Theoretical Computer Science 852 (2021) 84–103

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Influence maximization in the presence of vulnerable nodes: A ratio perspective

Huiping Chen[a], Grigorios Loukides[a,*], Solon P. Pissis[b,c], Hau Chan[d]

[a] Dept. of Informatics, King's College London, London, UK
[b] CWI, Amsterdam, the Netherlands
[c] Vrije Universiteit, Amsterdam, the Netherlands
[d] Dept. of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, USA

A R T I C L E I N F O

Article history:
Received 6 August 2019
Received in revised form 20 March 2020
Accepted 8 November 2020
Available online 20 November 2020

Keywords:
Influence maximization
Social networks
Vulnerable nodes

A B S T R A C T

Influence maximization is a key problem seeking to identify users who will diffuse information to influence the largest number of other users in a social network. A drawback of the influence maximization problem is that it could be socially irresponsible to influence users many of whom would be harmed, due to their demographics, health conditions, or socioeconomic characteristics (e.g., predominantly overweight people influenced to buy junk food). Motivated by this drawback and by the fact that some of these vulnerable users will be influenced inadvertently, we introduce the problem of finding a set of users (*seeds*) that limits the influence to vulnerable users while maximizing the influence to the non-vulnerable users. We define a measure that captures the quality of a set of seeds as an *additively smoothed ratio* (*ASR*) between the expected number of influenced non-vulnerable users and the expected number of influenced vulnerable users. Then, we develop methods which aim to find a set of seeds that maximizes the measure: greedy heuristics, an approximation algorithm, as well as several variations of the approximation algorithm. We evaluate our methods on synthetic and real-world datasets and demonstrate they substantially outperform a state-of-the-art competitor in terms of both effectiveness and efficiency. We also demonstrate that the variations of our approximation algorithm offer different trade-offs between effectiveness and efficiency.

1. Introduction

There has been an increased interest from public and private sectors and organizations in leveraging social networks to spread information of adopting certain behavior (e.g., for buying computer tablets or alcoholic beverages) [1–8]. A typical methodology of an organization is to influence few carefully selected users (or *seeds*), through free gifts, discounts, and information sessions, to adopt the desirable behavior (e.g., by posting a picture of the advertised computer tablet or alcoholic beverage) [9]. The hope is that these seeds will influence other users in their social circles to adopt the same behavior, and the subsequent influenced users will influence others in their respective social circles. As the information propagates throughout the social network, eventually some number of users will adopt the desirable behavior.

As a result, the organization's goal is to select a set of k seeds which maximize the largest expected number of adoptions (or *spread*) of *all the users* in the social network. This problem is known as the *influence maximization* problem in social

* Corresponding author.
E-mail addresses: huiping.chen@kcl.ac.uk (H. Chen), grigorios.loukides@kcl.ac.uk (G. Loukides), solon.pissis@cwi.nl (S.P. Pissis), hchan3@unl.edu (H. Chan).

https://doi.org/10.1016/j.tcs.2020.11.020
0304-3975/© 2020 Elsevier B.V. All rights reserved.

networks [1] and has been widely studied in the recent decade [9]. A main drawback of influence maximization is that it could be socially irresponsible to influence users many of whom could be harmed due to their demographics, health conditions, or socioeconomic profile [10]. The users who could be harmed are referred to as *vulnerable* and can be identified based on domain knowledge (e.g., user message content and sentiment analysis) [11,8]. For example, when an organization aims to promote alcoholic beverages, it should avoid influencing users many of whom have drinking problems. Similarly, when it aims to promote junk food, it should avoid influencing users many of whom are overweight. This is important for performing socially responsible influence maximization [12], which benefits not only the vulnerable users but also the companies, because most users are often willing to pay more for products marketed in a socially responsible way [13]. Motivated by the presence of vulnerable users, we initiate the study of influence maximization in social networks with both vulnerable and non-vulnerable users. In particular, we consider the problem of finding a set of seeds that limit the influence to vulnerable users while maximizing the influence to the non-vulnerable users in social networks. Our goal is to ensure that many non-vulnerable users are influenced, which is the main reason for which an organization spreads information in social networks, without influencing many vulnerable users. A solution to our problem may lead to influencing some vulnerable users, due to the diversity of social networks and the connections between vulnerable and non-vulnerable users. Yet, this can easily be avoided by a post-processing step, in which edges to vulnerable nodes are deleted [8].

Contribution. Our work makes the following specific contributions.

(1) Influence Measure. To deal with influence maximization in our setting, we need a measure to quantify the quality of a set S of seeds (or *seed-set*). The measure should ideally: (I) consider both vulnerable and non-vulnerable users, (II) limit influencing users many of whom are vulnerable, and (III) allow constructing a seed-set with guaranteed quality in our setting. We first examine the following natural measures and show that they are inappropriate to be used for influence maximization in our setting: (a) the difference $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$ and (b) the ratio $\frac{\sigma_{\mathcal{N}}(S)}{\sigma_{\mathcal{V}}(S)}$, where S is a seed-set and $\sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S)$ are the expected number of influenced non-vulnerable users and the expected number of vulnerable users, respectively. Then, we propose an *additively smoothed ratio* (*ASR*) measure $\frac{\sigma_{\mathcal{N}}(S)+c}{\sigma_{\mathcal{V}}(S)+c}$, where $c > 0$ is a specified constant. We show that *ASR* satisfies all the aforementioned properties I, II, and III and examine the impact of c in our influence maximization setting. Thus, our problem becomes finding a seed-set S of size at most k that maximizes *ASR*. This is a challenging problem because, as we show, *ASR* is non-monotone and neither submodular nor supermodular, which implies that it cannot be approximated through algorithms for submodular or supermodular maximization directly [14,7,15].

(2) Baseline Heuristics for Finding an ASR*-Maximizing Seed-set.* We develop a natural greedy heuristic (*GR*) that finds a seed-set of size at most k and large *ASR* iteratively. In each iteration, *GR* selects as seed a non-vulnerable node which influences a large (expected) number of additional non-vulnerable nodes for a small (expected) number of additional vulnerable nodes. *GR* is inspired by the *GreedRatio* framework [16] for maximizing a ratio of two submodular functions, because *ASR* is the ratio of the functions $\sigma_{\mathcal{N}}(S) + c$ and $\sigma_{\mathcal{V}}(S) + c$, which are submodular since $\sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S)$ are submodular [17] and the addition of a constant does not change submodularity [18]. Different from *GreedRatio* though, *GR* finds a seed-set of bounded size, which is necessary because influencing seeds entails monetary costs to an organization (e.g., for free gifts) that need to be controlled as the organization has limited budget. We then develop GR_{MB}, a variation of *GR* that estimates the spread efficiently, by considering paths from seeds to other nodes that amount for a "large" fraction of the spread, instead of considering all paths as *GR* does.

(3) Approximation Algorithm for Finding an ASR*-Maximizing Seed-set and its Variations.* We design *SAS* (Sandwich Approximation algorithm with Spread bounds), an efficient approximation algorithm for finding a seed-set to maximize *ASR*. Since *ASR* is not submodular, *SAS* cannot approximately maximize it directly. Instead, *SAS* constructs three candidates seed-sets (one with *ASR*, another with a submodular lower bound function of *ASR*, and a third with a submodular upper bound function of *ASR*) and selects the best candidate seed-set with respect to *ASR*. To efficiently construct each candidate seed-set, *SAS* creates a uniform random sample of the non-vulnerable nodes and adds into the candidate seed-set the node from the sample that yields the maximum marginal gain in the function (i.e., *ASR* or a bound function of *ASR*). We also propose a heuristic, called *ISS* (Iterative Subsample with Spread bounds), that follows the main principle of *SAS* but uses tighter lower and upper bound functions of *ASR*, which are non-monotone and non-submodular. *ISS* works iteratively, aiming to increase the *ASR* of the final seed-set. In addition, we propose two variations of *ISS*, namely $ISS^{\mathbf{U}}$, and $ISS^{\mathbf{Gr}}$, which explore differing trade-offs between efficiency and effectiveness. $ISS^{\mathbf{U}}$ aims to maximize only the upper bound of *ASR* that is used in *ISS*; it performs worse than *ISS*, but it is much more efficient. $ISS^{\mathbf{Gr}}$ differs from *ISS* in that it selects seeds from the entire set of non-vulnerable nodes, instead of a random sample of this set. $ISS^{\mathbf{Gr}}$ outperforms *ISS* in terms of *ASR* in practice but is less efficient.

(4) Experimental Evaluation. Our experiments using three publicly available datasets from Twitter, a political blog website, and Wikipedia, as well as a synthetic dataset, show that *SAS* and *ISS* outperform a state-of-the-art heuristic [6] that is based on the difference $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$, as well as our *GR* and GR_{MB} methods. For example, *ISS* constructed seed-sets that had on average 9 times larger *ASR* compared to those constructed by the heuristic in [6], while it was also 3 orders of magnitude more efficient. The experiments also show that *ISS* achieves a good trade-off between effectiveness and efficiency. It is 5 times faster than the best-performing $ISS^{\mathbf{Gr}}$ method and only slightly worse in terms of *ASR*, while it is slower than the most-efficient $ISS^{\mathbf{U}}$ method but 2 times better than $ISS^{\mathbf{U}}$ on average in terms of *ASR*.

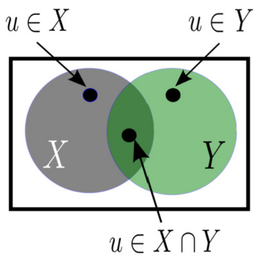


Fig. 1. Illustration of the computation of Equation (1).

This work updates and extends a preliminary work that was presented at [19].[1] In this version, we provide a more complete exposition of the underlying theoretical ideas of our approach. This includes identifying non-trivial connections between our algorithms and algorithms for maximizing general submodular functions (monotone and non-monotone) as well as non-submodular functions, showing that the problem addressed in our work is NP-hard and proposing the *SAS* approximation algorithm. In addition, we propose two new heuristics, *ISS*$^{\mathbf{U}}$ and *ISS*$^{\mathbf{Gr}}$. Lastly, we present a thorough experimental evaluation of the proposed algorithms.

Paper organization. Section 2 provides the necessary background. Section 3 introduces measures for quantifying the quality of a seed-set in our setting and defines the problem we address. Section 4 discusses the greedy baselines we propose. Section 5 and 6 discusses the *SAS* and *ISS* algorithm, respectively, while 7 discusses different variations of *ISS*. Section 8 presents our experimental evaluation. Section 9 discusses related work. Section 10 concludes the paper.

2. Preliminaries

In this section, we first define some preliminary concepts about submodular functions and discuss the Independent Cascade (IC) model [1] employed in our work. After that, we briefly discuss algorithms for optimizing submodular and non-submodular functions.

2.1. Submodular functions

Let U be a universe of elements and 2^U be its power set. A function $f : 2^U \to \mathbb{R}$ is *monotone*, if $f(X) \leq f(Y)$ for all subsets $X \subseteq Y \subseteq U$, and *non-monotone* otherwise.

A function $f : 2^U \to \mathbb{R}$ is *submodular*, if it satisfies the *diminishing returns* property: $f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y)$, for all $X \subseteq Y \subseteq U$ and any $u \in U \setminus Y$ [18]. If the property holds with equality, then f is called *modular*. A function $f : 2^U \to \mathbb{R}$ is *supermodular* if and only if $-f$ is submodular [18]. A modular function $f : 2^U \to \mathbb{R}$ is both submodular and supermodular. For brevity, we may write $f(X|u)$ for the *marginal gain* $f(X \cup \{u\}) - f(X)$.

Let $f : 2^U \to \mathbb{R}_{\geq 0}$ be a submodular function. For any $Y \subseteq U$, the *modular upper bound* $\widehat{f_Y}(X)$ of $f(X)$ is a modular function [20]

$$\widehat{f_Y}(X) = f(Y) + \sum_{u \in X \setminus Y} (f(\{u\}) - f(\{\})) - \sum_{u \in Y \setminus X} (f(Y) - f(Y \setminus \{u\})). \tag{1}$$

Y is referred to as the *parameter* of the bound.

The intuition behind the computation of Equation (1) is as follows (see Fig. 1). Instead of computing f for a set $X \subseteq U$, we compute f for a different set Y and take into account each element in $X \setminus Y$ and each element in $Y \setminus X$. An element $u \in X \setminus Y$ contributed to the value of $f(X)$ but does not contribute to the value of $f(Y)$. Therefore, in the bound calculation, we make a worst case assumption for the difference between $f(X)$ and $f(Y)$ that is caused by not taking into account u and add into $f(Y)$ the maximum possible contribution of u to $f(X)$. That is, we assume that the marginal gain of u is $f(\{u\}) - f(\{\})$, which is maximum due to the submodularity of f, and add the marginal gain into $f(Y)$. Doing this for every $u \in X \setminus Y$ is equivalent to adding the first sum in Equation 1 into $f(Y)$. On the contrary, an element $u \in Y \setminus X$ contributes to the value of $f(Y)$ but did not contribute to the value of $f(X)$. Therefore, in the bound calculation, we make a worst case assumption for the difference between $f(X)$ and $f(Y)$ by taking into account $u \in Y \setminus X$ and subtract from $f(Y)$ the minimum possible contribution of u to $f(Y)$. That is, we assume that the marginal gain of u is $f(Y) - f(Y \setminus \{u\})$, which is minimum due to the submodularity of f, and subtract the marginal gain from $f(Y)$. Doing this for every $u \in Y \setminus X$ is equivalent to subtracting the second sum in Equation 1 from $f(Y)$. Due to the assumptions in the calculation of the sums in Equation 1, we may overestimate the contribution of the elements in $X \setminus Y$ (for the first sum) and may underestimate the contribution of the elements in $Y \setminus X$ (for the second sum). Therefore, the right hand side of Equation (1) is at least equal to $f(X)$ (equal when $Y = X$), and $\widehat{f_Y}(X)$ is an upper bound of $f(X)$.

[1] It was also presented at the London Stringology Days & London Algorithmic Workshop 2019.

For any $Y \subseteq U$, the *modular lower bound* $\widetilde{f_{Y,\pi^Y}}(X)$ of $f(X)$ is a modular function [20]

$$\widetilde{f_{Y,\pi^Y}}(X) = \sum_{u \in X} f_{Y,\pi^Y}(u). \tag{2}$$

Y is referred to as the *parameter* of the bound, π^Y is a random permutation of the elements of Y (i.e., one-to-one mapping of Y onto itself), and

$$f_{Y,\pi^Y}(u) = \begin{cases} f(\pi^Y_u) - f(\pi^Y_{u^-}), & \text{if } u \in Y \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\pi^Y_{u^-}$ is the prefix of π^Y comprised of all elements of π^Y that appear before u in π^Y, and π^Y_u is the prefix of π^Y comprised of all elements of $\pi^Y_{u^-}$ and u.

The intuition behind the computation in Equation (2) is to use a random permutation of the elements of Y instead of the elements of X and exclude the contribution of the elements of X that are not in Y. Therefore, the right hand side of Equation (2) is at most equal to $f(X)$ (equal when $Y = X$) and $\widetilde{f_{Y,\pi^Y}}(X)$ is a lower bound of $f(X)$.

Let $f : 2^U \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. The (submodular) curvature of f is defined as $\kappa_f = 1 - \min_{u \in U} \frac{f(U) - f(U \setminus \{u\})}{f(\{u\})}$ [21] and measures how close f is to being modular. The minimum value $\kappa_f = 0$ implies that f is modular and the maximum value $\kappa_f = 1$ implies that f is fully curved. The (submodular) curvature of f with respect to a set $X \subseteq U$ is defined as $\hat{\kappa_f}(X) = 1 - \frac{\sum_{u \in X}(f(X) - f(X \setminus \{u\}))}{\sum_{u \in X} f(\{u\})}$ [22]. Note that the definition of $\hat{\kappa}_f(x)$ differs from that of κ_f in that: (I) it takes into account each element in the set X, instead of the element with the smallest ratio between the maximum and minimum marginal gain, and (II) it uses the ratio between the sum of the maximum marginal gain for each element in the set X and the sum of the minimum marginal gain for each element in X.

Let $g : 2^U \to \mathbb{R}_{\geq 0}$ be a non-negative supermodular function. The (supermodular) curvature of g is defined as $\kappa^g = 1 - \min_{u \in U} \frac{g(\{u\})}{g(U) - g(U \setminus \{u\})}$ [23]. Note, the definition of κ^g differs from that of κ_f in that the smallest marginal gain of an element u corresponds to when u is added into the smallest possible subset $\{\}$ of U (numerator in the min term of κ^g) instead of the largest possible subset $U \setminus \{u\}$ of U. Similarly, the largest marginal gain of u corresponds to when u is added into $U \setminus \{u\}$ (denominator in the min term of κ^g) instead of $\{\}$.

2.2. Independent cascade model

We model influence based on the classical independent cascade (IC) model [1,24,6]. The model views the social network as a weighted directed graph $G(V, E)$, where V and E are the sets of nodes and edges of G, respectively. In our setting, V is partitioned into $\mathcal{N}$ and $\mathcal{V}$, comprised of all *non-vulnerable* and *vulnerable* nodes, respectively. We assume that $\mathcal{N} \neq \varnothing$, otherwise no seed can be selected, and that $\mathcal{V}$ is determined by the organization performing influence maximization (e.g., the social network provider) based on domain knowledge regarding users profiles [11,8]. We also assume that seeds are selected from the set of non-vulnerable nodes (i.e., that the seed-set S is a subset of $\mathcal{N}$). This is natural, since selecting vulnerable nodes as seeds would harm them (i.e., it is against the goal of our approach). The set of in-neighbors (respectively, out-neighbors) of a node u is denoted by $n^-(u)$ (respectively, $n^+(u)$), and its size is referred to as the *in-degree* (respectively, *out-degree*) of u. In the IC model, each newly activated node u' tries to activate each inactive out-neighbor $u \in n^+(u')$ once with probability $p((u', u))$, which is modeled as the weight of edge (u', u) in E. The edge probability $p((u', u))$ is typically set to $\frac{1}{|n^-(u)|}$ [24]. If multiple newly activated nodes have the same inactive out-neighbor, they all try to activate it in an arbitrary order independently. The diffusion process starts from a set S of initial nodes (or *seeds*), which are active at time 0. Each seed tries to activate its out-neighbors at time 0, each activated out-neighbor stays active and tries to activate its own inactive out-neighbors at time 1, and the process proceeds similarly and ends when no new node becomes active. A seed-set S activates a node u with probability $P_S(u)$, and the *spread* of S over V, $\mathcal{N}$, and $\mathcal{V}$ is defined as $\sigma(S) = \sum_{u \in V} P_S(u)$, $\sigma_{\mathcal{N}}(S) = \sum_{u \in \mathcal{N}} P_S(u)$, and $\sigma_{\mathcal{V}}(S) = \sum_{u \in \mathcal{V}} P_S(u)$, respectively. For any seed-set S, $\sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S)$ are monotone submodular functions [1]. We may omit the argument and value of $\sigma_{\mathcal{N}}$ and $\sigma_{\mathcal{V}}$ when it is clear from the context (e.g., write a seed-set with zero $\sigma_{\mathcal{N}}$ instead of a seed-set S with $\sigma_{\mathcal{N}}(S) = 0$).

There are several methods for computing spread in the IC model. These include Monte Carlo simulation [1], dynamic programming algorithms [25,26], heuristics such as the MIA (Maximum Influence Arborescence) method [24], as well as sampling-based algorithms [27–29]. Dynamic programming algorithms are exact and generally faster than Monte Carlo simulation, since the latter requires a very large number of simulations to estimate spread well [1]. Sampling-based algorithms are approximate, unlike heuristics. Both sampling-based algorithms and heuristics are generally faster than dynamic programming algorithms. Our algorithms can use any method to compute spread. However, we employed the dynamic programming algorithm of [25], because, being exact, it allows to us to more clearly compare the effectiveness of the different algorithms for dealing with our problem.

2.3. Algorithms for optimizing submodular and non-submodular functions

In this section, we briefly review algorithms for optimizing submodular and non-submodular functions that are relevant to our work.

Greedy [14]. Given a non-negative monotone submodular function $f : 2^U \to \mathbb{R}_{\geq 0}$, and a parameter k, *Greedy* finds a subset $S \subseteq U$ of size $|S| \leq k$ with $f(S) \geq (1 - \frac{1}{\mathrm{e}}) \cdot \arg\max_{S' \subseteq U, |S'| \leq k} f(S')$, where e is the base of the natural logarithm. *Greedy* performs k iterations. In each iteration, it adds into S the element $u \in U$ with the maximum marginal gain $f(S \cup \{u\}) - f(S)$. Thus, it performs $O(|U| \cdot k)$ evaluations of function f assuming value oracle access to f (i.e., assuming that f can be queried at any subset of U in polynomial time). A variation of *Greedy*, referred to as *Lazy Greedy* or *CELF* [30,1], performs fewer evaluations of f in practice and is faster by orders of magnitude. CELF uses a priority queue to efficiently find the element of U that has the largest marginal gain and should be added into S. The priority queue is initialized with the marginal gain $f(S \cup \{u\}) - f(S)$ of each element $u \in U$ and is sorted in decreasing order. In the first iteration, the top entry is removed from the queue and its corresponding element u_1 is added into S. In the second iteration, the top entry of the queue (i.e., the second topmost entry in the previous iteration) is updated to reflect the addition of u_1 into S. If the entry stays on the top of the queue (i.e., it still has the largest marginal gain), it is removed from the queue and its corresponding element u_2 is added into S. This is because, after the update, the marginal gain of u_2 is at least equal to that of any element in $U \setminus S$, and due to submodularity, the marginal gain of any such element cannot increase. Otherwise, CELF updates the current top entry of the priority queue and repeats the process. After an element is added into S, CELF proceeds into the next iteration, which is similar to the second iteration. In practice, the number of updates on the priority queue is small, which makes the algorithm efficient.

GreedRatio [16,22]. Let $f : 2^U \to \mathbb{R}_{\geq 0}$ and $g : 2^U \to \mathbb{R}_{\geq 0}$ be non-negative submodular functions such that $f(\{\}) \geq 0$, $g(\{\}) \geq 0$, $f(\{u\}) > 0$ for each $u \in U$ and $g(\{u\}) > 0$ for each $u \in U$. GreedRatio works by iteratively adding into S the element u with the maximum ratio of marginal gains $\frac{g(S \cup \{u\}) - g(S)}{f(S \cup \{u\}) - f(S)}$, as long as there is at least one element with $f(S \cup \{u\}) - f(S) > 0$. Let λ be the number of iterations performed by Greedratio and S_i be the subset of S comprised of the elements that were added into S in the first $i \in [1, \lambda]$ iterations. Then, GreedRatio returns the subset S_i that has maximum ratio $\frac{g(S_i)}{f(S_i)}$ over $S_1, \ldots, S_\lambda$. Let $S^* = \arg\max_{S' \subseteq U} \frac{g(S')}{f(S')}$ be the subset of U with the maximum ratio and $S^{*,min}$ be the set S^* with the minimum size. *GreedRatio* finds a subset $S \subseteq U$ with $\frac{g(S)}{f(S)} \geq (1 - \mathrm{e}^{(\kappa_f - 1)}) \cdot \frac{g(S^*)}{f(S^*)}$, where κ_f is the curvature of the submodular function f. Recently, the bound for GreedRatio was improved [22] to $\frac{g(S)}{f(S)} \geq \frac{1 + (|S^{*,min}| - 1)(1 - \hat{\kappa_f}(S^{*,min})}{|S^{*,,min}|} \cdot \frac{g(S^{*,min})}{f(S^{*,min})}$, where $\hat{\kappa_f}(X)$ is the curvature with respect to a set $X \subseteq U$. Note that the bound in [16] for a modular function f (i.e., a function f with $\kappa_f = 0$) is $\frac{\mathrm{e}}{\mathrm{e}-1}$, while the improved bound in [22] is 1. In addition, the bound in [16] for a *fully curved* function f (i.e., a function f with $\kappa_f = 1$) is ∞, while the improved bound is $\frac{1}{|S^{*,min}|}$.

GreedMax [23]. Given a non-negative monotone submodular function $f : 2^U \to \mathbb{R}_{\geq 0}$, a non-negative monotone supermodular function $g : 2^U \to \mathbb{R}_{\geq 0}$, and a parameter k, *GreedMax* finds a subset $S \subseteq U$ of size $|S|$ at most k such that $f(S) + g(S) \geq (\frac{1}{\kappa_f} \cdot [1 - e^{-(1-\kappa^g) \cdot \kappa_f}]) \cdot \underset{S' \subseteq U, |S'| \leq k}{\arg\max} (f(S') + g(S'))$, where κ_f is the curvature of the submodular function f and κ^g is the curvature of the supermodular function g. *GreedMax* performs k iterations. In each iteration, it adds into S the element u with the maximum sum of marginal gains $f(S \cup \{u\}) - f(S) + g(S \cup \{u\}) - g(S)$.

Subsample Greedy [31]. Given a non-negative submodular function $f : 2^U \to \mathbb{R}_{\geq 0}$, and a parameter k, *Subsample Greedy* finds a subset $S \subseteq U$ of size $|S| \leq k$ with $\mathbb{E}[f(S)] \geq \frac{1}{\mathrm{e}} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot \arg\max_{S' \subseteq U : |S'| \leq k} f(S')$, where $\mathbb{E}[f(S)]$ denotes the expected value of $f(S)$. If f is additionally monotone, the guarantee improves to $\mathbb{E}[f(S)] \geq (1 - \frac{1}{\mathrm{e}}) \cdot \arg\max_{S' \subseteq U : |S'| \leq k} f(S')$. The expected value is computed over every possible S constructed by *Subsample Greedy*. *Subsample Greedy* performs k iterations. In each iteration, it constructs a uniform random sample of U, adds into the sample a dummy element e (i.e., an element with marginal gain $f(X \cup \{e\}) - f(X) = 0$, for each $X \subseteq U$), and adds into the subset S the element with the maximum marginal gain in the sample. The sample has size $\frac{|U|}{k}$ which must be an integer. If it is not an integer, the minimum number of dummy elements are added into U. After k iterations, any dummy elements are removed from the subset S, and the subset is returned. *Subsample Greedy* performs $O(|U|)$ evaluations of f, assuming value oracle access to f. Thus, it is more efficient than competitors [7] which perform $O(|U| \cdot k)$ evaluations.

Sandwich Approximation (*SA*) strategy [32]. The *SA* strategy approximates the following problem: Given a non-negative non-submodular function $f : 2^U \to \mathbb{R}_{\geq 0}$, non-negative monotone submodular functions $l_f : 2^U \to \mathbb{R}_{\geq 0}$ and $u_f : 2^U \to \mathbb{R}_{\geq 0}$ such that $l_f(S) \leq f(S) \leq u_f(S)$ for each subset $S \subseteq U$, and a parameter k, find a subset S of size $|S| \leq k$ with maximum $f(S)$. The *SA* strategy applies *Greedy* three times: with f to produce a subset S_f; with l_f to produce a subset S_{l_f}, and with u_f to produce a subset S_{u_f}. Then, *SA* returns the subset in $\{S_f, S_{l_f}, S_{u_f}\}$ with the largest value in f. Interestingly, S satisfies $f(S) \geq \max\left\{\frac{f(S_{u_f})}{u_f(S_{u_f})}, \frac{l_f(S^*)}{f(S^*)}\right\} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot f(S^*)$, where $S^* = \arg\max_{S \subseteq U, |S| \leq k} f(S)$ is the optimal subset of size at least

k with respect to f. That is, SA allows approximating a non-submodular function f based on *Greedy* to obtain a solution that is worse than the solution that would be obtained by *Greedy* if f was monotone submodular by no more than a factor $\max\{\frac{f(S_{u_f})}{u_f(S_{u_f})}, \frac{l_f(S^*)}{f(S^*)}\}$. The factor cannot be computed in polynomial time because it requires computing S^*. Yet, for some functions f such as those in [32], it can be approximated fairly well by $\frac{f(S_{u_f})}{u_f(S_{u_f})}$, and therefore, the following slightly weaker lower bound applies $f(S) \geq \frac{f(S_{u_f})}{u_f(S_{u_f})} \cdot (1 - \frac{1}{e}) \cdot f(S^*)$.

3. Measures and problem definition

To study influence maximization in our setting, we need a measure that quantifies the quality of a seed-set and can be incorporated into methods to construct a high quality seed-set. The measure should favor a seed-set S that influences many non-vulnerable but few vulnerable nodes and also satisfy the following properties:

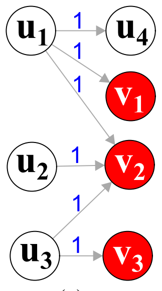
1. It should consider the influence of vulnerable and non-vulnerable nodes. In fact, we observed experimentally that constructing S based on only $\sigma_{\mathcal{N}}(S)$ (resp., $\sigma_{\mathcal{V}}(S)$) results in large $\sigma_{\mathcal{V}}(S)$ (resp., small $\sigma_{\mathcal{N}}(S)$), which is undesirable.
2. It should consider what fraction of all influenced users are vulnerable. This is important to penalize seed-sets that influence a large expected number of users many of whom are vulnerable.
3. It should allow constructing a seed-set with guaranteed quality (e.g., not "too far" from the optimal seed-set in the worst case) [1].

Natural measures. A first measure is the difference $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$ given a seed-set S (i.e., the measure used in [6], with vulnerable nodes being treated as non-target nodes). This measure does not consider what fraction of all influenced users are vulnerable. Therefore, it may lead to constructing seed-sets with a large expected number of influenced users many of whom are vulnerable. For example, this measure would favor promoting an alcoholic beverage to 140 users out of whom 40 have drinking problems, instead of 59 users with no drinking problems, since $(140 - 40) - 40 > 59 - 0$.

In addition, $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$ is a non-submodular function [6], expressed as a difference between two submodular functions, so it is difficult to approximately maximize [20]. Thus, to construct a seed-set S, one has to settle with heuristics, such as [6], which offer no approximation guarantees. One may notice that $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$ can be written as a sum of the submodular function $\sigma_{\mathcal{N}}(S)$ and the supermodular function $-\sigma_{\mathcal{V}}(S)$ ($-\sigma_{\mathcal{V}}(S)$ is supermodular since $\sigma_{\mathcal{V}}(S)$ is submodular); and recall from Section 2 that the *GreedMax* algorithm [23] can be applied to a sum of a submodular and a supermodular function. However, we cannot use the *GreedMax* algorithm to find a seed-set of size at most k with approximately maximum $f(S) + g(S) = \sigma_{\mathcal{N}}(S) + (-\sigma_{\mathcal{V}}(S))$. This is because *GreedMax* also requires $g(S)$ to be (I) non-negative and (II) monotone, whereas $g(S) = -\sigma_{\mathcal{V}}(S)$ can clearly not take positive values and it is also not monotone (since $\sigma_{\mathcal{V}}(S)$ is monotone [1]).

Another natural measure is the ratio $\frac{\sigma_{\mathcal{N}}(S)}{\sigma_{\mathcal{V}}(S)}$. The ratio considers what fraction of all influenced users are vulnerable, because it can be rewritten as $\frac{\sigma(S)-\sigma_{\mathcal{V}}(S)}{\sigma_{\mathcal{V}}(S)} = \frac{\sigma(S)}{\sigma_{\mathcal{V}}(S)} - 1$ and the constant can be removed when it is maximized. However, the ratio is undefined for every seed-set S with $\sigma_{\mathcal{V}}(S) = 0$ (i.e., S that does not influence vulnerable nodes). Thus, it cannot distinguish between any two seed-sets S_1, S_2 such that $\sigma_{\mathcal{V}}(S_1) = \sigma_{\mathcal{V}}(S_2) = 0$ and $\sigma_{\mathcal{N}}(S_1) > \sigma_{\mathcal{N}}(S_2)$ (e.g., it cannot favor promoting an alcoholic beverage to 59 users with no drinking problems vs. 2 users with no drinking problems) and also it is not clear how it can be approximately maximized. For example, the *GreedRatio* framework [16,22] (see Section 2) for maximizing a ratio of two monotone submodular functions $f(S) = \sigma_{\mathcal{N}}(S)$ and $g(S) = \sigma_{\mathcal{V}}(S)$ would output a seed-set S of unbounded size, which is not useful for influence maximization. This is because there is a limited budget that an organization can devote to seeds (i.e., free sample products or discounts that are given to users in order to attract them to start the diffusion process). The inverse ratio $\frac{\sigma_{\mathcal{V}}(S)}{\sigma_{\mathcal{N}}(S)}$ is defined for $\sigma_{\mathcal{V}}(S) = 0$ but it cannot be used to distinguish between the seed-sets S_1 and S_2 above, and it is equally difficult to minimize (minimizing it is equivalent to maximizing $\frac{\sigma_{\mathcal{N}}(S)}{\sigma_{\mathcal{V}}(S)}$). Thus, it cannot be used to find a seed-set with small or zero $\sigma_{\mathcal{V}}(S)$ and large $\sigma_{\mathcal{N}}(S)$, which helps our goal (to attract many users few of whom are vulnerable).

Our proposed measure. To retain the benefits of the ratio $\frac{\sigma_{\mathcal{N}}(S)}{\sigma_{\mathcal{V}}(S)}$, while fixing the issues caused by seed-sets that do not influence any vulnerable nodes, we apply additive smoothing [33] to the ratio. This leads to our *additively smoothed ratio* (*ASR*) measure, defined as $ASR(S, c) = \frac{\sigma_{\mathcal{N}}(S)+c}{\sigma_{\mathcal{V}}(S)+c}$, where S is a seed-set and $c > 0$ is a constant determined by the organization performing influence maximization. *ASR* is well defined (and larger than zero) when $\sigma_{\mathcal{V}}(S) = 0$. Furthermore, among the seed-sets S_1 and S_2 mentioned above, it favors the seed-set S_1, which influences a larger expected number of non-vulnerable nodes. In *ASR*, the constant c can be seen as a weight whose addition to $\sigma_{\mathcal{N}}(S)$ and to $\sigma_{\mathcal{V}}(S)$ changes their ratio and determines seed selection. The impact of c on seed selection will be discussed in Sections 4 and 8. Given the *ASR* measure, we are in a position to define our influence maximization problem.



(a)

Seed-set S	$\sigma_{\mathcal{N}}(S)$	$\sigma_{\mathcal{V}}(S)$	$ASR(S,1)$
$S_0 = \{u_2, u_4\}$	2	1	$3/2$
$S_0 \cup \{u_1\}$	3	2	$4/3$
$S_1 = \{u_2, u_3, u_4\}$	3	2	$4/3$
$S_1 \cup \{u_1\}$	4	3	$5/4$
$S_2 = \{u_3\}$	1	2	$2/3$
$S_2 \cup \{u_2\}$	2	2	1
$S_3 = \{u_1, u_3\}$	3	3	1
$S_3 \cup \{u_2\}$	4	3	$5/4$

(b)

Fig. 2. (a) Example graph. $\mathcal{N} = \{u_1, \ldots, u_4\}$, $\mathcal{V} = \{v_1, v_2, v_3\}$, and each edge probability is equal to 1. (b) The spread over non-vulnerable nodes, the spread over vulnerable nodes, and *ASR* for different seeds-sets with $c = 1$.

Problem 1 *(*ASR*-Maximization (*ASR*-MAX)). Given a graph G whose nodes are partitioned into $\mathcal{N}$ and $\mathcal{V}$ and parameters k and c, find a seed-set $S \subseteq \mathcal{N}$ of size at most k that maximizes* ASR(S, c).

The next theorem establishes the hardness of the *ASR*-MAX problem.

Theorem 1. *The* ASR*-MAX problem is NP-hard.*

Proof. The Influence Maximization (IM) problem is NP-hard under the Independent Cascade model [1]. We show that *ASR*-MAX is NP-hard by restriction. Specifically, from any given instance $\mathcal{I}_{\text{IM}}$ of the IM problem, we create an instance $\mathcal{I}_{ASR-\text{MAX}}$ of the *ASR*-MAX problem in polynomial time. The only difference between the instances is that the graph, in the instance of the *ASR*-MAX problem, has only non-vulnerable nodes. Clearly, a solution of $\mathcal{I}_{ASR-\text{MAX}}$ has at most k nodes and a maximum *ASR*, or equivalently a maximum spread, among seed-sets of size at most k (since all nodes are non-vulnerable). Thus, the corresponding seed-set of $\mathcal{I}_{\text{IM}}$ has also at most k nodes and a maximum spread among seed-sets of size at most k, and therefore it is a solution of $\mathcal{I}_{\text{IM}}$. The converse can be shown similarly (omitted). $\square$

The *ASR*-MAX problem is fundamentally different from the IM problem. This is because, as we show below, the *ASR*-MAX problem seeks to optimize a non-monotone non-submodular function, unlike IM that seeks to optimize a monotone submodular function.

Theorem 2. ASR *is non-monotone and is neither submodular nor supermodular.*

Proof. We prove the theorem by means of a counterexample. Consider the graph of Fig. 2a, whose set of nodes is partitioned into $\mathcal{N} = \{u_1, \ldots, u_4\}$ and $\mathcal{V} = \{v_1, v_2, v_3\}$, and the *ASR* of the seed-sets in Fig. 2b with $c = 1$. $ASR(S, c)$ is: (I) *non-monotone*, because for $S_0 \subseteq S_0 \cup \{u_1\}$, $ASR(S_0, 1) = 3/2 > ASR(S_0 \cup \{u_1\}, 1) = 4/3$; (II) *not* submodular, because for $S_0 \subseteq S_1$ and $u_1 \in \mathcal{N} \setminus S_1$,
$ASR(S_0 \cup \{u_1\}, 1) - ASR(S_0, 1) = -1/6 < ASR(S_1 \cup \{u_1\}, 1) - ASR(S_1, 1) = -1/12$, and (III) *not* supermodular, because for $S_2 \subseteq S_3$ and $u_2 \in \mathcal{N} \setminus S_3$, $ASR(S_2 \cup \{u_2\}, 1) - ASR(S_2, 1) = 1/3 > ASR(S_3 \cup \{u_2\}, 1) - ASR(S_3, 1) = 1/4$. $\square$

One may wonder whether the GreedRatio [16,22] framework can be used to solve the *ASR*-MAX problem, since *ASR* is a ratio between two monotone submodular functions. This is not possible, because the algorithms in [16,22] construct an unbounded seed-set, which is not useful in our setting, and also because the analysis in [22] assumes that $\sigma_{\mathcal{V}}(\{\}, c) = 0$, which does not hold in our case.

4. Baselines: greedy heuristics for maximizing *ASR*

We explore two greedy baseline methods for constructing a seed-set S with size at most k and large $ASR(S, c)$. The first is *GR*, a natural heuristic for limiting the influence to vulnerable nodes. *GR* is conceptually similar to GreedRatio in that it aims to maximize the ratio of two submodular functions iteratively. However, it differs from *GR* in two important dimensions. First, it creates seed-sets of size at most k, instead of seed-sets of unbounded size. Second, it uses $ASR(S, c)$ instead of $\frac{\sigma_{\mathcal{N}}(S)}{\sigma_{\mathcal{V}}(S)}$ (i.e., it adds c into the numerator and denominator of the latter ratio to avoid the zero-spread problem in the denominator). Intuitively, *GR* aims to add a node which many non-vulnerable and few vulnerable nodes to be influenced.

Table 1
(a) Non-vulnerable nodes u that are considered for addition into $S_0 = \{\}$, and the expected number of vulnerable and non-vulnerable nodes they influence. (b) The node that is added into S_0 for different values of c.

Node	u_1	u_2	u_3	u_4
$\sigma_{\mathcal{V}}$	0	0.01	1	10
$\sigma_{\mathcal{N}}$	3	5	150	300

(a)

c	0.01	0.02	1	10
Added Node	u_1	u_2	u_3	u_4

(b)

```
Algorithm: GR (GReedy heuristic)
Input: $\mathcal{N} \subseteq V$, $\mathcal{V} \subseteq V$, graph $G$, parameter $k$, constant $c$
Output: Subset $S \subseteq \mathcal{N}$ of size $|S| \leq k$
1 $i \leftarrow 0$   // Iteration counter
2 $S_i \leftarrow \{\}$
3 while $i < k$ do
4     $u \in \arg\max_{v \in \mathcal{N} \setminus \{S_i\}} \frac{\sigma_{\mathcal{N}}(S_i|v) + c}{\sigma_{\mathcal{V}}(S_i|v) + c}$
5     $S_{i+1} \leftarrow S_i \cup \{u\}$
6     $i \leftarrow i + 1$
7 return $S \leftarrow \arg\max_{S' \in \{S_1,\dots,S_k\}} ASR(S', c)$
```

GR performs k iterations. In each iteration i (steps 3 to 6), it adds into the subset S_i the node u with the maximum ratio between: (I) the sum of the marginal gain in $\sigma_{\mathcal{N}}$, caused by adding u, and the constant c, and (II) the sum of the marginal gain in $\sigma_{\mathcal{V}}$, caused by adding u, and the constant c. Since *ASR* is non-monotone, a subset constructed in an iteration before i may have a larger *ASR* than S_i. Therefore, in Step 7, *GR* considers the subsets constructed in all iterations and returns the one with the largest *ASR*.

We now discuss how *GR* deals with a non-vulnerable node v that influences no vulnerable nodes. Adding v into S_i makes the objective function of *GR* equal to $\frac{\sigma_{\mathcal{N}}(S_i|v)+c}{c}$ (see Step 4), since S_i does not influence more vulnerable nodes after the addition of v (i.e., $\sigma_{\mathcal{V}}(S_i|v) = 0$). If $\sigma_{\mathcal{N}}(S_i|v)$ is small, it is better to add a different node v' which influences few vulnerable nodes but "through" these vulnerable nodes reaches out to many more non-vulnerable nodes than v. In fact, *GR* adds v' instead of v if $\frac{\sigma_{\mathcal{N}}(S_i|v')+c}{\sigma_{\mathcal{V}}(S_i|v')+c} > \frac{\sigma_{\mathcal{N}}(S_i|v)+c}{c}$, and uses the parameter c to control the bias towards nodes such as v'. Such a node v' influences a small number of vulnerable nodes but many more non-vulnerable nodes than u, as shown in Example 1 and experimentally in Section 8.

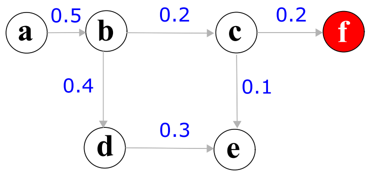
Example 1. In iteration $i = 0$, the non-vulnerable nodes u_1 to u_4 in Table 1a are considered and the node $u \in \{u_1, \dots, u_4\}$ with the largest $\frac{\sigma_{\mathcal{N}}(S_0|u)+c}{\sigma_{\mathcal{V}}(S_0|u)+c}$ is added into $S_0 = \{\}$. As shown in Table 1b, the node c determines the added node. For $c = 0.01$, u_1 that influences no vulnerable and few non-vulnerable nodes is added, for $c = 1$, u_3 that influences one vulnerable and many non-vulnerable nodes is added, and for $c = 10$, u_4 that influences more vulnerable and non-vulnerable nodes than u_2 is added. □

To improve the efficiency of *GR*, we propose a variant, GR_{MB} (MB stands for Maximum influence arborescence Batch-update). Unlike *GR* which computes spread exactly by adapting the method of [25] to the IC model, GR_{MB} estimates the spread efficiently using the MIA method [24] (see Section 2). MIA estimates the probability $P_S(u)$ for a node u and seed-set S based on the union of paths from S that have the largest probability to influence u, instead of all paths, and this may lead to a different value of spread (see Example 2). Since GR_{MB} computes the activation probability based on generally fewer paths, it is faster than *GR*. Specifically, our experiments in Section 8 show that GR_{MB} is more efficient than *GR* by two orders of magnitude on average.

Example 2. Consider the graph in Fig. 3, where the set of non-vulnerable nodes is $\mathcal{N} = \{a, b, c, d, e\}$ and the set of vulnerable nodes is $\mathcal{V} = \{f\}$. When the seed-set is $S = \{a\}$, *GR* computes the probability that e is activated as $P_S(e) = 1 - [1 - p((a,b)) \cdot p((b,c)) \cdot p((c,e))] \cdot [1 - p((a,b)) \cdot p((b,d)) \cdot p((d,e))] = 0.0694$, where the expression in the first (respectively, second) pair of square brackets corresponds to the probability that a does not activate e through the path $\{(a,b),(b,c),(c,e)\}$ (respectively, $\{(a,b),(b,d),(d,e)\}$). That is, *GR* computes $P_S(e)$ based on both paths from a to e. On the other hand, GR_{MB} computes the probability $P_S(e)$ using MIA as $P_S(e) = [p((a,b)) \cdot p((b,d)) \cdot p((d,e))] = 0.06$, which corresponds to the probability that a activates e through the path $\{(a,b),(b,d),(d,e)\}$. This is because the path $\{(a,b),(b,d),(d,e)\}$ has a larger probability to activate e compared to that of the path $\{(a,b),(b,c),(c,e)\}$. Since the activation probabilities are different, *GR* and GR_{MB} compute a different value of spread $\sigma_{\mathcal{N}}$, as shown in Fig. 3b. The value of spread $\sigma_{\mathcal{V}}$ in Fig. 3b is the same, because there is a single path from a to the vulnerable node f.

5. The *SAS* algorithm for maximizing *ASR*

This section presents *SAS* (Sandwich Approximation algorithm with Spread bounds), starting from the bound functions of *ASR* that the algorithm employs.



(a)

	GR	GR_{MB}
$\sigma_{\mathcal{N}}(S)$	0.87	0.86
$\sigma_{\mathcal{V}}(S)$	0.02	0.02

(b)

Fig. 3. (a) Example graph. $\mathcal{N} = \{a, b, c, d, e\}$, $\mathcal{V} = \{f\}$, and each edge probability is shown as an edge label. (b) The spread $\sigma_{\mathcal{N}}$ over $\mathcal{N} = \{a, b, c, d, e\}$ and the spread $\sigma_{\mathcal{V}}$ over $\mathcal{V} = \{e\}$ for GR and GR_{MB}, when the seed-set $S = \{a\}$ in the graph of Fig. 3a.

Lower and Upper Bound Function of *ASR*. $ASR(S, c)$ is non-monotone non-submodular for any subset S (see Section 3) and, thus, it is difficult to approximate directly. Our *SAS* algorithm finds a seed-set S with approximately maximum $ASR(S, c)$, using two submodular functions $ASR^{\mathbf{L}}$ and $ASR^{\mathbf{U}}$ that bound *ASR* from below and from above, respectively. These functions are defined as follows:

$$ASR^{\mathbf{L}}(S, c) = \frac{\sigma_{\mathcal{N}}(S) + c}{|\mathcal{V}| + c}$$
$$ASR^{\mathbf{U}}(S, c) = \frac{\sigma_{\mathcal{N}}(S) + c}{c}.$$

$ASR^{\mathbf{L}}$ is obtained by replacing $\sigma_{\mathcal{V}}(S)$ in *ASR* with $|\mathcal{V}|$. Since the expected number of influenced vulnerable nodes is at most $|\mathcal{V}|$, the following holds for any $S \subseteq \mathcal{N}$:

$$\sigma_{\mathcal{V}}(S) \leq |\mathcal{V}|. \tag{4}$$

Thus, $ASR^{\mathbf{L}}$ is a lower bound of *ASR*.

$ASR^{\mathbf{U}}$ is obtained by removing $\sigma_{\mathcal{V}}(S)$ from the denominator of *ASR* (i.e., replacing $\sigma_{\mathcal{V}}(S)$ with 0). Since $\sigma_{\mathcal{V}}(S) > 0$ be definition, for each $S \subseteq \mathcal{N}$, $ASR^{\mathbf{U}}$ is an upper bound of *ASR*.

The reader can easily verify that both $ASR^{\mathbf{L}}$ and $ASR^{\mathbf{U}}$ are non-negative. We now show that $ASR^{\mathbf{L}}$ and $ASR^{\mathbf{U}}$ are monotone submodular. These properties are important for designing our *SAS* algorithm, based on the SA strategy.

Lemma 1. $\mathrm{ASR}^{\mathbf{L}}$, *as well as* $\mathrm{ASR}^{\mathbf{U}}$, *is monotone submodular with respect to a seed-set S for any given k and* $c > 0$.

Proof. We first consider $ASR^{\mathbf{L}}$. The monotonicity of $ASR^{\mathbf{L}}$ follows directly from the monotonicity of $\sigma_{\mathcal{N}}$ (i.e., $\sigma_{\mathcal{N}}(S) \leq \sigma_{\mathcal{N}}(S')$, for all subsets $S \subseteq S' \subseteq \mathcal{N}$) and from the fact that $|\mathcal{V}| + c$ is a fixed positive number.

We now show the submodularity of $ASR^{\mathbf{L}}$. Since $\sigma_{\mathcal{N}}$ is submodular [1], the following inequality holds for any $S \subseteq S' \subseteq \mathcal{N}$ and $u \in \mathcal{N} \setminus S'$:

$$\sigma_{\mathcal{N}}(S \cup \{u\}) - \sigma_{\mathcal{N}}(S) \geq \sigma_{\mathcal{N}}(S' \cup \{u\}) - \sigma_{\mathcal{N}}(S'). \tag{5}$$

By adding and subtracting c to each part of Equation (5) and then dividing it by $|\mathcal{V}| + c > 0$, we obtain the following inequalities, which hold for any $S \subseteq S' \subseteq \mathcal{N}$ and $u \in \mathcal{N} \setminus S'$

$$\frac{(\sigma_{\mathcal{N}}(S \cup \{u\}) + c) - (\sigma_{\mathcal{N}}(S) + c)}{|\mathcal{V}| + c} \geq \frac{(\sigma_{\mathcal{N}}(S' \cup \{u\}) + c) - (\sigma_{\mathcal{N}}(S') + c)}{|\mathcal{V}| + c}$$
$$\frac{\sigma_{\mathcal{N}}(S \cup \{u\}) + c}{|\mathcal{V}| + c} - \frac{\sigma_{\mathcal{N}}(S) + c}{|\mathcal{V}| + c} \geq \frac{\sigma_{\mathcal{N}}(S' \cup \{u\}) + c}{|\mathcal{V}| + c} - \frac{\sigma_{\mathcal{N}}(S') + c}{|\mathcal{V}| + c}$$

The latter inequality gives:

$$ASR^{\mathbf{L}}(S \cup \{u\}, c) - ASR^{\mathbf{L}}(S, c) \geq ASR^{\mathbf{L}}(S' \cup \{u\}, c) - ASR^{\mathbf{L}}(S', c),$$

which implies that $ASR^{\mathbf{L}}$ is submodular.

We now consider $ASR^{\mathbf{U}}$. The monotonicity of $ASR^{\mathbf{U}}$ follows directly from the monotonicity of $\sigma_{\mathcal{N}}$ and the fact that c is a fixed positive number.

By adding and subtracting c to each part of Equation (5) and then diving it by c, we obtain the following inequalities:

$$\frac{(\sigma_{\mathcal{N}}(S \cup \{u\}) + c) - (\sigma_{\mathcal{N}}(S) + c)}{c} \geq \frac{(\sigma_{\mathcal{N}}(S' \cup \{u\}) + c) - (\sigma_{\mathcal{N}}(S') + c)}{c}$$
$$\frac{\sigma_{\mathcal{N}}(S \cup \{u\}) + c}{c} - \frac{\sigma_{\mathcal{N}}(S) + c}{c} \geq \frac{\sigma_{\mathcal{N}}(S' \cup \{u\}) + c}{c} - \frac{\sigma_{\mathcal{N}}(S') + c}{c}.$$

The latter inequality holds for any $S \subseteq S' \subseteq \mathcal{N}$ and $u \in \mathcal{N} \setminus S'$ and implies that $ASR^{\mathbf{U}}$ is submodular. □

***SAS* algorithm.** As can be seen from the pseudocode, the algorithm works in three phases: (I) dummy element creation; (II) construction of three candidates seed-sets (one using *ASR*, a second using $ASR^{\mathbf{L}}$ and a third using $ASR^{\mathbf{U}}$); and (III) selection of the best candidate seed-set and removal of dummy elements from it.

```
Algorithm: SAS (Sandwich Approximation algorithm with Spread bounds)
Input: Set of non-vulnerable nodes $\mathcal{N} \subseteq V$, set of vulnerable nodes $\mathcal{V} \subseteq V$, graph $G$, parameter $k$, constant $c$
Output: Subset $S \subseteq \mathcal{N}$ of size $|S| \leq k$
1  $S_{cur} \leftarrow \mathcal{N}$
   // Phase I
2  $\mathcal{D} \leftarrow$ set of $k$ dummy elements $\{u_1, \dots, u_k\}$ such that, for each element $u_i$, $i \in [1,k]$, and every set $S \subseteq \mathcal{N}$: $\sigma_{\mathcal{N}}(S \cup \{u_i\}) = \sigma_{\mathcal{N}}(S)$ and
     $\sigma_{\mathcal{V}}(S \cup \{u_i\}) = \sigma_{\mathcal{V}}(S)$
3  $\mathcal{N}' \leftarrow \mathcal{N}$
4  while $\frac{|\mathcal{N}'|}{k}$ is not an integer do
5      Add into $\mathcal{N}'$ a dummy element $u' \notin \mathcal{D}$ such that $\sigma_{\mathcal{N}}(S \cup \{u'\}) = \sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S \cup \{u'\}) = \sigma_{\mathcal{V}}(S)$
   // Phase II
6  $i \leftarrow 0$; $S^{\mathbf{O}} \leftarrow \{\}$; $S^{\mathbf{L}} \leftarrow \{\}$; $S^{\mathbf{U}} \leftarrow \{\}$
7  while $i < k$ do
8      $\mathcal{R} \leftarrow$ uniform random sample of $\mathcal{N}'$ with $\frac{|\mathcal{N}'|}{k}$ elements
9      Add into $\mathcal{R}$ a random element from $\mathcal{D}$
10     $u^{\mathbf{O}} \in \arg\max_{u \in \mathcal{R}} (ASR(S^{\mathbf{O}} \cup \{u\}, c) - ASR(S^{\mathbf{O}}, c))$
11     $S^{\mathbf{O}} \leftarrow S^{\mathbf{O}} \cup \{u^{\mathbf{O}}\}$
12     $u^{\mathbf{L}} \in \arg\max_{u \in \mathcal{R}} (ASR^{\mathbf{L}}(S^{\mathbf{L}} \cup \{u\}, c) - ASR^{\mathbf{L}}(S^{\mathbf{L}}, c))$
13     $S^{\mathbf{L}} \leftarrow S^{\mathbf{L}} \cup \{u^{\mathbf{L}}\}$
14     $u^{\mathbf{U}} \in \arg\max_{u \in \mathcal{R}} (ASR^{\mathbf{U}}(S^{\mathbf{U}} \cup \{u\}, c) - ASR^{\mathbf{U}}(S^{\mathbf{U}}, c))$
15     $S^{\mathbf{U}} \leftarrow S^{\mathbf{U}} \cup \{u^{\mathbf{U}}\}$
   // Phase III
16 $S_{cur} \leftarrow \arg\max_{S \in \{S^{\mathbf{O}}, S^{\mathbf{L}}, S^{\mathbf{U}}\}} ASR(S, c)$
17 $S_{cur} \leftarrow$ Remove all dummy elements from $S_{cur}$
18 return $S_{cur}$
```

Phase I (Steps 2 to 5): A set $\mathcal{D}$ of k *dummy* elements, whose addition into any seed-set S does not change $\sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S)$, are created. Then, a dummy element $u' \notin \mathcal{D}$ is added into a subset $\mathcal{N}'$ (initially containing all non-vulnerable nodes), until $\frac{|\mathcal{N}'|}{k}$ is an integer.

Phase II (Steps 6 to 15): A random sample of $\frac{|\mathcal{N}'|}{k}$ elements from $\mathcal{N}'$ and a dummy element is created. Next, a node in the sample causing the largest marginal gain with respect to *ASR*, $ASR^{\mathbf{L}}$, and $ASR^{\mathbf{U}}$ is added into the candidate subset $S^{\mathbf{O}}$, $S^{\mathbf{U}}$, and $S^{\mathbf{L}}$, respectively.

Phase III (Steps 16 to 18): The best candidate subset with respect to *ASR* is selected as S_{cur}, and all dummy elements are removed from it. After that, S_{cur} is returned.

We now establish that the *SAS* algorithm offers the following approximation guarantee.

Theorem 3. SAS *constructs a seed-set S satisfying*

$$\mathbb{E}[\text{ASR}(S,c)] \geq \frac{\sigma_{\mathcal{V}}(S^*)+c}{|\mathcal{V}|+c} \cdot (1-\frac{1}{\mathrm{e}}) \cdot \text{ASR}(S^*,c) \tag{6}$$

where $S^* = \arg\max_{S \subseteq \mathcal{N}, |S| \leq k} \text{ASR}(S,c)$ *is an optimal seed-set of size at most k with respect to* ASR, e *is the base of the natural logarithm, and* $\mathbb{E}[\text{ASR}(S,c)]$ *denotes the expectation of* ASR *over every possible S constructed by* SAS.

Proof. Since $ASR^{\mathbf{L}}$ bounds *ASR* from below, we have $ASR(S^{\mathbf{L}},c) \geq ASR^{\mathbf{L}}(S^{\mathbf{L}},c)$, for any $S^{\mathbf{L}} \subseteq \mathcal{N}$. Also, from the monotonicity of expectation, this inequality can be written as

$$\mathbb{E}[ASR(S^{\mathbf{L}},c)] \geq \mathbb{E}[ASR^{\mathbf{L}}(S^{\mathbf{L}},c)], \tag{7}$$

where the expectation is over every $S^{\mathbf{L}}$. Let $S^{\mathbf{L},*} = \arg\max_{S \subseteq \mathcal{N}, |S| \leq k} ASR^{\mathbf{L}}(S,c)$ be an optimal seed-set of size at most k with respect to $ASR^{\mathbf{L}}$. We observe that:

$$\mathbb{E}[ASR^{\mathbf{L}}(S^{\mathbf{L}},c)] \geq (1-\frac{1}{\mathrm{e}}) \cdot ASR^{\mathbf{L}}(S^{\mathbf{L},*},c) \tag{8}$$

$$\geq (1-\frac{1}{\mathrm{e}}) \cdot ASR^{\mathbf{L}}(S^*,c) \tag{9}$$

$$\geq \frac{ASR^{\mathbf{L}}(S^*,c)}{ASR(S^*,c)} \cdot (1-\frac{1}{\mathrm{e}}) \cdot ASR(S^*,c) \tag{10}$$

$$\geq \frac{\sigma_{\mathcal{V}}(S^*, c) + c}{|\mathcal{V}| + c} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot ASR(S^*, c). \tag{11}$$

Equation (8) holds because $S^{\mathbf{L}}$ is constructed based on the *Subsample Greedy* algorithm [31] with $ASR^{\mathbf{L}}$, which is a monotone submodular function according to Lemma 1 (recall from Section 2 the guarantee of *Subsample Greedy* in the case of a monotone submodular function). Equation (9) holds from the definition of $S^{\mathbf{L},*}$ and of S^*. Equation (10) holds, because $ASR(S^*, c)$ is positive. Equation (11) holds from the definition of $ASR^{\mathbf{L}}(S^*, c)$ and $ASR(S^*, c)$.

Thus, from Equations (7) and (11), we obtain:

$$\mathbb{E}[ASR(S^{\mathbf{L}}, c)] \geq \frac{\sigma_{\mathcal{V}}(S^*, c) + c}{|\mathcal{V}| + c} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot ASR(S^*, c). \tag{12}$$

We now prove the following:

$$ASR(S^{\mathbf{U}}, c) = \frac{\sigma_{\mathcal{N}}(S^{\mathbf{U}}) + c}{\sigma_{\mathcal{V}}(S^{\mathbf{U}}) + c} \cdot \frac{c}{c} \tag{13}$$

$$= ASR^{\mathbf{U}}(S^{\mathbf{U}}, c) \cdot \frac{c}{\sigma_{\mathcal{V}}(S^{\mathbf{U}}) + c} \tag{14}$$

$$\geq ASR^{\mathbf{U}}(S^{\mathbf{U}}, c) \cdot \frac{c}{|\mathcal{V}| + c}. \tag{15}$$

Equations (13) and (14) follow from the definition of *ASR* and $ASR^{\mathbf{U}}$, respectively, and Equation (15) follows from $\sigma_{\mathcal{V}}(S^{\mathbf{U}}) \leq |\mathcal{V}|$.

In addition, we show the following:

$$\mathbb{E}[ASR(S^{\mathbf{U}}, c)] \geq \mathbb{E}\left[ASR^{\mathbf{U}}(S^{\mathbf{U}}, c) \cdot \frac{c}{|\mathcal{V}| + c}\right] \tag{16}$$

$$\geq \frac{c}{|\mathcal{V}| + c} \cdot \mathbb{E}\left[ASR^{\mathbf{U}}(S^{\mathbf{U}}, c)\right] \tag{17}$$

$$\geq \frac{c}{|\mathcal{V}| + c} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot ASR^{\mathbf{U}}(S^{\mathbf{U},*}, c) \tag{18}$$

$$\geq \frac{c}{|\mathcal{V}| + c} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot ASR^{\mathbf{U}}(S^*, c) \tag{19}$$

$$\geq \frac{c}{|\mathcal{V}| + c} \cdot (1 - \frac{1}{\mathrm{e}}) \cdot ASR(S^*, c) \tag{20}$$

where the expectation is over each $S^{\mathbf{U}}$ and $S^{\mathbf{U},*} = \arg\max_{u \in \mathcal{N}, |S| \leq k} ASR^{\mathbf{U}}(S, c)$ is an optimal seed-set of size at most k with respect to $ASR^{\mathbf{U}}$. Equation (16) follows from Equation (15) and the linearity of expectation. Equation (17) holds because the fraction in this equation is a constant which is independent of $S^{\mathbf{U}}$. Equation (18) holds because $S^{\mathbf{U}}$ is constructed based on the *Subsample Greedy* algorithm [31] with $ASR^{\mathbf{U}}$, which is a monotone submodular function according to Lemma 1. Equation (19) holds from the definitions of $S^{\mathbf{U},*}$ and of S^*, and Equation (20) holds because $ASR^{\mathbf{U}}$ upper-bounds *ASR*.

The statement follows from Equations (12) and (20), the fact that $\sigma_{\mathcal{V}}(S^*, c) + c \geq c$, and the fact that the output seed-set

$$S \in \operatorname*{arg\,max}_{S' \in \{S^{\mathbf{O}}, S^{\mathbf{L}}, S^{\mathbf{U}}\}} ASR(S', c)$$

(see Step 16 of *SAS*). □

We now show the number of the evaluations of the spread function performed by *SAS* in the proposition below.

Proposition 1. SAS *performs* $O(|\mathcal{N}|)$ *evaluations of the spread function.*

Proof. The statement follows directly from the fact that Subsample Greedy performs $O(|\mathcal{N}|)$ evaluations of the spread function [31] and from the fact that *SAS* executes this algorithm on three functions (*ASR* and the bound functions). □

We express the computational cost of *SAS* in terms of evaluations of the spread function, as the specific cost of evaluating the spread function heavily depends on the influence diffusion model and on the algorithm for computing spread. For example, the cost of evaluating the spread function (i.e., $\sigma_{\mathcal{N}}$ or $\sigma_{\mathcal{V}}$ using the dynamic programming algorithm of [25] is $O(|\mathcal{N}|^2|E|)$ [25]. However, this cost is pessimistic [25], as it assumes a complete graph. Thus, by Proposition 1, when spread is computed using the dynamic programming algorithm of [25], the cost of *SAS* is $O(|\mathcal{N}|^3|E|)$.

The strategy of selecting seeds from a sample of $\mathcal{N}$ of size approximately $\frac{\mathcal{N}}{k}$, instead of the entire set $\mathcal{N}$, is inspired from *Subsample Greedy* (see Section 2). The difference is that Subsample Greedy is applied to a single submodular function and therefore it uses one random sample per iteration. On the other hand, *SAS* is applied to two functions and uses a single random sample in Phase II for all three functions for efficiency.

It is possible, however, to select seeds from the entire set $\mathcal{N}$. In this case, an algorithm, referred to as SAS^{Gr}, applies *Greedy* instead of *Subsample Greedy*, with $ASR^{\mathbf{L}}$, ASR, and $ASR^{\mathbf{U}}$. Thus, SAS^{Gr} essentially employs the SA strategy (see Section 2); using the functions $ASR^{\mathbf{L}}$, ASR, and $ASR^{\mathbf{U}}$ as functions l_f, f, and u_f, respectively. We now show that SAS^{Gr} offers the following approximation guarantee.

Lemma 2. SAS^{Gr} *constructs a seed-set S satisfying*

$$\mathrm{ASR}(S,c) \geq \max\left\{\frac{c}{\sigma_{\mathcal{V}}(S^{\mathbf{U}},c)+c}, \frac{\sigma_{\mathcal{V}}(S^*,c)+c}{|\mathcal{V}|+c}\right\} \cdot (1-\frac{1}{\mathrm{e}}) \cdot \mathrm{ASR}(S^*,c) \tag{21}$$

where $S^* = \arg\max_{S\subseteq\mathcal{N},|S|\leq k} \mathrm{ASR}(S,c)$ *is an optimal seed-set of size at most k with respect to* ASR, e *is the base of the natural logarithm, and* $S^{\mathbf{U}}$ *is the seed-set obtained by applying Greedy with* $\mathrm{ASR}^{\mathbf{U}}$.

Proof. Given a non-negative non-submodular function $f: 2^U \to \mathbb{R}_{\geq 0}$, non-negative monotone submodular functions $l_f: 2^U \to \mathbb{R}_{\geq 0}$ and $u_f: 2^U \to \mathbb{R}_{\geq 0}$ such that $l_f(S) \leq f(S) \leq u_f(S)$ for each subset $S \subseteq U$, and a parameter k, the SA strategy finds a seed-set S that satisfies $f(S) \geq \max\left\{\frac{f(S_{u_f})}{u_f(S_{u_f})}, \frac{l_f(S^*)}{f(S^*)}\right\} \cdot (1-\frac{1}{\mathrm{e}}) \cdot f(S^*)$, where $S^* = \arg\max_{S\subseteq U,|S|\leq k} f(S)$ is an optimal subset of size at most k with respect to f [32]. SAS^{Gr} applies the SA strategy with $ASR^{\mathbf{L}}$, ASR, and $ASR^{\mathbf{U}}$ as functions l_f, f, and u_f, respectively. Thus, it achieves the following guarantee

$$ASR(S,c) \geq \max\left\{\frac{ASR(S^{\mathbf{U}},c)}{ASR^{\mathbf{U}}(S^{\mathbf{U}},c)}, \frac{ASR^{\mathbf{L}}(S^*,c)}{ASR(S^*,c)}\right\} \cdot (1-\frac{1}{\mathrm{e}}) \cdot ASR(S^*,c). \tag{22}$$

The statement of the lemma follows from Equation (22) and the definitions of $ASR^{\mathbf{L}}$, ASR, and $ASR^{\mathbf{U}}$. □

The algorithm in Lemma 2 performs $O(|\mathcal{N}|\cdot k)$ [1] evaluations of the spread function, hence it is much slower than *SAS* in practice when k is large.

6. The *ISS* algorithm for maximizing *ASR*

This section presents *ISS* (Iterative Subsample with Spread bounds), starting from the bound functions of *ASR* that *ISS* employs.

Lower and upper bound function of *ASR*. Our *ISS* algorithm employs the following two functions that bound *ASR* from below and from above, respectively:

$$\widetilde{ASR^{\mathbf{L}}}(S,c,Y) = \frac{\sigma_{\mathcal{N}}(S)+c}{\widehat{\sigma_{\mathcal{V},Y}}(S)+c} = \frac{\sigma_{\mathcal{N}}(S)+c}{\sigma_{\mathcal{V}}(Y) + \sum\limits_{u\in S\setminus Y}\sigma_{\mathcal{V}}(\{u\}) - \sum\limits_{u\in Y\setminus S}(\sigma_{\mathcal{V}}(Y)-\sigma_{\mathcal{V}}(Y\setminus\{u\}))+c}$$

$$\widetilde{ASR^{\mathbf{U}}}(S,c,\pi^Y) = \frac{\sigma_{\mathcal{N}}(S)+c}{\widetilde{\sigma_{\mathcal{V},\pi^Y}}(S)+c} = \frac{\sigma_{\mathcal{N}}(S)+c}{\sum\limits_{u\in S}(\sigma_{\mathcal{V},Y,\pi^Y}(u))+c}$$

where $Y \subseteq \mathcal{N}$ is the *parameter* in each bound function, and

$$\sigma_{\mathcal{V},Y,\pi^Y}(u) = \begin{cases} \sigma_{\mathcal{V}}(\pi_u^Y) - \sigma_{\mathcal{V}}(\pi_{u^-}^Y) & \text{, if } u \in Y \\ 0 & \text{, otherwise.} \end{cases}$$

$\widetilde{ASR^{\mathbf{L}}}$ is obtained by replacing $\sigma_{\mathcal{V}}(S)$ in *ASR* with its modular upper bound $\widehat{\sigma_{\mathcal{V},Y}}(S)$ (see Equation (1)) and using the fact that $\sigma_{\mathcal{V}}(\{\}) = 0$. $\widetilde{ASR^{\mathbf{U}}}$ is obtained by replacing $\sigma_{\mathcal{V}}(S)$ in *ASR* with its modular lower bound $\widetilde{\sigma_{\mathcal{V},\pi^Y}}(S)$ (see Equation (2)).

These bound functions differ from those used in *SAS* in that their denominators are modular functions that depend on a parameter (Y for the upper bound function and π^Y for the lower bound function, respectively). Also, they differ in that they are non-monotone non-submodular, as shown below. These properties are important for designing our *ISS* algorithm.

Lemma 3. $\widetilde{\mathrm{ASR}^{\mathbf{L}}}$, *as well as* $\widetilde{\mathrm{ASR}^{\mathbf{U}}}$, *is non-monotone and non-submodular.*

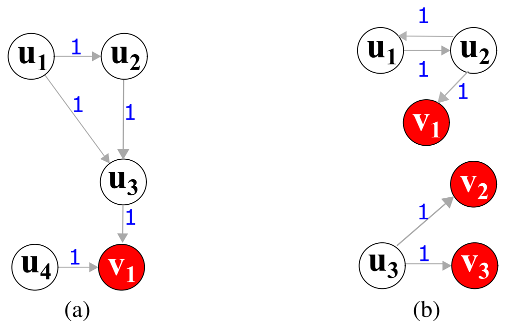


Fig. 4. (a) Example graph. $\mathcal{N}=\{u_1,u_2,u_3,u_4\}$, $\mathcal{V}=\{v_1\}$, and each edge probability is equal to 1. (b) Example graph. $\mathcal{N}=\{u_1,u_2,u_3\}$, $\mathcal{V}=\{v_1,v_2,v_3\}$, and each edge probability is equal to 1.

Proof. We first prove that $\widetilde{ASR^{\mathbf{L}}}$ is non-monotone by means of a counterexample. Consider the graph in Fig. 2 and let $c=1$ and $Y=S'=\{\}$. Since $\widetilde{ASR^{\mathbf{L}}}(S_1,1,S')=4/4>\widetilde{ASR^{\mathbf{L}}}(S_1\cup\{u_1\},1,S')=5/6$, $\widetilde{ASR^{\mathbf{L}}}$ is non-monotone.

We now prove that $\widetilde{ASR^{\mathbf{L}}}$ is non-submodular by means of a counterexample. Consider the graph in Fig. 4a, whose set of nodes is partitioned into $\mathcal{N}=\{u_1,\dots,u_4\}$ and $\mathcal{V}=\{v_1\}$. Let $c=1$ and $Y=\{u_4\}$. Let also $S=\{u_1\}\subseteq S'=\{u_1,u_2\}$ and $u=u_3\in\mathcal{N}\setminus S'$. Since $\widetilde{ASR^{\mathbf{L}}}(S\cup\{u\},c,Y)-\widetilde{ASR^{\mathbf{L}}}(S,c,Y)=\frac{3+c}{2+c}-\frac{3+c}{1+c}=-\frac{2}{3}<\widetilde{ASR^{\mathbf{L}}}(S'\cup\{u\},c,Y)-\widetilde{ASR^{\mathbf{L}}}(S',c,Y)=\frac{3+c}{3+c}-\frac{3+c}{2+c}=-\frac{1}{3}$, $\widetilde{ASR^{\mathbf{L}}}$ is non-submodular with respect to a seed-set S.

We also prove that $\widetilde{ASR^{\mathbf{U}}}$ is non-monotone by means of a counterexample. Consider the graph in Fig. 4b, whose set of nodes is partitioned into $\mathcal{N}=\{u_1,u_2,u_3\}$ and $\mathcal{V}=\{v_1,v_2,v_3\}$. Let $c=1$, $Y=\{u_3,u_2\}$, and $\pi^Y=(u_3,u_2)$. Let also $S=\{u_1\}\subseteq S'=\{u_1,u_3\}$ and $c=1$. Since $\widetilde{ASR^{\mathbf{U}}}(S,1,\pi^Y)=\frac{2+c}{c}=3>\widetilde{ASR^{\mathbf{U}}}(S',1,\pi^Y)=\frac{3+c}{2+c}=\frac{4}{3}$, $\widetilde{ASR^{\mathbf{U}}}$ is non-monotone.

Last, we prove that $\widetilde{ASR^{\mathbf{U}}}$ is non-submodular by means of a counterexample. Consider the graph in Fig. 4b, whose set of nodes is partitioned into $\mathcal{N}=\{u_1,u_2,u_3\}$ and $\mathcal{V}=\{v_1,v_2,v_3\}$. Let $c=1$, $Y=\{u_3,u_2\}$, and $\pi^Y=(u_3,u_2)$. Let also $S=\{u_1\}\subseteq S'=\{u_1,u_3\}$ and $u=u_2\in\mathcal{N}\setminus S'$. Since $\widetilde{ASR^{\mathbf{U}}}(S\cup\{u\},c,\pi^Y)-\widetilde{ASR^{\mathbf{U}}}(S,c,\pi^Y)=\frac{2+c}{1+c}-\frac{2+c}{c}=-\frac{3}{2}<\widetilde{ASR^{\mathbf{U}}}(S'\cup\{u\},c,\pi^Y)-\widetilde{ASR^{\mathbf{U}}}(S',c,\pi^Y)=\frac{3+c}{3+c}-\frac{3+c}{2+c}=-\frac{1}{3}$, $\widetilde{ASR^{\mathbf{U}}}$ is non-submodular with respect to a seed-set S. □

Operation of *ISS*. The algorithm works iteratively, as can be seen from the pseudocode. In each iteration (i.e., execution of the while loop in Step 3), it creates a seed-set S_{cur} in three phases: (I) dummy element creation; (II) construction of three candidates seed-sets (one using *ASR*, a second using $\widetilde{ASR^{\mathbf{L}}}$ and a third using $\widetilde{ASR^{\mathbf{U}}}$); and (III) selection of the best candidate seed-set and removal of dummy elements from it. The iterations stop when S_{cur} is not better than the previously created seed-set S_{pr} in terms of *ASR* (Steps 21-22). This guarantees that the algorithm terminates [20].

Phase I (Steps 4 to 7): This phase is identical to Phase I of *SAS*. Note that a *dummy* element cannot be thought of as a node with no edges. This is because adding such a node into any S would increase $\sigma_{\mathcal{N}}(S)$ by 1, whereas adding a dummy node into any S does not change $\sigma_{\mathcal{N}}(S)$.

Phase II (Steps 8 to 18): A random sample of $\frac{|\mathcal{N}'|}{k}$ elements from $\mathcal{N}'$ and a dummy element is created. Next, a node in the sample causing the largest marginal gain with respect to *ASR*, $\widetilde{ASR^{\mathbf{L}}}$, and $\widetilde{ASR^{\mathbf{U}}}$ is added into the candidate subset $S_i^{\mathbf{O}}$, $S_i^{\mathbf{U}}$, and $S_i^{\mathbf{L}}$, respectively. The parameter of $\widetilde{ASR^{\mathbf{L}}}$ is the seed-set S_{pr}, constructed in the previous iteration and that of $\widetilde{ASR^{\mathbf{U}}}$ is a random permutation $\pi^{S_{pr}}$ of S_{pr}.

Phase III (Steps 19 to 23): The best candidate subset with respect to *ASR* is selected as S_{cur}, and all dummy elements are removed from it. If S_{cur} is not better than S_{pr} in terms of *ASR*, the while loop in Step 3 is terminated and S_{cur} is returned. Otherwise, another iteration is performed with the aim of generating a seed-set with larger *ASR*, due to the use of different (and often better [20]) lower and upper bounds (i.e., a lower bound with a different parameter S_{pr} and upper bound with a different parameter S_{pr} and random permutation $\pi^{S_{pr}}$).

Relation of *ISS* to other methods. Alike *SAS*, *ISS* selects seeds from a sample of $\mathcal{N}$ of size approximately $\frac{|\mathcal{N}|}{k}$, instead of the entire set $\mathcal{N}$. This is different from *GR* which selects seeds from the entire set $\mathcal{N}$ and thus scales worse with k.

However, *ISS* differs from *SAS* in that it is iterative and in that it, in each iteration, its bound functions have a different seed-set as parameter. In our experiments, we observed that more iterations result in seed-sets with larger *ASR*. This is because the parameter S_{pr} of the lower bound function $\widetilde{ASR^{\mathbf{L}}}$, as well as S_{pr} and $\pi^{S_{pr}}$ of the upper bound function $\widetilde{ASR^{\mathbf{U}}}$, are updated in every iteration which often improves the bound functions [20]. We also observed that *ISS* needed at most 4 iterations to terminate.

```
Algorithm: ISS (Iterative Subsample with Spread bounds)
Input: Set of non-vulnerable nodes $\mathcal{N} \subseteq V$, set of vulnerable nodes $\mathcal{V} \subseteq V$, graph $G$, parameter $k$, constant $c$
Output: Subset $S \subseteq \mathcal{N}$ of size $|S| \leq k$
1  $S_{pr} \leftarrow \{\}$
2  $S_{cur} \leftarrow \mathcal{N}$
3  while true do
       // Phase I
4      $\mathcal{D} \leftarrow$ set of $k$ dummy elements $\{u_1, \ldots, u_k\}$ such that, for each element $u_i$, $i \in [1,k]$, and every set $S \subseteq \mathcal{N}$:
        $\sigma_{\mathcal{N}}(S \cup \{u_i\}) = \sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S \cup \{u_i\}) = \sigma_{\mathcal{V}}(S)$
5      $\mathcal{N}' \leftarrow \mathcal{N}$
6      while $\frac{|\mathcal{N}'|}{k}$ is not an integer do
7          Add into $\mathcal{N}'$ a dummy element $u' \notin \mathcal{D}$ such that $\sigma_{\mathcal{N}}(S \cup \{u'\}) = \sigma_{\mathcal{N}}(S)$ and $\sigma_{\mathcal{V}}(S \cup \{u'\}) = \sigma_{\mathcal{V}}(S)$
       // Phase II
8      $i \leftarrow 0$; $S_0^{\mathbf{O}} \leftarrow \{\}$; $S_0^{\mathbf{L}} \leftarrow \{\}$; $S_0^{\mathbf{U}} \leftarrow \{\}$
9      while $i < k$ do
10         $\mathcal{R} \leftarrow$ uniform random sample of $\mathcal{N}'$ with $\frac{|\mathcal{N}'|}{k}$ elements
11         Add into $\mathcal{R}$ a random element from $\mathcal{D}$
12         $u^{\mathbf{O}} \in \arg\max_{u \in \mathcal{R}} (ASR(S_i^{\mathbf{O}} \cup \{u\}, c) - ASR(S_i^{\mathbf{O}}, c))$
13         $S_{i+1}^{\mathbf{O}} \leftarrow S_i^{\mathbf{O}} \cup \{u^{\mathbf{O}}\}$
14         $u^{\mathbf{L}} \in \arg\max_{u \in \mathcal{R}} (\widetilde{ASR^{\mathbf{L}}}(S_i^{\mathbf{L}} \cup \{u\}, c, S_{pr}) - \widetilde{ASR^{\mathbf{L}}}(S_i^{\mathbf{L}}, c, S_{pr}))$
15         $S_{i+1}^{\mathbf{L}} \leftarrow S_i^{\mathbf{L}} \cup \{u^{\mathbf{L}}\}$
16         $u^{\mathbf{U}} \in \arg\max_{u \in \mathcal{R}} (\widetilde{ASR^{\mathbf{U}}}(S_i^{\mathbf{U}} \cup \{u\}, c, \pi^{S_{pr}}) - \widetilde{ASR^{\mathbf{U}}}(S_i^{\mathbf{U}}, c, \pi^{S_{pr}}))$
17         $S_{i+1}^{\mathbf{U}} \leftarrow S_i^{\mathbf{U}} \cup \{u^{\mathbf{U}}\}$
18         $i \leftarrow i+1$
       // Phase III
19     $S_{cur} \leftarrow \arg\max_{S \in \{S_k^{\mathbf{O}}, S_k^{\mathbf{L}}, S_k^{\mathbf{U}}\}} ASR(S, c)$
20     $S_{cur} \leftarrow$ Remove all dummy elements from $S_{cur}$
21     if ASR$(S_{cur}, c) \leq$ ASR$(S_{pr}, c)$ then
22         break
23     $S_{pr} \leftarrow S_{cur}$
24 return $S_{cur}$
```

Cost of *ISS*. The number of evaluations of the spread function performed by *ISS* is given in the proposition below, where I denotes the number of iterations.

Proposition 2. ISS *performs $O(|\mathcal{N}| \cdot I)$ evaluations of the spread function.*

Proof. The statement follows directly from the fact that Subsample Greedy performs $O(|\mathcal{N}|)$ evaluations of the spread function [31] and from the fact that *ISS* executes this algorithm on three functions (*ASR* and the bound functions) once per iteration. □

As discussed in Section 5, the cost of evaluating the spread function depends on the method used for computing the spread, and this cost is $O(|\mathcal{N}|^2|E|)$ when the method of [25] is used. Thus, by Proposition 2, when spread is computed using the dynamic programming algorithm of [25], the cost of *ISS* is $O(|\mathcal{N}|^3 \cdot |E| \cdot I)$. As mentioned above, *ISS* always terminates, so I is bounded.

7. Variations of *ISS*

In the following, we discuss two variations of *ISS* that explore different trade-offs between efficiency and effectiveness.

***ISS*$^{\mathbf{U}}$ algorithm.** We consider a variation of *ISS* that applies Subsample Greedy only to the upper bound function $\widetilde{ASR^{\mathbf{U}}}$ of *ASR*. We refer to this algorithm as *ISS*$^{\mathbf{U}}$. *ISS*$^{\mathbf{U}}$ was inspired by a heuristic [32] that applied Greedy only to the upper bound function u_f used in the SA strategy (see Section 2).

The benefit of *ISS*$^{\mathbf{U}}$ is that it is more efficient than *ISS*. This is mainly because *ISS*$^{\mathbf{U}}$ executes Subsample Greedy once (with $\widetilde{ASR^{\mathbf{U}}}$), whereas *ISS* executes Subsample Greedy three times (with $\widetilde{ASR^{\mathbf{U}}}$, $\widetilde{ASR^{\mathbf{L}}}$, and $ASR^{\mathbf{U}}$). In fact, in our experiments, we show that *ISS*$^{\mathbf{U}}$ is one order of magnitude faster than *ISS* on average.

***ISS*^{Gr} algorithm.** *ISS*$^{\mathbf{Gr}}$ is a variation of *ISS* in which *Greedy* is used instead of *Subsample Greedy*. Although applying *Greedy* to non-monotone non-submodular functions, such as *ASR*, $\widetilde{ASR^{\mathbf{U}}}$ and $\widetilde{ASR^{\mathbf{L}}}$, still provides no approximation guarantees, *ISS*$^{\mathbf{Gr}}$ performs very well in practice (see Section 8). This experimental finding agrees with several studies demonstrating that *Greedy* can be considered as an effective heuristic for non-monotone and/or non-submodular functions (e.g., [34,35]) because influence functions tend to be “close” to being submodular (e.g., [36,37]).

Table 2
Characteristics of the datasets we used and default values for the maximum probability threshold θ.

Dataset	# of nodes ($\lvert V\rvert$)	# of edges ($\lvert E\rvert$)	avg in-degree	max in-degree	# of vuln. nodes ($\lvert\mathcal{V}\rvert$)	max prob. threshold θ
WI	7115	103689	13.7	452	100	0.01
TW	235	2479	10.5	52	25	0.01
POL	1490	19090	11.9	305	100	0.003
AB	840	10008	11.9	137	10	0.01

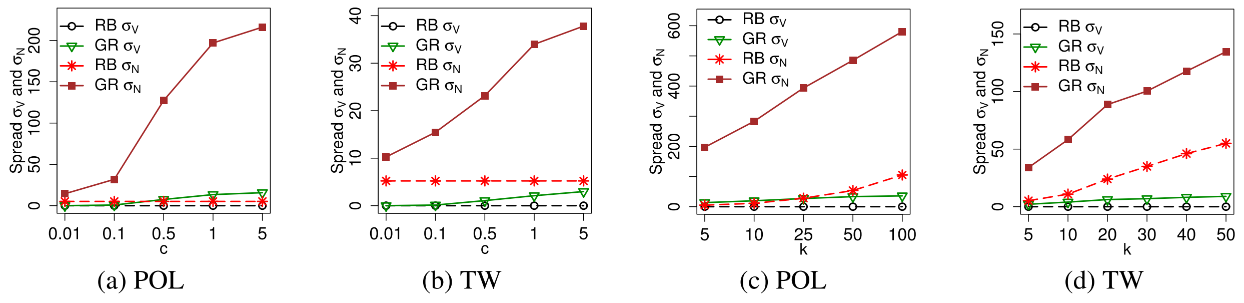


Fig. 5. Spread of vulnerable and non-vulnerable nodes: (a) POL vs. c, (b) TW vs. c, (c) POL vs. k, and (d) TW vs. k.

8. Experimental evaluation

In this section, we evaluate our methods *GR*, GR_{MB}, *SAS*, *ISS*, $ISS^{\mathbf{U}}$, and $ISS^{\mathbf{Gr}}$, in terms of effectiveness and efficiency, by comparing them against *TIM* [6], a heuristic for finding a seed-set S with size at most k and large $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$, and two baselines that employ *Greedy* [14]: *RB*, which applies *Greedy* [14] to the subset of non-vulnerable nodes that do not influence vulnerable nodes, and *RB*′, which applies *Greedy* with the objective function $\sigma_{\mathcal{N}}$. *RB* creates a seed-set S with $\sigma_{\mathcal{V}}(S) = 0$ and was used to see whether S can have large $\sigma_{\mathcal{N}}(S)$. *RB*′ creates a seed-set S with large $\sigma_{\mathcal{N}}(S)$ and was used to see whether S can have small $\sigma_{\mathcal{V}}(S)$. *RB*′ found seed-sets that influenced many more vulnerable nodes than those of all other methods, thus, we omit its results. Since the bound functions used in $SAS^{\mathbf{Gr}}$ only depend on $\sigma_{\mathcal{V}}$, $SAS^{\mathbf{Gr}}$ finds the same seed-set as *RB*′. Hence, we do not present results for $SAS^{\mathbf{Gr}}$.

All algorithms were implemented in C++ and applied to the *Wiki-vote* (WI), Twitter (TW), and *PolBlogs* (POL) datasets (see Table 2).[2] We also used synthetic datasets, generated by the Albert-Barabasi model, as in [8], with a varying number of edges in [500, 10000]. We refer to the dataset with 10000 edges as *AB*. We set $p(u', u) = \frac{1}{|n^-(u)|}$ for each edge (u', u) as in [38,8]. We also set θ, the maximum probability threshold for a path, to a small value (see Table 2), so that all methods achieve a good accuracy/efficiency trade-off by discarding paths that have smaller than θ probability to influence a node, as in [39]. The default value for k was 5 and for c was 1.

The set of vulnerable nodes $\mathcal{V}$ was constructed by selecting nodes: (a) randomly, (b) based on their out-degree, and (c) based on their PageRank score. We consider setting a unless otherwise specified. In settings b and c, the nodes with the largest out-degree or PageRank score were selected. Nodes with large PageRank scores can influence many other nodes when activated [8], and the same holds for nodes with large out-degree [1]. Thus, the settings b and c are challenging, because it is difficult to reduce the spread of such vulnerable nodes while achieving a large spread for the non-vulnerable nodes. The default number of vulnerable nodes $|\mathcal{V}|$ for each dataset is shown in Table 2.

To improve the efficiency of *ISS*, we used the CELF optimization [1] for the submodular bound functions (Steps 14 and 16). The results for all randomized algorithms (e.g., *SAS* and *ISS*) were averaged over 10 runs.

All experiments ran on an Intel Xeon CPU E5-2640 @2.66 GHz with 64GB RAM. For brevity, we omit some results that were qualitatively similar to the reported ones (e.g., results for varying $|\mathcal{V}|$ in the WI dataset).

Comparison to *RB*. *GR* constructs seed-sets that influence at least 5.5 and up to 38 times more non-vulnerable nodes than those constructed by *RB*, for different values of c (see Figs. 5a and 5b) and k (see Figs. 5c and 5d). The reason is that, for all c and k values, vulnerable nodes were distributed across the graph. So, the seed-sets constructed by *RB* did not influence vulnerable nodes, but they also did not influence many non-vulnerable nodes which defeats the main purpose of influence maximization that is to inform many non-vulnerable users. On the other hand, the seed-sets constructed by *GR* influenced a small number of vulnerable nodes but could reach to and influence many more non-vulnerable nodes. Moreover, *TIM*, GR_{MB}, *SAS*, and *ISS* outperformed *RB* (the results for them are omitted). Thus, in all subsequent experiments, we omit results for *RB*, since it does not construct practically useful solutions and set $c = 1$ because this allows all methods to construct seed-sets with good $\sigma_{\mathcal{N}}/\sigma_{\mathcal{V}}$ trade-off.

[2] POL is available at http://www-personal.umich.edu/~mejn/ and all other datasets at http://snap.stanford.edu/data.

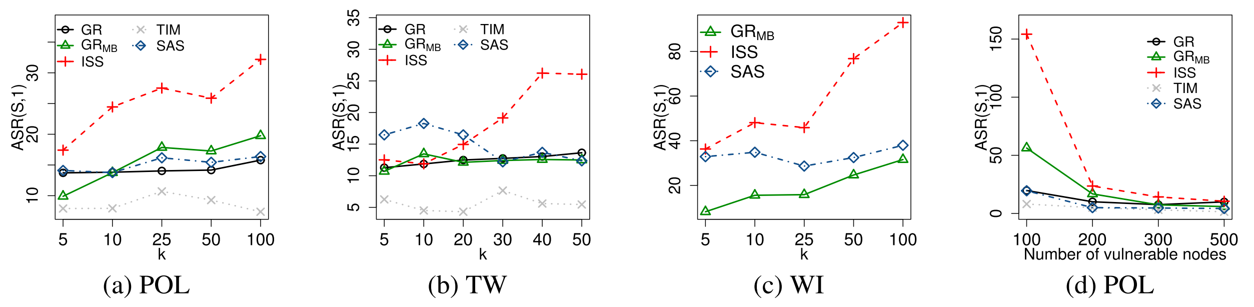


Fig. 6. *ASR* with $c = 1$ vs. k for (a) POL, (b) TW, and (c) WI. (d) *ASR* with $c = 1$ vs. $|\mathcal{V}|$.

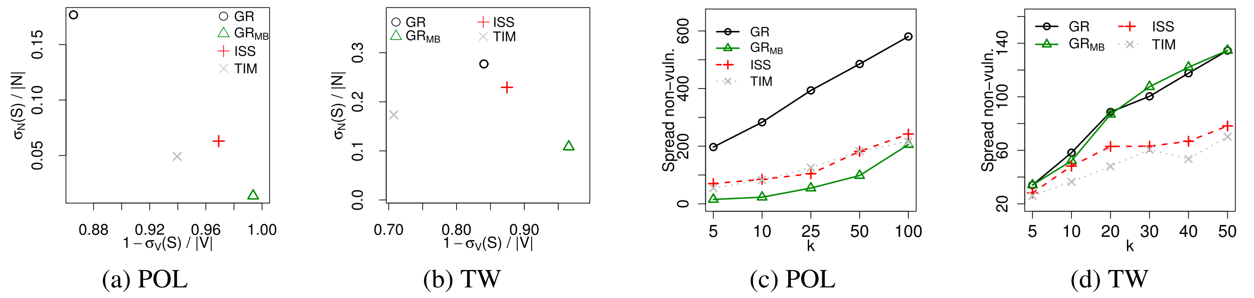


Fig. 7. Map for (a) POL and $k = 5$, (b) TW and $k = 10$ (larger values in x and y axis imply better *protection* and *utility*, respectively). Spread of non-vulnerable nodes vs. k for (c) POL, and (d) TW.

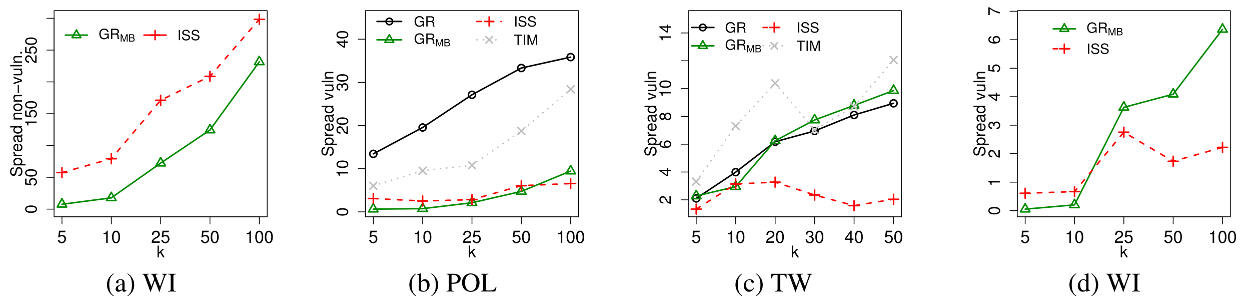


Fig. 8. Spread of non-vulnerable nodes vs. k for (a) TW, and (b) WI. Spread of vulnerable nodes vs. k for (c) POL. Spread of vulnerable nodes for (d) TW, and (e) WI.

***ASR* with $c = 1$.** All our algorithms substantially outperform *TIM* in terms of *ASR* for varying k (see Figs. 6a, 6b, and 6c) and varying number of vulnerable nodes $|\mathcal{V}|$ (see Fig. 6d). *ISS* outperformed all other methods, being 3, 1.7, 2 and 1.6 times better than *TIM*, *GR*, *GR*$_{\text{MB}}$ and *SAS* on average (over all datasets and k values), respectively. *ISS* was also 8.9, 3.3, 1.9 and 4.7 times better than *TIM*, *GR*, *GR*$_{\text{MB}}$ and *SAS* on average (over all $|\mathcal{V}|$ values in Fig. 6d), respectively. We omit the results for *GR* and *TIM* for the largest dataset *WI* from all subsequent experiments, since *GR* and *TIM* did not finish within 3 days. We also omit the results for *SAS* from all subsequent experiments, since it performed worse than *ISS* on average. The better performance of *ISS* compared to *SAS* is attributed to its bound functions, which depend on the seed-set and improve as more iterations are performed.

Spread of vulnerable and non-vulnerable nodes. We demonstrate that all our algorithms substantially outperform *TIM* in terms of $\sigma_{\mathcal{N}}$ and/or $\sigma_{\mathcal{V}}$. First, we report Figs. 7a and 7b, where each point (x, y) corresponds to the values $(1 - \frac{\sigma_{\mathcal{V}}(S)}{|\mathcal{V}|}, \frac{\sigma_{\mathcal{N}}(S)}{|\mathcal{N}|})$, referred to as *protection* and *utility* of a seed-set S. Note that larger values in protection and utility are preferred. *ISS* outperformed *TIM* with respect to both protection and utility. Also, *ISS* achieved overall better protection than *GR* and better utility than *GR*$_{\text{MB}}$. We also report $\sigma_{\mathcal{N}}$ and $\sigma_{\mathcal{V}}$ in Figs. 7c to 8d. *GR* and *TIM* constructed seed-sets that influence too many vulnerable nodes. *GR*$_{\text{MB}}$ performed inconsistently (e.g., its seed-sets influenced few vulnerable nodes in Fig. 8b and too many vulnerable nodes in Fig. 8c). *ISS* influenced few vulnerable nodes and a moderate number of non-vulnerable nodes, achieving a good $\sigma_{\mathcal{N}}/\sigma_{\mathcal{V}}$ trade-off.

Impact of vulnerable node selection. We demonstrate that our algorithms substantially outperform *TIM* in terms of *ASR* for different strategies of selecting vulnerable nodes. This can be seen from Figs. 9a and 9b (respectively, Figs. 9c and 9d), which report *ASR* when the vulnerable nodes were selected as the nodes with the largest PageRank scores (respectively,

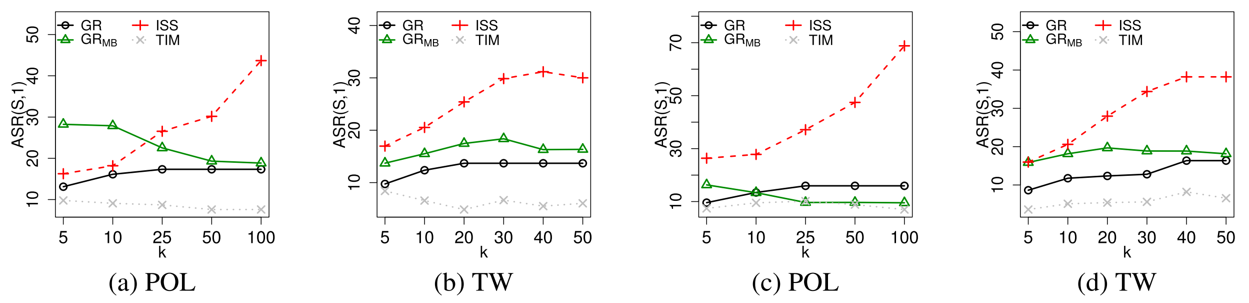


Fig. 9. *ASR* with $c = 1$ vs. k when vulnerable nodes are selected as the nodes with the largest PageRank scores for (a) POL, and (b) TW. *ASR* with $c = 1$ vs. k when vulnerable nodes are selected as the nodes with the largest out-degree for (c) POL, and (d) TW.

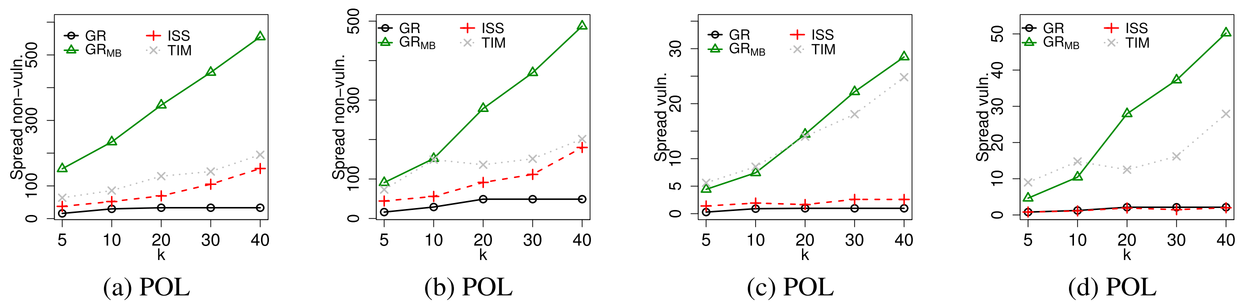


Fig. 10. Spread of non-vulnerable nodes vs. k for POL when the vulnerable nodes are selected as the nodes with the largest (a) PageRank score, and (b) out-degree. Spread of vulnerable nodes vs. k for POL when the vulnerable nodes are selected as the nodes with the largest (a) PageRank score, and (b) out-degree.

out-degree). In these experiments, *ISS* was 4.4, 2.2, and 2.1 times on average better in terms of *ASR* than *TIM*, *GR*, and GR_{MB}, respectively. In addition, *GR* and GR_{MB} outperformed *TIM* by 2.02 and 2.57 times on average, respectively. The results are qualitatively similar to those in Fig. 6.

We also demonstrate that *ISS* is the only algorithm that achieves large spread of non-vulnerable nodes $\sigma_{\mathcal{N}}$ and well as small spread of vulnerable nodes $\sigma_{\mathcal{V}}$ (i.e., it achieves a good trade-off). This can be seen from Figs. 10a and 10b, which report the spread of non-vulnerable nodes, and from Figs. 10c and 10d, which report the spread of vulnerable nodes. This is an encouraging result because all other algorithms either influenced a large (expected) number of vulnerable nodes, or did not influence a large (expected) number of non-vulnerable nodes. In both cases, the result is less useful for viral marketing. Specifically, the seed-sets constructed by *GR* had a smaller $\sigma_{\mathcal{N}}$ than those of *ISS* by 35.4% on average. Thus, *GR* influenced a smaller (expected) number of non-vulnerable nodes compared to *ISS*. On the other hand, *TIM* and GR_{MB} have a larger $\sigma_{\mathcal{V}}$ than that of *ISS* by 9.8 and 7.55 times on average, respectively. Thus, these two algorithms influenced a larger (expected) number of vulnerable nodes compared to the *ISS* algorithm.

Impact of number of iterations in *ISS*. We demonstrate that *ISS* can find a better seed-set in terms of *ASR* by performing iterations with different lower bounds (see the while loop in Step 3 of *ISS* and note that the parameter S_{pr} changes in Step 23). In particular, we measured the relative improvement in terms of *ASR* from the first to the last iteration of *ISS*, which is defined as

$$\mathcal{RI} = \frac{ASR(S^{l}_{cur}, c) - ASR(S^{1}_{cur}, c)}{ASR(S^{1}_{cur}, c)},$$

where S^{l}_{cur} (resp., S^{1}_{cur}) is the seed-set constructed by *ISS* in the last (resp., first) iteration and $c = 1$. Table 3a shows the maximum $\mathcal{RI}$, which varies from 27.7% to 93.85%. On average (over all k values and datasets) the maximum $\mathcal{RI}$ was 46%. Table 3b shows the average $\mathcal{RI}$, computed over 10 different runs of *ASR*, which varies from 13.8% to 48.31% and has an average value of 24.3% (over all k values and dataset). The fact that the maximum and average $\mathcal{RI}$ increase substantially implies that the iterative scheme in *ISS* (see Step 3) is able to trade-off efficiency for effectiveness.

Efficiency. All our methods are much faster than *TIM* for varying k (see Figs. 11a and 11b). *TIM* required 10 hours when $k = 50$ in the case of TW which only has 235 nodes, and 17 days when $k = 25$ in the case of POL. *GR* was faster but did not terminate within 3 days in the case of WI, and GR_{MB} was the fastest due to its efficient spread estimation function [24]. *ISS* was significantly faster than *GR* and *TIM* and the most scalable method with respect to k. Fig. 11c shows the runtime for varying $|\mathcal{V}|$. All our algorithms become faster with $|\mathcal{V}|$, since fewer nodes can be selected as seeds and are at least three orders of magnitude faster than *TIM* on average. Fig. 11d shows the runtime for varying number of edges using the

Table 3
(a) Maximum and (b) Average $\mathcal{RI}$ with $c = 1$ as well as number of iterations performed by *ISS* vs. k, for TW, POL and WI.

	TW		POL		WI	
k	Max. $\mathcal{RI}$	# iterations	Max. $\mathcal{RI}$	# iterations	Max. $\mathcal{RI}$	# iterations
5	36.17%	3	31.06%	4	27.7%	3
10	33.74%	3	49.06%	4	43.12%	3
50	58.66%	3	93.85%	3	43.06%	3

	TW		POL		WI	
k	Avg. $\mathcal{RI}$	# iterations	Avg. $\mathcal{RI}$	# iterations	Avg. $\mathcal{RI}$	# iterations
5	13.8%	3	13.49%	3	23.43%	2
10	20.32%	2	21.09%	2	25.45%	3
50	26.26%	2	48.31%	3	27.87%	2

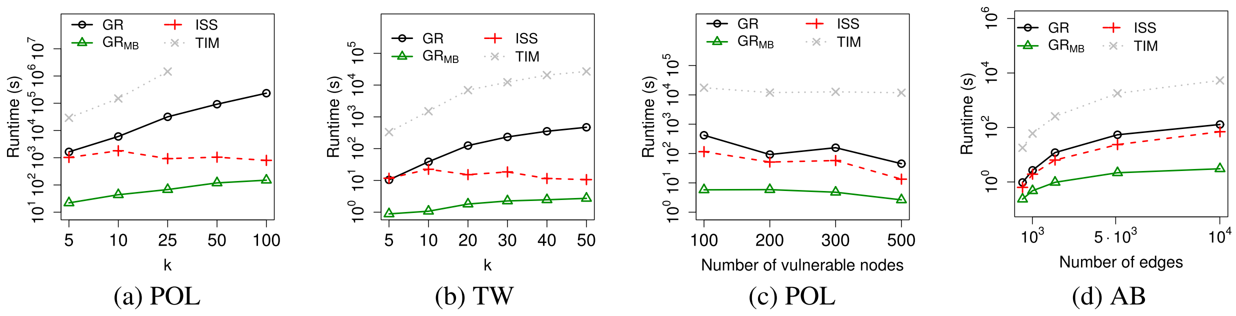


Fig. 11. Runtime vs. k for (a) POL, and (b) TW. Runtime vs. (c) number of vulnerable nodes for POL, and (d) number of edges for AB.

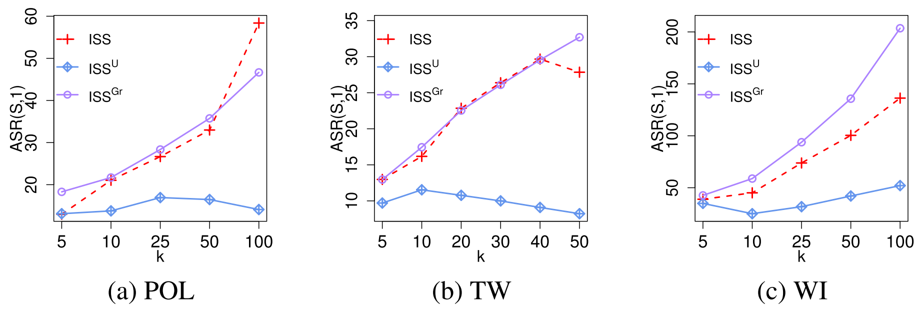


Fig. 12. *ASR* with $c = 1$ vs. k for (a) POL, (b) TW, and (c) WI.

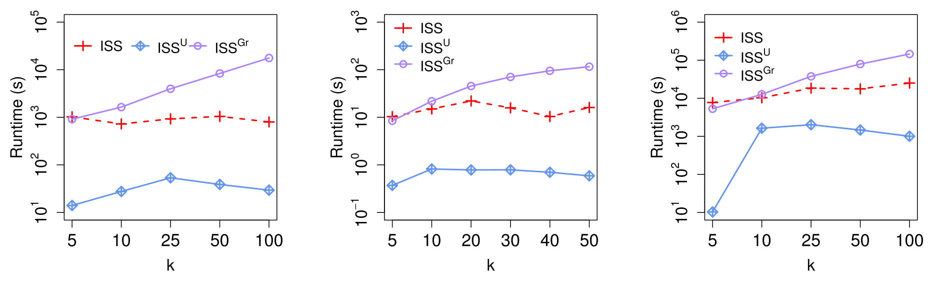
synthetic dataset *AB*. Our algorithms were faster than *TIM* by up to three orders of magnitude. Results on other datasets were similar (omitted for brevity).

Comparison of *ISS* to *ISS*$^{\mathbf{U}}$ and *ISS*$^{\mathbf{Gr}}$. We demonstrate that *ISS* is comparable to *ISS*$^{\mathbf{Gr}}$ in terms of effectiveness, while being much more efficient (Figs. 12 and 13). Specifically, the *ASR* scores for *ISS* are approximately 12% on average lower (worse) than those for *ISS*$^{\mathbf{Gr}}$, but *ISS* is 5 times faster on average, and it scales better with k. The *ISS*$^{\mathbf{U}}$ algorithm was faster than *ISS* by one order of magnitude on average, but it performed much worse in terms of *ASR*; its scores were 2 times lower on average, over all k values and datasets than those of *ISS*. Thus, the experiments demonstrate that *ISS* offers a good balance between the slightly more effective but much slower *ISS*$^{\mathbf{Gr}}$ algorithm and the much less effective but faster *ISS*$^{\mathbf{U}}$ algorithm.

9. Related work

The problem of influence maximization has attracted substantial research interest (see [9] for a survey). However, no existing work aims to address influence maximization when there are vulnerable nodes. The most related works to ours are [6] and [8].

The work in [6] aims to maximize the difference between the expected number of influenced users who belong to a target group and the expected number of all other influenced users. Our work differs from [6] along three dimensions. First, [6] can select target nodes as seeds, but we cannot do the same for vulnerable nodes, as this would harm them. Second, our *ASR* measure has desired properties unlike the measure $\sigma_{\mathcal{N}}(S) - \sigma_{\mathcal{V}}(S)$ in [6] (see Section 3). Third, our methods are substantially more effective and efficient than the heuristic in [6] (see Section 8), while *SAS* also offers approximation guarantees.



(a) POL (b) TW (c) WI

Fig. 13. Runtime vs. k for (a) POL, (b) TW, and (c) WI.

The work in [8] is applied after influence maximization (i.e., it considers a given seed-set), and it also has different objectives and influence diffusion model compared to our work. Thus, it is orthogonal to our work. Specifically, it seeks to delete edges in order to limit the activation probability of vulnerable nodes in the Linear Threshold (LT) model [1]. Since edge deletion corresponds to blocking communication between users, the approach of [8] is more invasive than our approach which simply selects an appropriate seed-set. However, an approach based on edge deletion for the IC model would be interesting to develop and apply after our approach, in cases where the activation probability of each vulnerable node should be limited.

There are many works on targeted viral marketing (e.g., [40–43,6]). The work in [40] studied the problem of influence maximization in the presence of target nodes, under the IC model, and proposed greedy-based heuristics to tackle the problem. The work in [41] considered influence maximization when each target node has a constant profit, and [42] considered the impact of the location and login time of target nodes. The work in [43] considered the problem of revenue maximization under the IC and LT models. In this problem, a social network provider aims to perform the campaigns of different organizations wanting to promote complimentary products while maximizing its revenue. Unlike ours, the works in [41,43,42,6,40] do not consider vulnerable nodes.

There are also works on influence maximization considering nodes with negative impact on the influence diffusion process [38,44]. The work in [38] studied influence maximization under a model where each node can diffuse information of opposite content to the information that is being spread from the seed-set. The work in [44] studied influence maximization, when some nodes reject the diffused information. Different from these works, no node negatively impacts the influence diffusion process in our approach.

Last, there are works considering fairness in influence maximization (e.g., [45,46]). In these works, there are multiple groups of users, and the goal is to influence all of these groups in a “balanced” way, so that none is disadvantaged. The work in [45] aims at maximizing the influence over all groups, while lower-bounding the influence of a specified group, or alternatively, maximizing the influence of the specified group, while lower-bounding the influence of all groups. The work in [46] aims at maximizing the minimum fraction of influenced users within each group. Different from these works, we aim at minimizing the spread of a group of users while maximizing the spread of the other. Thus, the methods in [45,46] are not alternatives to ours.

10. Concluding remarks

In this paper, we study influence maximization when there are vulnerable nodes. We first propose a measure for limiting the influence to vulnerable nodes, which is obtained by applying additive smoothing to the ratio between the expected number of influenced non-vulnerable nodes and the expected number of influenced vulnerable nodes. Based on the measure, we define a new influence maximization problem that seeks to find a seed-set of size at most k that maximizes the measure. To solve our influence maximization problem, we propose two greedy baseline heuristics, the *SAS* approximation algorithm, as well as the *ISS* heuristic and two variations of it. We evaluate our methods on synthetic and real-world datasets and show that our methods outperform the method of [6] in terms of effectiveness and efficiency, while the variations of *ISS* offer differing trade-offs between effectiveness and efficiency.

In the future, we plan to study the *ASR*-Maximization problem under different influence diffusion models. These include time-aware models, such as [47], in which an activated node tries to influence its inactive out-neighbors after a random delay, as well as other models in which some activated nodes do not try to activate their out-neighbors [4].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Huiping Chen is supported by a CSC scholarship (CSC201809120007). We acknowledge the use of the Rosalind HPC cluster hosted by King's College London.

References

[1] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in: KDD, 2003, pp. 137–146.
[2] S. Cheng, H. Shen, J. Huang, G. Zhang, X. Cheng, Staticgreedy: solving the scalability-accuracy dilemma in influence maximization, in: CIKM '13, 2013, pp. 509–518.
[3] N. Ohsaka, Y. Yamaguchi, N. Kakimura, K. Kawarabayashi, Maximizing time-decaying influence in social networks, in: ECML PKDD, 2016, pp. 132–147.
[4] G. D'Angelo, L. Severini, Y. Velaj, Recommending links through influence maximization, Theor. Comput. Sci. 764 (2019) 30–41, Selected papers of ICTCS 2016 (the Italian Conference on Theoretical Computer Science (ICTCS).
[5] T. Maehara, H. Suzuki, M. Ishihata, Exact computation of influence spread by binary decision diagrams, in: WWW, 2017, pp. 947–956.
[6] R. Pasumarthi, R. Narayanam, B. Ravindran, Near optimal strategies for targeted marketing in social networks, in: AAMAS, 2015, pp. 1679–1680.
[7] N. Buchbinder, M. Feldman, J. Naor, R. Schwartz, Submodular maximization with cardinality constraints, in: SODA, 2014, pp. 1433–1452.
[8] G. Loukides, R. Gwadera, Preventing the diffusion of information to vulnerable users while preserving pagerank, I, J. Data Sci. Anal. 5 (2018) 19–39.
[9] Y. Li, J. Fan, Y. Wang, K. Tan, Influence maximization on social graphs: a survey, IEEE Trans. Knowl. Data Eng. 30 (2018) 1852–1872.
[10] S. Gupta, A conceptual framework that identifies antecedents and consequences of building socially responsible international brands, Thunderbird Int. Bus. Rev. 58 (2018) 225–237.
[11] G. Shaw, A. Karami, Computational content analysis of negative tweets for obesity, diet, diabetes, and exercise, Proc. Assoc. Inform. Sci. Technol. 54 (2017) 357–365.
[12] C. Communications, Corporate social responsibility study, http://www.conecomm.com/research-blog/2017-csr-study, 2017.
[13] Nielsen, Sustainable selections: how socially responsible companies are turning a profit, https://bit.ly/2DW99pE, 2015.
[14] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions, Math. Program. 14 (1978) 265–294.
[15] Z. Svitkina, L. Fleischer, Submodular approximation: sampling-based algorithms and lower bounds, SIAM J. Comput. 40 (2011) 1715–1737.
[16] W. Bai, R. Iyer, K. Wei, J. Bilmes, Algorithms for optimizing the ratio of submodular functions, in: ICML, 2016, pp. 2751–2759.
[17] D. Kempe, J. Kleinberg, E. Tardos, Influential nodes in a diffusion model for social networks, in: ICALP, 2005, pp. 1127–1138.
[18] A. Krause, D. Golovin, Submodular function maximization, in: Tractability, 2013, pp. 71–104.
[19] H. Chen, G. Loukides, J. Fan, H. Chan, Limiting the influence to vulnerable users in social networks: a ratio perspective, in: IEEE AINA, 2019, pp. 1106–1122.
[20] R. Iyer, J. Bilmes, Algorithms for approximate minimization of the difference between submodular functions, with applications, in: UAI, 2012, pp. 407–417.
[21] M. Conforti, G. Cornuéjols, Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem, Discrete Appl. Math. 7 (1984) 251–274.
[22] C. Qian, J. Shi, Y. Yu, K. Tang, Z. Zhou, Optimizing ratio of monotone set functions, in: IJCAI, 2017, pp. 2606–2612.
[23] W. Bai, J. Bilmes, Greed is still good: maximizing monotone submodular+supermodular (BP) functions, in: Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 304–313.
[24] C. Wang, W. Chen, Y. Wang, Scalable influence maximization for independent cascade model in large-scale social networks, DMKD 25 (2012) 545–576.
[25] R. Gwadera, G. Loukides, Cost-effective viral marketing in the latency aware independent cascade model, in: PAKDD, 2017, pp. 251–265.
[26] B. Liu, G. Cong, Y. Zeng, D. Xu, Y.M. Chee, Influence spreading path and its application to the time constrained social influence maximization problem and beyond, IEEE Trans. Knowl. Data Eng. 26 (2014) 1904–1917.
[27] Y. Tang, Y. Shi, X. Xiao, Influence maximization in near-linear time: a martingale approach, in: SIGMOD, 2015, pp. 1539–1554.
[28] K. Huang, S. Wang, G.S. Bevilacqua, X. Xiao, L.V.S. Lakshmanan, Revisiting the stop-and-stare algorithms for influence maximization, Proc. VLDB Endow. 10 (2017) 913–924.
[29] Y. Tang, X. Xiao, Y. Shi, Influence maximization: near-optimal time complexity meets practical efficiency, in: SIGMOD, 2014, pp. 75–86.
[30] M. Minoux, Accelerated greedy algorithms for maximizing submodular set functions, in: Optimization Techniques, 1978, pp. 234–243.
[31] M. Mitrovic, M. Bun, A. Krause, A. Karbasi, Differentially private submodular maximization: data summarization in disguise, in: ICML, 2017, pp. 2478–2487.
[32] W. Lu, W. Chen, L.V.S. Lakshmanan, From competition to complementarity: comparative influence diffusion and maximization, Proc. VLDB Endow. 9 (2015) 60–71.
[33] C. Manning, P. Raghavan, M. Schütze, Introduction to Information Retrieval, 2008.
[34] S. Feng, X. Chen, G. Cong, Y. Zeng, Y.M. Chee, Y. Xiang, Influence maximization with novelty decay in social networks, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14, 2014, pp. 37–43.
[35] T. Horel, Y. Singer, Maximization of approximately submodular functions, in: NIPS, 2016, pp. 3045–3053.
[36] Y. Yang, J. Jia, B. Wu, J. Tang, Social role-aware emotion contagion in image social networks, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, 2016, pp. 65–71.
[37] L. Backstrom, D. Huttenlocher, J. Kleinberg, X. Lan, Group formation in large social networks: membership, growth, and evolution, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, 2006, pp. 44–54.
[38] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincón, X. Sun, Y. Wang, W. Wei, Y. Yuan, Influence maximization in social networks when negative opinions may emerge and propagate, in: SDM, 2011, pp. 379–390.
[39] A. Goyal, W. Lu, L.V.S. Lakshmanan, Simpath: an efficient algorithm for influence maximization under the linear threshold model, in: ICDM, 2011, pp. 211–220.
[40] Y.T. Wen, W. Peng, H. Shuai, Maximizing social influence on target users, in: PAKDD, 2018, pp. 701–712.
[41] F. Li, C. Li, M. Shan, Labeled influence maximization in social networks for target marketing, in: PASSAT/SocialCom 2011, 2011, pp. 560–563.
[42] C. Song, W. Hsu, M.L. Lee, Targeted influence maximization in social networks, in: CIKM, 2016, pp. 1683–1692.
[43] A. Khan, B. Zehnder, D. Kossmann, Revenue maximization by viral marketing: a social network host's perspective, in: ICDE, 2016, pp. 37–48.
[44] R. Abebe, L.A. Adamic, J.M. Kleinberg, Mitigating overexposure in viral marketing, in: AAAI, 2018, pp. 241–248.
[45] S. Gershtein, T. Milo, B. Youngmann, G. Zeevi, IM balanced: influence maximization under balance constraints, in: CIKM, CIKM '18, 2018, pp. 1919–1922.
[46] A. Tsang, B. Wilder, E. Rice, M. Tambe, Y. Zick, Group-fairness in influence maximization, in: IJCAI, 2019, pp. 5997–6005.
[47] B. Liu, G. Cong, D. Xu, Y. Zeng, Time constrained influence maximization in social networks, in: ICDM, 2012, pp. 439–448.