is valid.

Magnanti and Wong [32] propose a method to strengthen Benders cuts in cases when the dual of the subproblems is degenerate (see also, [41,48], for other enhancements of this method). The method chooses, among alternative dual optimal solutions to the subproblem, one that is not dominated. While this idea is useful to strengthen the weak Benders cut (16) (in particular, inequality (17) corresponding to one choice of optimal dual solutions), we note that it alone cannot lead to the stronger cuts given by the submodular inequalities (7). To see this note, first, that all extreme points of the dual subproblem (14) are integral. So any non-integral dual feasible solution is a convex combination of these extreme points. Then note that an optimality cut of the form (16) obtained from the dual is non-dominated only if the corresponding dual solution is an extreme point (otherwise the optimality cut would be a convex combination of the optimality cuts corresponding to the extreme points). As a result, $\bar{v}^\omega_{si} \in \mathbb{Z}$ for all $i \in V$, hence submodular inequalities (7) cannot be expressed as inequalities (16) obtained from non-dominated extreme point optimal dual solutions to the subproblem (14).

Finally, note that because the first-stage problem is a pure binary optimization problem, one can also consider the optimality cuts proposed in the integer L-shaped method of Laporte and Louveaux [25]. The resulting inequality, for $\omega \in \Lambda$ and a given $\bar{x}$, with an associated seed set $\bar{X}$, is

$$\theta_\omega \leq \sigma_\omega(\bar{x}) + \sum_{i \in V \setminus \bar{X}} (n - \sigma_\omega(\bar{x})) x_i . \tag{18}$$

This inequality can be strengthened by the same observation that replacing a node $j \in \bar{X}$ with a node $\ell \in R(\bar{X})$ does not increase the number of reachable nodes. Therefore, we can reduce the coefficient of x_ℓ in inequality (18) to obtain a strengthened

version of the integer L-shaped optimality cut (18):

$$\theta_\omega \leq \sigma_\omega(\bar{x}) + \sum_{i \in \bar{R}(\bar{X})} (n - \sigma_\omega(\bar{x}))x_i, \tag{19}$$

which is clearly valid. We refer to inequalities (19) as the *strengthened integer L-shaped optimality cuts*.

Proposition A.1 *The submodular optimality cuts* (7) *dominate the combinatorial optimality cuts* (19).

Proof This follows because $r_j^\omega(S) \leq n - \sigma_\omega(\bar{x})$ for any $j \in \bar{R}(S)$. □

A.1 Computations with Benders using strengthened L-shaped cuts

In our computational study in Sect. 5.2, we set $\pi_{ij} = p = 0.1$, $(i, j) \in A$ in the real world network. Because the influence probability p is very small, the live-arc graphs corresponding to each scenario are large-scale sparse networks. We were not able to solve even the smallest instances (with $k = 1$ and $|\Lambda| = 50$) using Benders-LC after one day. To demonstrate the inefficiency of Benders-LC, we consider a sparse network under one scenario, depicted in Fig. 3 with 15 nodes and 4 directed arcs, and compare the performance of DCG-SubIneqs and Benders-LC. In other words, we let $p_1 = 1$, which leads to a deterministic problem (i.e., a unique scenario with objective θ_1). We vary the value of k from 1 to 5.

The total number of user cuts added to the corresponding master problem is shown in Table 8. We observe that compared to DCG-SubIneqs the number of user cuts added

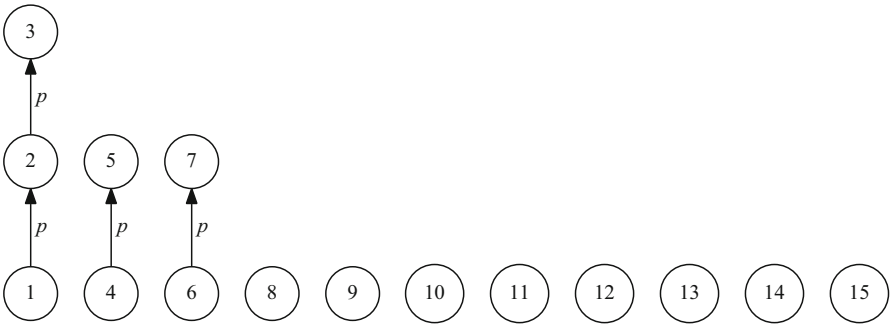

Fig. 3 Sparse network with 15 nodes and 4 arcs with equal influence probabilities p

Table 8 Comparison of DCG-SubIneqs and Benders-LC

Algorithm	Number of user cuts with different k				
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
DCG-SubIneqs	2	5	11	16	51
Benders-LC	15	106	458	1365	3003

to the master problem of Benders-LC grows rapidly as the number of seed nodes k increases. Indeed, the number of user cuts for Benders-LC approached $\binom{15}{k}$, indicating that Benders-LC is effectively a pure enumeration algorithm for this problem. The strengthened integer L-shaped optimality cuts (19) do not provide any useful information on the objective value when the solution is different from the one that generates the cut. In contrast, submodular inequalities are highly effective for this set of problems. To see why, consider the problem of finding $k = 1$ seed node. The master problem of both DCG-SubIneqs and Benders-LC selects $k = 1$ node arbitrarily, because they do not have any cut at the beginning. Because the sparse network is constituted of many singleton nodes (with no incoming and outgoing arcs), there is a high probability that the master problem selects one singleton at the first iteration. Suppose that the master problem chooses node 15, which was also the choice of CPLEX. DCG-SubIneqs generates the cut

$$\theta_1 \leq 1 + 3x_1 + 2x_2 + x_3 + 2x_4 + x_5 + 2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14},$$

and Benders-LC generates the cut

$$\theta_1 \leq 1 + \sum_{i=1}^{14} 14x_i,$$

to be added to the corresponding master problem. At the second iteration, due to the use of the stronger optimality cut, DCG-SubIneqs chooses node 1 and reaches optimality, but Benders-LC chooses one of the 14 nodes arbitrarily. Note that, in the worst case, Benders-LC traces all 15 nodes in the network (and generates 15 optimality cuts) before reaching the optimal solution. Therefore, in the large-scale network of Sect. 5.2, Benders-LC fails due to the need for a large number of iterations and computational time. In contrast, the submodular inequality guides the master problem to choose nodes with higher marginal influence.

References

1. Ahmed, S., Atamtürk, A.: Maximizing a class of submodular utility functions. Math. Program. **128**(1), 149–169 (2011)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall Inc, Upper Saddle River (1993)
3. Arulselvan, A., Commander, C.W., Elefteriadou, L., Pardalos, P.M.: Detecting critical nodes in sparse graphs. Comput. Oper. Res. **36**(7), 2193–2200 (2009)
4. Balasundaram, B., Butenko, S., Hicks, I.V.: Clique relaxations in social network analysis: the maximum k-plex problem. Oper. Res. **59**(1), 133–142 (2011)
5. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**(1), 238–252 (1962)
6. Bertsimas, D., Tsitsiklis, J.: Introduction to Linear Optimization, 1st edn. Athena Scientific, Belmont, Massachusetts (1997)
7. Bimpikis, K., Ozdaglar, A., Yildiz, E.: Competitive targeted advertising over networks. Oper. Res. **64**(3), 705–720 (2016). (article in advance)

8. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer, New York (1997)
9. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing social influence in nearly optimal time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '14, pp. 946–957. SIAM (2014)
10. Chen, W., Lakshmanan, L.V., Castillo, C.: Information and influence propagation in social networks. Synth. Lect. Data Manag. **5**(4), 1–177 (2013)
11. Chen, W., Lu, W., Zhang, N.: Time-critical influence maximization in social networks with time-delayed diffusion process. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence (2012)
12. Chen, W., Wang, Y., and Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, New York, NY, USA, pp. 199–208. ACM (2009)
13. Contreras, I., Fernández, E.: Hub location as the minimization of a supermodular set function. Oper. Res. **62**(3), 557–570 (2014)
14. Cornuéjols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manag. Sci. **23**(8), 789–810 (1977)
15. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, New York, NY, USA, pp. 57–66. ACM (2001)
16. Ertem, Z., Veremyev, A., Butenko, S.: Detecting large cohesive subgroups with high clustering coefficients in social networks. Soc. Netw. **46**, 1–10 (2016)
17. Gade, D., Küükyavuz, S., Sen, S.: Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. Math. Program. **144**(1–2), 39–64 (2014)
18. Günne, D., Raghavan, S., Zhang, R.: Tailored incentives and least cost influence maximization on social networks. Technical report (2016)
19. Hines, P., Balasubramaniam, K., Sanchez, E.: Cascading failures in power grids. IEEE Potentials **28**(5), 24–30 (2009)
20. Kawahara, Y., Nagano, K., Tsuda, K., Bilmes, J.A.: Submodularity cuts and applications. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems, NIPS '09, pp. 916–924 (2009)
21. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, New York, NY, USA, pp. 137–146. ACM (2003)
22. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. Theory Comput. **11**(4), 105–147 (2015)
23. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Inf. Process. Lett. **70**(1), 39–45 (1999)
24. Klimt, B., Yang, Y.: Introducing the Enron corpus. In: First Conference on Email and Anti-Spam, CEAS '04 (2004)
25. Laporte, G., Louveaux, F.: The integer L-shaped method for stochastic integer programs with complete recourse. Oper. Res. Lett. **13**(3), 133–142 (1993)
26. Lee, H., Nemhauser, G.L., Wang, Y.: Maximizing a submodular function by integer programming: polyhedral results for the quadratic case. Eur. J. Oper. Res. **94**(1), 154–166 (1996)
27. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. ACM Trans. Knowl. Discov. Data **1**(1), 2 (2007a)
28. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, New York, NY, USA, pp. 420–429. ACM (2007b)
29. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math. **6**(1), 29–123 (2009)
30. Liu, B., Cong, G., Xu, D., Zeng, Y.: Time constrained influence maximization in social networks. In: 2012 IEEE 12th International Conference on Data Mining (ICDM), pp. 439–448 (2012)
31. Madar, N., Kalisky, T., Cohen, R., Ben-Avraham, D., Havlin, S.: Immunization and epidemic dynamics in complex networks. Eur. Phys. J. B Condens. Matter Complex Syst. **38**(2), 269–276 (2004)
32. Magnanti, T.L., Wong, R.T.: Accelerating benders decomposition: algorithmic enhancement and model selection criteria. Oper. Res. **29**(3), 464–484 (1981)

33. Mossel, E., Roch, S.: On the submodularity of influence in social networks. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, STOC '07, New York, NY, USA, pp. 128–134. ACM (2007)
34. Mossel, E., Roch, S.: Submodularity of influence in social networks: from local to global. SIAM J. Comput. **39**(6), 2176–2188 (2010)
35. Nemhauser, G., Wolsey, L.: Maximizing submodular set functions: formulations and analysis of algorithms. In: Hansen, P. (ed.) Studies on Graphs and Discrete Programming. Annals of Discrete Mathematics, vol. 59, pp. 279–301. North-Holland Mathematics Studies, North-Holland (1981)
36. Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functions-I. Math. Program. **14**(1), 265–294 (1978)
37. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
38. Oliveira, F., Grossmann, I., Hamacher, S.: Accelerating benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. Comput. Oper. Res. **49**, 47–58 (2014)
39. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. Soc. Netw. **31**(2), 155–163 (2009)
40. Ostfeld, A., Salomons, E.: Optimal layout of early warning detection stations for water distribution systems security. J. Water Resour. Plan. Manag. **130**(5), 377–385 (2004)
41. Papadakos, N.: Practical enhancements to the Magnanti–Wong method. Oper. Res. Lett. **36**(4), 444–449 (2008)
42. Raghavan, S., Zhang, R.: Weighted target set selection on social networks. Technical report (2015)
43. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the gnutella network: properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Comput. **6**(1), 50–57 (2002)
44. Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A.: A stochastic programming approach for supply chain network design under uncertainty. Eur. J. Oper. Res. **167**(1), 96–115 (2005)
45. Seeman, L., Singer, Y.: Adaptive seeding in social networks. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 459–468 (2013)
46. Sen, S.: Stochastic mixed-integer programming algorithms: beyond Benders' decomposition. In: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley, New York (2010)
47. Shapiro, A., Dentcheva, D., Ruszczyński, A.: Lectures on Stochastic Programming: Modeling and Theory. Society for Industrial Mathematics, Philadelphia (2009)
48. Sherali, H.D., Lunday, B.J.: On generating maximal nondominated Benders cuts. Ann. Oper. Res. **210**(1), 57–72 (2013)
49. Song, Y., Dinh, T.N.: Optimal containment of misinformation in social media: a scenario-based approach. In: Zhang, Z., Wu, L., Xu, W., Du, D.-Z. (eds.) Combinatorial Optimization and Applications: 8th International Conference Proceedings, COCOA 2014, Wailea, Maui, HI, USA, December 19–21, 2014, pp. 547–556. Springer, Cham (2014)
50. Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint. Oper. Res. Lett. **32**(1), 41–43 (2004)
51. Van Slyke, R., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J. Appl. Math. **17**(4), 638–663 (1969)
52. Wallace, S.W.: Investing in arcs in a network to maximize the expected max flow. Networks **17**(1), 87–103 (1987)
53. Wang, C., Chen, W., Wang, Y.: Scalable influence maximization for independent cascade model in large-scale social networks. Data Min. Knowl. Discov. **25**(3), 545–576 (2012)
54. Wollmer, R.D.: Investments in stochastic maximum flow networks. Ann. Oper. Res. **31**(1), 457–467 (1991)
55. Xanthopoulos, P., Arulselvan, A., Boginski, V., Pardalos, P.: A retrospective review of social networks. In: Social Network Analysis and Mining, 2009, ASONAM '09, International Conference on Advances in, pp. 300–305 (2009)
56. Yu, J., Ahmed, S.: Maximizing a class of submodular utility functions with constraints. Math. Program. **162**(1–2), 145–164 (2017)
57. Zhang, M., Küükyavuz, S.: Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. SIAM J. Optim. **24**(4), 1933–1951 (2014)