Journal of Machine Learning Research 21 (2020) 1-36 Submitted 3/18; Revised 10/19; Published 02/20

Generalized Nonbacktracking Bounds on the Influence in Independent Cascade Models

Emmanuel Abbe EABBE@PRINCETON.EDU
Program in Applied and Computational Mathematics and Department of Electrical Engineering
Princeton University
Princeton, NJ 08540, USA

Sanjeev Kulkarni KULKARNI@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Princeton, NJ 08540, USA

Eun Jee Lee EJLEE.MAIL@GMAIL.COM
Program in Applied and Computational Mathematics
Princeton University
Princeton, NJ 08540, USA

Editor: Bert Huang

Abstract

This paper develops deterministic upper and lower bounds on the influence measure in a network, more precisely, the expected number of nodes that a seed set can influence in the independent cascade model. In particular, our bounds exploit r-nonbacktracking walks and Fortuin—Kasteleyn—Ginibre (FKG) type inequalities, and are computed by message passing algorithms. Further, we provide parameterized versions of the bounds that control the trade-off between efficiency and accuracy. Finally, the tightness of the bounds is illustrated on various network models.

Keywords: Influence Estimation, Nonbacktracking Walk, Message Passing, Social Networks, Independent Cascade Model

1. Introduction

Social interaction is a building block of the society. Through interacting with each other, people exchange their information, ideas, opinions, etc. and *influence* one another. Influence propagation is a study which investigates how influence spread given a social network from initially influenced nodes, called *seeds*, in the network. Studying how influence spreads in a network allows us to answer many important questions in a broad range of fields, such as viral marketing (Leskovec et al., 2007), sociology (Granovetter, 1978; Lopez-Pintado and Watts, 2008; Watts, 2002), communication (Khelil et al., 2002), epidemiology (Shulgin et al., 1998), and social network analysis (Yang and Counts, 2010).

With the advent of the Internet and the availability of large data on social networks, influence propagation is becoming the center of interests in machine learning and data mining community. The community has proposed several different mathematical models to abstractize influence propagation in the real world, and one of the most widely adopted models is the independent cascade model (ICM). In this model, a network is defined as a directed graph $G = (V, E)$ where every edge is given transmission probability. Each edge decides its state, live or not, with a Bernoulli trial with the success rate equals to its transmission probability. Then, the influence propagation starts from the initially influenced nodes, called the seeds, and spreads along the live edges. A node is influenced if it is reachable from the seeds with the live edges.

While various important information can be extracted by studying ICM, the most fundamental and essential problem in influence propagation is to find *the influence*, the expected number of influenced nodes given a network and a set of initially influenced nodes. Unfortunately, computing the influence of a network in ICM is proven to be #P hard (Wang et al., 2012) and thus many previous works have used Monte Carlo (MC) simulations to estimate the true influence (Kempe et al., 2003; Chen et al., 2009). Despite its simplicity, approximating the influence via MC simulations can be computationally expensive. MC approximation requires $O(|V|^2)$ trials to concentrate and each trial requires $O(|V| + |E|)$ computation. Considering that modern real-world networks often have multiple millions of nodes, this amount of computation quickly becomes unreasonable.

To overcome the limitations of Monte Carlo simulations, many researchers have been taking both algorithmic and theoretical approaches to approximate the influence of given seeds in a network. Chen and Teng (Chen and Teng, 2017) provided a probabilistic guarantee on estimating the influence of a single seed with a relative error bound with the expected running time $O(\ell(|V| + |E|)|V| \log |V|/\varepsilon^2)$, such that with probability $1 - 1/n^\ell$, for all node v, the computed influence of v has relative error at most ε. Draief et al., (Draief et al., 2006) introduced an upper bound for the influence of any set of seeds by using the spectral radius of the adjacency matrix. Tighter upper bounds were later suggested in (Lemonnier et al., 2014) which relate the ratio of influenced nodes in a network to the spectral radius of the so-called *Hazard* matrix. Further, improved upper bounds which account for *sensitive* edges were introduced in (Lee et al., 2016). In contrast, there has been little work on finding a tight lower bound for the influence. An exception is a work by Khim et al. (Khim et al., 2016), where the lower bound is obtained by only considering the influence through the paths whose length is no more than a parameter m.

In this article, we extend the results in the paper (Abbe et al., 2017) and propose upper and lower bounds on the influence using nonbacktracking walks and Fortuin—Kasteleyn—Ginibre (FKG) type inequalities. FKG inequality is useful to prove correlation between non-decreasing functions, which is instrumental in influence propagation since the probability that a node is influenced is non-decreasing with respect to the partial order of random variables describing the states of the edges. The bounds can be efficiently obtained by message passing algorithms. This shows that nonbacktracking walks can also impact influence propagation, making another case for the use of nonbacktracking walks in graph and network problems as in (Krzakala et al., 2013; Karrer et al., 2014; Bordenave et al., 2015; Abbe and Sandon, 2015), discussed later in the paper. Next, we show that the proposed upper bound is monotone and submodular. This property allows us to use the upper bound, rather than some approximations, when greedily finding a set of seeds that maximizes the influence. Further, we provide parametrized versions of the bounds that can adjust the trade-off between the efficiency and the accuracy of the bounds.

2. Background

We introduce here the independent cascade model and provide background for the main results.

Definition 1 (Independent Cascade Model). *Consider a directed graph $G = (V, E)$ with $|V| = n$, a transmission probability matrix $B \in [0,1]^{n \times n}$, and a seed set $S_0 \subseteq V$. For any $u \in V$, let $N^+(u)$ be the set of out-neighbors of node u. The independent cascade model $IC(G, B)$ sequentially generates the influenced set $S_t \subseteq V$ starting from the seed set S_0 for each step $t \geq 1$ as follows. At the beginning of step t, S_t is initialized to be an empty set. Then, each node $u \in S_{t-1}$ attempts to influence $v \in N^+(u) \backslash \cup_{i=0}^{t-1} S_i$ with probability B_{uv}, i.e., node u influences its uninfluenced out-neighbor v with probability B_{uv}. If v is influenced as a result of the attempts, add v to S_t. At the end of step t, S_t contains all nodes that are influenced during step t. The process stops at T if $S_T = \emptyset$ at the end of the step $t = T$. The set of the influenced nodes at the end of propagation is defined as $S = \cup_{i=0}^{T-1} S_t$.*

We often refer an edge (u, v) being *live* if node u influences node v. The independent cascade model is equivalent to the live-edge graph model, where the influence happens at once, rather than sequentially. The live-edge graph model first decides the state of every edge with a Bernoulli trial, i.e., edge (u, v) becomes live independently with probability B_{uv}. Then, the set of the influenced nodes is defined as the nodes that are reachable from at least one of the seeds by the live edges.

Definition 2 (Influence). *The expected number of nodes that are influenced at the end of the propagation process is called the* influence *(rather than the expected influence, with a slight abuse of terminology) of a seed set S_0 on $IC(G, B)$, and is defined as*

$$\sigma(S_0) \quad = \quad \sum_{v \in V} \mathbb{P}(v \text{ is influenced}).$$

It is shown in (Wang et al., 2012) that computing the influence $\sigma(S_0)$ in the independent cascade model $IC(G, B)$ is #P-hard, even with a single seed, i.e., $|S_0| = 1$.

Next, we define nonbacktracking (NB) walks on a directed graph. Nonbacktracking walks have been used for studying the characteristics of networks. To the best of our knowledge, the use of NB walks in the context of epidemics was first introduced in the paper of Karrer et al. (Karrer and Newman, 2010) and later applied to percolation in (Karrer et al., 2014). In particular, Karrer et al. reformulate the spread of influence as a message passing process and demonstrate how the resulting equations can be used to calculate an upper bound on the number of nodes that are susceptible at a given time. As we shall see, we take a different approach to the use of the NB walks, which focuses on the effective contribution of a node in influencing another node and accumulates such contributions to obtain upper and lower bounds. More recently, nonbacktracking walks are used for community detection (Krzakala et al., 2013; Bordenave et al., 2015; Abbe and Sandon, 2015).

Definition 3 (Nonbacktracking Walk). *Let $G = (V, E)$ be a directed graph. A nonbacktracking walk of length k is defined as $w^{(k)} = (v_0, v_1, \ldots, v_k)$, where $v_i \in V$ and $(v_{i-1}, v_i) \in E$ for all $i \in [k]$, and $v_{i-1} \neq v_{i+1}$ for all $i \in [k-1]$.*

We next recall a key inequality introduced by Fortuin et. al (Fortuin et al., 1971).

Theorem 1 (FKG Inequality). *Let $(\Gamma, \prec)$ be a distributive lattice, where Γ is a finite partially ordered set, ordered by $\prec$, and let μ be a positive measure on Γ satisfying the following condition: for all $x, y \in \Gamma$,*

$$\mu(x \wedge y)\mu(x \vee y) \quad \geq \quad \mu(x)\mu(y),$$

where $x \wedge y = \max\{z \in \Gamma : z \preceq x, z \preceq y\}$ and $x \vee y = \min\{z \in \Gamma : y \preceq z, y \preceq z\}$. Let f and g be both increasing (or both decreasing) functions on Γ. Then,

$$(\sum_{x\in\Gamma} \mu(x))(\sum_{x\in\Gamma} f(x)g(x)\mu(x)) \quad \geq \quad (\sum_{x\in\Gamma} f(x)\mu(x))(\sum_{x\in\Gamma} g(x)\mu(x)).$$

3. Nonbacktracking Upper Bounds

In this section, we present three types of nonbacktracking upper bounds on the influence in the independent cascade model and explain the motivations and intuitions of the bounds. The bounds use nonbacktracking walks and FKG inequalities and are computed efficiently by message passing algorithms. The proposed upper bounds listed below have varying efficiency and accuracy trade-offs.

- Nonbacktracking upper bound (NB-UB)
- Tunable nonbacktracking upper bound (tNB-UB)
- r-nonbacktracking upper bound (rNB-UB)

In particular, the nonbacktracking upper bound on a network based on a graph $G(V, E)$ runs in $O(|V|^2 + |V||E|)$, whereas Monte Carlo simulation would require $O(|V|^3 + |V|^2|E|)$ computations without knowing the variance of the influence, which is harder to estimate than the influence. The reason for the large computational complexity of MC is that in order to ensure that the standard error of the estimation does not grow with respect to $|V|$, MC requires $O(|V|^2)$ computations. Hence, for large networks, where MC may not be feasible, our algorithms can still provide bounds on the influence. Furthermore, the proposed upper bound is monotone and submodular, so that it can be used in place of MC approximation in greedy algorithm for influence maximization problem. Details of the greedy algorithm in influence maximization problem can be found in (Kempe et al., 2003).

The tunable nonbacktracking upper bound better accounts for the *sensitive* (as defined in (Lee et al., 2016)) edges near the seeds. However, for large values of t, it may result in exponential computational complexity for dense networks.

The r-nonbacktracking upper bound generalizes the nonbacktracking upper bound by considering r-nonbacktracking walks, i.e., avoiding cycles of length r rather than just backtracking, and runs in $O((|V| + |E|)|V|^{r-1})$.

This section is organized as follows. First, sections 3.1 through 3.3 present the definition of each proposed upper bound followed by an efficient algorithm to compute the bound and its computational complexity. Next, section 3.4 shows how each upper bound is computed on a small example network and compares the bounds with the influence.

3.1. Nonbacktracking Upper Bound (NB-UB)

We start by defining the following terms for the independent cascade model $IC(G, B)$, where $G = (V, E)$ and $|V| = n$.

Definition 4. *For any $v \in V$, we define the set of in-neighbors $N^-(v) = \{u \in V : (u, v) \in E\}$ and the set of out-neighbors $N^+(v) = \{u \in V : (v, u) \in E\}$.*

Definition 5. *For any $v \in V$ and $l \in [n-1]$, the set $P_l(S_0 \to v)$ is defined as the set of all paths with length l from any seed $s \in S_0$ to v. We call a path P is* live *iff every edge in P is live. For $l = 0$, we define $P_0(S_0 \to v)$ as the set (of size one) of the zero-length path containing node v and assume the path $P \in P_0(S_0 \to v)$ is live iff $v \in S_0$.*

Definition 6. *For any $v \in V$ and $l \in \{0, \dots, n-1\}$, we define*

$$\begin{aligned} p(v) &= \mathbb{P}(v \textit{ is influenced}) \\ p_l(v) &= \mathbb{P}(\cup_{P \in P_l(S_0 \to v)}\{P \textit{ is live}\}) \\ p_l(u \to v) &= \mathbb{P}(\cup_{P \in P_l(S_0 \to u), P \not\ni v}\{P \textit{ is live and edge } (u,v) \textit{ is live}\}) \end{aligned}$$

In other words, $p_l(v)$ is the probability that node v is influenced by live paths of length l, i.e., there exists a live path of length l from a seed to v, and $p_l(u \to v)$ is the probability that v is influenced by node u with live paths of length $l+1$, i.e., there exists a live path of length $l+1$ from a seed to v that ends with edge (u, v).

Lemma 1. *For any $v \in V$,*

$$p(v) \leq 1 - \prod_{l=0}^{n-1}(1 - p_l(v)). \tag{1}$$

For any $v \in V$ and $l \in [n-1]$,

$$p_l(v) \leq 1 - \prod_{u \in N^-(v)}(1 - p_{l-1}(u \to v)). \tag{2}$$

Lemma 1, which can be proved by FKG inequalities, suggests that given $p_{l-1}(u \to v)$, we may compute an upper bound on the influence. Ideally, $p_{l-1}(u \to v)$ can be computed by considering all paths that end with (u, v) having length l. However, this results in exponential complexity $O(n^l)$, as l goes up to $n-1$. Thus, in Definition 7, we present an efficient way to compute an upper bound $\mathrm{UB}_{l-1}(u \to v)$ on $p_{l-1}(u \to v)$, which in turns gives an upper bound $\mathrm{UB}_l(v)$ on $p_l(v)$, with the following recursion formula.

Definition 7. *For all $l \in \{0, \dots, n-1\}$ and $u, v \in V$ such that $(u, v) \in E$, $\mathrm{UB}_l(u) \in [0, 1]$ and $\mathrm{UB}_l(u \to v) \in [0, 1]$ are defined recursively as follows.*
Initial condition: *For every $s \in S_0$, $s^+ \in N^+(s)$, $u \in V \setminus S_0$, and $v \in N^+(u)$,*

$$\mathrm{UB}_0(s) = 1,\ \mathrm{UB}_0(s \to s^+) = B_{ss^+} \tag{3}$$

$$\mathrm{UB}_0(u) = 0,\ \mathrm{UB}_0(u \to v) = 0. \tag{4}$$

Recursion: *For every* $l \in [n-1]$, $s \in S_0$, $s^+ \in N^+(s)$, $s^- \in N^-(s)$, $u \in V \setminus S_0$, *and* $v \in N^+(u) \setminus S_0$,

$$\mathrm{UB}_l(s) = 0, \mathrm{UB}_l(s \to s^+) = 0, \mathrm{UB}_l(s^- \to s) = 0 \tag{5}$$

$$\mathrm{UB}_l(u) = 1 - \prod_{w \in N^-(u)} (1 - \mathrm{UB}_{l-1}(w \to u)) \tag{6}$$

$$\mathrm{UB}_l(u \to v) = \begin{cases} B_{uv}(1 - \frac{1-\mathrm{UB}_l(u)}{1-\mathrm{UB}_{l-1}(v \to u)}), & \textit{if } v \in N^-(u) \\ B_{uv}\mathrm{UB}_l(u), & \textit{otherwise.} \end{cases} \tag{7}$$

Equation (5) follows from that for any seed node $s \in S_0$ and for all $l > 0$, the probabilities $p_l(s) = 0$, $p_l(s \to s^+) = 0$, and $p_l(s^- \to s) = 0$. A naive way to compute $\mathrm{UB}_l(u \to v)$ is $\mathrm{UB}_l(u \to v) = B_{uv}\mathrm{UB}_{l-1}(u)$, but this results in an extremely loose bound due to the backtracking. For a tighter bound, we use nonbacktracking in Equation (7), i.e., when computing $\mathrm{UB}_l(u \to v)$, we ignore the contribution of $\mathrm{UB}_{l-1}(v \to u)$.

Finally, an upper bound on the probability that node v is influence at the end of propagation is computed by $\mathrm{UB}(v) = (1 - \prod_{l=0}^{n-1}(1 - \mathrm{UB}_l(v)))$. The following theorem shows that the summation of $\mathrm{UB}(v)$ for $v \in V$ results in an upper bound on the influence.

Theorem 2 (Nonbacktracking Upper Bound (NB-UB)). *For any independent cascade model $IC(G, B)$ and a seed set $S_0 \subseteq V$,*

$$\sigma(S_0) \quad \leq \quad \sum_{v \in V} (1 - \prod_{l=0}^{n-1} (1 - \mathrm{UB}_l(v))) =: \sigma^+(S_0), \tag{8}$$

where $\mathrm{UB}_l(v)$ *is obtained recursively as in Definition 7.*

The Nonbacktracking Upper Bound is monotone and submodular. Therefore, it can be used in place of approximations on the influence in various greedy algorithms for the influence maximization problem.

Theorem 3. *For any independent cascade model $IC(G, B)$, the nonbacktracking upper bound $\sigma^+ : 2^V \to [0, |V|]$ is monotone and submodular.*

Next, we present Nonbacktracking Upper Bound (NB-UB) algorithm which computes $\mathrm{UB}_l(v)$ and $\mathrm{UB}_l(u \to v)$ by message passing. At the l-th iteration, the variables in NB-UB have the following meanings.

· S_l is the set of nodes that are considered in computation at the l-th iteration.

· $\mathrm{M}_{\mathrm{curr}}(v) = \{(u, \mathrm{UB}_{l-1}(u \to v)) : u \text{ is an in-neighbor of } v \text{ and } u \in S_{l-1}\}$ contains the current messages to node v and their sources, where the source is an in-neighbor u of v that is in S_{l-1} and the message is the upper bound $\mathrm{UB}_{l-1}(u \to v)$.

· $\mathrm{MSrc}(v) = \{u : u \text{ is a in-neighbor of } v \text{ and } u \in S_{l-1}\}$ is the set of in-neighbors of v in S_{l-1}.

· $\mathrm{M}_{\mathrm{curr}}(v)[u] = \mathrm{UB}_{l-1}(u \to v)$ is the current message from u to v.

· $\mathrm{M}_{\mathrm{next}}(v) = \{(u, \mathrm{UB}_l(u \to v)) : u \text{ is an in-neighbor of } v \text{ and } u \in S_l\}$ contains the set of messages to node v and their sources for the algorithm to use next.

Algorithm 1 Nonbacktracking Upper Bound (NB-UB)

```
Initialize: $\mathrm{UB}_l(v) = 0$ for all $0 \le l \le n-1$ and $v \in V$
Initialize: Insert $(s, 1)$ to $\mathrm{M}_{\mathrm{next}}(s)$ for all $s \in S_0$
for $l = 0$ to $n-1$ do
    for $u \in S_l$ do
        $\mathrm{M}_{\mathrm{curr}}(u) = \mathrm{M}_{\mathrm{next}}(u)$
        Clear $\mathrm{M}_{\mathrm{next}}(u)$
        $\mathrm{UB}_l(u) = \texttt{ProcessIncomingMsg}_{\texttt{UB}}(\mathrm{M}_{\mathrm{curr}}(u))$
    for $u \in S_l$ do
        for $v \in N^+(u) \setminus S_0$ do
            $S_{l+1}$.insert$(v)$
            if $v \in \mathrm{MSrc}(u)$ then
                $\mathrm{UB}_l(u \to v) = \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\mathrm{M}_{\mathrm{curr}}(u)[v], \mathrm{UB}_l(u), B_{uv})$
                $\mathrm{M}_{\mathrm{next}}(v)$.insert$((u, \mathrm{UB}_l(u \to v)))$.
            else
                $\mathrm{UB}_l(u \to v) = \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(0, \mathrm{UB}_l(u), B_{uv})$
                $\mathrm{M}_{\mathrm{next}}(v)$.insert$((u, \mathrm{UB}_l(u \to v)))$.
Output: $\mathrm{UB}_l(u)$ for all $l$, $u$
```

At the beginning, every seed node $s \in S_0$ is initialized such that $\mathrm{M}_{\mathrm{curr}}(s) = \{(s, 1)\}$ in order to satisfy the initial condition, $\mathrm{UB}_0(s) = 1$. For l-th iteration, every node u in S_l is processed as follows. First, $\texttt{ProcessIncomingMsg}_{\texttt{UB}}(\mathrm{M}_{\mathrm{curr}}(u))$ computes $\mathrm{UB}_l(u)$ as in Equation (6). Second, u passes a message to its neighbor $v \in N^+(u) \setminus S_0$ along the edge (u, v), and v stores (inserts) the message in $\mathrm{M}_{\mathrm{next}}(v)$ in preparation for the next iteration. The message contains 1) the source of the message, u, and 2) the upper bound, $\mathrm{UB}_l(u \to v)$, which is computed as in Equation (7), by the function $\texttt{GenerateOutgoingMsg}_{\texttt{UB}}$. Finally, the algorithm outputs $\mathrm{UB}_l(u)$ for all $u \in V$ and $l \in \{0, \ldots, n-1\}$, and the upper bound $\sigma^+(S_0)$ is computed by Equation (8).

Computational complexity: Notice that for each iteration $l \in \{0, \ldots, n-1\}$, the algorithm accesses at most n nodes, and for each node v, the functions $\texttt{ProcessIncomingMsg}_{\texttt{UB}}$ and $\texttt{GenerateOutgoingMsg}_{\texttt{UB}}$ are computed in $O(\deg(v))$ and $O(1)$, respectively. Therefore, the worst case computational complexity is $O(|V|^2 + |V||E|)$.

3.2. Tunable Nonbacktracking Upper Bound (tNB-UB)

One key observation on NB-UB is that when computing the nonbacktracking upper bound $\mathrm{UB}_l(v)$ recursively on l, the computations with small l's are more important than the ones with larger l's since $\mathrm{UB}_l(v)$ is a function of $\mathrm{UB}_{l-1}(u)$ for some $u \in V$. In other words, computing loose bounds $\mathrm{UB}_l(v)$ for smaller l hurts the accuracy of the bound on the influence more than computing loose bounds for larger l, because the effect of loose $\mathrm{UB}_l(v)$ with smaller l builds up more in the final bound $\sigma^+(S_0)$. Therefore, we propose a parameterized upper bound tNB-UB that computes $p_l(v)$ up to $l \le t$ precisely.

As shown in Algorithm 2, tNB-UB inputs the parameter t, which indicates the maximum length of the paths that the algorithm finds to compute the exact, rather than the upper bound on, probability of influence. That is, the algorithm computes $p_{\le t}(u)$ that node u is influenced by live paths whose length is less than or equal to t.

$$p_{\leq t}(u) \quad = \quad \mathbb{P}(\cup_{P \in \{\cup_{i=0}^{t} P_i(S_0 \to u)\}} \{P \text{ is live}\}). \tag{9}$$

Then, we start (non-parameterized) NB-UB algorithm from $l = t+1$ with the new initial conditions: for all $u \in V$ and $v \in N^+(u)$,

$$\mathrm{UB}_t(u) = p_{\leq t}(u) \tag{10}$$

$$\mathrm{UB}_t(u \to v) = p_t(u \to v) \tag{11}$$

Finally, the upper bound tNB-UB is computed as $\sum_{v \in V}(1 - \prod_{l=t}^{n-1}(1 - \mathrm{UB}_l(v)))$.

Algorithm 2 Tunable nonbacktracking upper bound (tNB-UB)

```
parameter: non-negative integer $t \leq n-1$
Initialize: $\mathrm{UB}_t(v) = 0$ for all $t \leq l \leq n-1$ and $v \in V$
for $u \in V$ do
    $\mathrm{UB}_t(u) = p_{\leq t}(u)$
    for $v \in N^+(u) \setminus S_0$ do
        if $p_t(u \to v) > 0$ then
            $S_{t+1}$.insert($v$)
            $\mathrm{M}_{\text{next}}(v)$.insert($u, p_t(u \to v)$)
for $l = t+1$ to $n-1$ do
    for $u \in S_l$ do
        $\mathrm{M}_{\text{curr}}(u) = \mathrm{M}_{\text{next}}(u)$
        Clear $\mathrm{M}_{\text{next}}(u)$
        $\mathrm{UB}_l(u) = \texttt{ProcessIncomingMsg}_{\texttt{UB}}(\mathrm{M}_{\text{curr}}(u))$
    for $u \in S_l$ do
        for $v \in N^+(u) \setminus S_0$ do
            $S_{l+1}$.insert($v$)
            if $v \in \mathrm{M}_{\text{curr}}(u)$ then
                $\mathrm{UB}_l(u \to v) = \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\mathrm{M}_{\text{curr}}(u)[v], \mathrm{UB}_l(u), B_{uv})$
                $\mathrm{M}_{\text{next}}(v)$.insert(($u, \mathrm{UB}_l(u \to v)$)).
            else
                $\mathrm{UB}_l(u \to v) = \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(0, \mathrm{UB}_l(u), B_{uv})$
                $\mathrm{M}_{\text{next}}(v)$.insert(($u, \mathrm{UB}_l(u \to v)$)).
Output: $\mathrm{UB}_l(u)$ for all $l = \{t, t+1, \ldots, n-1\}$, $u \in V$
```

For larger values of t, the algorithm results in tighter upper bounds, while the computational complexity may increase exponentially for dense networks. Thus, this method is most applicable in sparse networks, where the degree of each node is bounded.

3.3. rNonbacktracking Upper Bound (rNB-UB)

In this section, we generalize the nonbacktracking upper bound by considering r-nonbacktracking walks, i.e., avoiding cycles of length r rather than just backtracks. The r-nonbacktracking upper bound is not necessarily a tNB-UB which computes $p_{\leq t}(v)$ for all $v \in V$ exactly, even in case of $r = t$.

Definition 8. *Consider a directed graph $G = (V, E)$. For any integer $r \geq 1$ and node $u \in V$, let $P_{r-1}(\cdot \to u)$ be the set of all paths of length $r-1$ ending at node u and $P_{r-1}(u \to \cdot)$ be the set of all paths of length $r-1$ starting from node u. To simplify the notation, we use $P_{r-1}(S \to \cdot)$ for a set of nodes S to denote the set of all paths of length $r-1$ starting from any one of the nodes in S.*

For two same length paths $P = (v_1, \ldots, v_r)$ and $Q = (u_1, \ldots, u_r)$, we use the notation $P \sim Q$ when $v_{i+1} = u_i$ for all $i \in [r-1]$ and $v_1 \neq u_r$. In other words, $P \sim Q$ iff $(v_1, v_2 = u_1, \ldots, v_r = u_{r-1}, u_r)$ is a path of length r.

For two paths $P = (v_0, \ldots, v_r)$ and $Q = (u_1, \ldots, u_r)$, we use the notation $Q \sqsubset P$ when $v_i = u_i$ for all $i \in [r]$. That is, Q is a subpath of P which is obtained by removing the first node v_0 from P. Also, let $P - Q = (v_0, v_1)$ be the edge in path P but not in path Q.

Recall that NB-UB computes the upper bounds $\mathrm{UB}_l(u)$ on $p_l(u)$ and $\mathrm{UB}_l(u \to v)$ on $p_l(u \to v)$ recursively. Similarly, rNB-UB computes the upper bounds $\mathrm{UB}_l(u)$ and $\mathrm{UB}_l(u, P)$ recursively, as defined in Definition 9. $\mathrm{UB}_l(u, P)$ is analogous to $\mathrm{UB}_l(u \to v)$, since $\mathrm{UB}_l(u, P)$ provides the effective influence of u along the path P whereas $\mathrm{UB}_l(u \to v)$ gives the effective influence of u along the edge (u, v).

Definition 9 (r-Nonbacktracking Upper Bound (rNB-UB))**.** *For all $l \in \{0, \ldots, n-1\}$, $u \in V$ and $P \in P_{r-1}(u \to \cdot)$, we define $\mathrm{UB}_l(u) \in [0, 1]$ and $\mathrm{UB}_l(u, P) \in [0, 1]$ as follows, using function f defined later in Definition 10.*
Initial condition: *For every $l \leq r-1$ and $u \in V$,*

$$\mathrm{UB}_l(u) = f(\cup_{P \in P_l(S_0 \to u)}(P, 1)). \tag{12}$$

For every $P \in P_{r-1}(s \to \cdot)$ and $s \in S_0$,

$$\mathrm{UB}_{r-1}(s, P) = 1. \tag{13}$$

Recursion: *For every $l \in \{r, \ldots, n-1\}$, $s \in S_0$, $u, x \in V \setminus S_0$, $v \in N^+(u) \setminus S_0$, and $Q \in P_{r-1}(v \to \cdot)$,*

$$\mathrm{UB}_l(s) = 0, \mathrm{UB}_l(s, P) = 0 \text{ for any path } P \tag{14}$$

$$\mathrm{UB}_l(v, Q) = 1 - \prod_{P \sim Q} (1 - B_{uv} \mathrm{UB}_{l-1}(u, P)) \tag{15}$$

$$\mathrm{UB}_l(x) = f(\cup_{P \in \{P' \in P_{r-1}(y \to x) : \forall y \in V \setminus S_0\}}(P, \mathrm{UB}_l(y, P))). \tag{16}$$

Equation (12) computes the upper bounds on the probability that node u is influenced by at least one length l path, for each $l \leq r-1$. To provide the initial values, we set $\mathrm{UB}_{r-1}(s, P) = 1$ indicating that the probability of a seed node s being influenced equals to 1. In the recursion steps, Equation (15) computes $\mathrm{UB}_l(v, Q)$ the upper bound on the probability that the starting node v of path Q is influenced. Note that the upper bound $\mathrm{UB}_l(v, Q) \neq \mathrm{UB}_{l-(r-1)}(v)$, since $\mathrm{UB}_l(v, Q)$ is the upper bound on the probability that node v is influenced by a path of length $l-(r-1)$ that does not contain any node in Q, whereas $\mathrm{UB}_{l-(r-1)}(v)$ is the upper bound on the probability that node v is influenced by any path of length $l-(r-1)$. Note that given path Q, the starting node v is deterministic. We use $\mathrm{UB}_l(v, Q)$ instead of $\mathrm{UB}_l(Q)$ in order to emphasize that the bound is for node v of path Q.

Finally, Equation (16) computes $\mathrm{UB}_l(x)$, the upper bound on the probability that node x is influenced by at least one r-nonbacktracking walks of length l by the function f. In the following, we define the function f.

Definition 10. *Consider a directed graph $G = (V, E)$ and a non-negative integer r. The function f takes input $P^{(r)} = \{P_1, \ldots, P_m\}$ and $x^{(r)} = \{x^{(r)}_{P_1}, \ldots, x^{(r)}_{P_m}\}$, where $P^{(r)}$ is a set of length r paths in G ending at the same node, say v, and $x^{(r)}$ is the set of real numbers in $[0,1]$ associated with the paths. When the input is the empty sets, $f(\emptyset, \emptyset) = 0$. Otherwise, $f(P^{(r)}, x^{(r)})$ computes the output by the following recursion. For every $i \in [r]$ from r to 1,*

$$P^{(i-1)} = \{P' : P' \sqsubset P, P \in P^{(i)}\} \tag{17}$$

and for every $P' \in P^{(i-1)}$,

$$x^{(i-1)}_{P'} = 1 - \prod_{P \in \{P : P' \sqsubset P, P \in P^{(i)}\}} (1 - B_e x^{(i)}_P) \tag{18}$$

where $e = P - P'$ and B_e is the transmission probability of edge e. By the definition of $P^{(r)}$, the set $P^{(0)}$ has only one element which is the path $P_0 = (v)$ of length 0 containing node v. Therefore, at the end of the recursion, $i = 1$, it computes the final output $x^{(0)}_{P_0} \in [0,1]$.

We next illustrates how to compute the function f in a small example. Consider a directed graph G where all edges have the same transmission probability p. Let the function f take the inputs $P^{(3)} = \{(a,b,c,d), (a,f,c,d), (e,f,c,d)\}$ and $f^{(3)} = \{1,1,1\}$. Then, $P^{(2)} = \{(b,c,d), (f,c,d)\}$ since $(b,c,d) \sqsubset (a,b,c,d)$ and $(f,c,d) \sqsubset (a,f,c,d), (e,f,c,d)$. Likewise, $P^{(1)} = \{(c,d)\}$ as $(c,d) \sqsubset (b,c,d), (f,c,d)$ and $P^{(0)} = \{(d)\}$ as $(d) \sqsubset (c,d)$. In figure 1, it shows the sets $P^{(3)}, \ldots, P^{(0)}$ in a radix tree structure. With the radix tree, we can easily find the set $\{P : P' \sqsubset P, P \in P^{(i)}\}$ by accessing the descendants of the path P'. For example, path $P' = (f,c,d)$ in the radix tree has two descendants a and e, where each corresponds to the path (a,f,c,d) and (e,f,c,d). This illustrates the set $\{P : (f,c,d) \sqsubset P, P \in P^{(i)}\} = \{(a,f,c,d), (e,f,c,d)\}$.

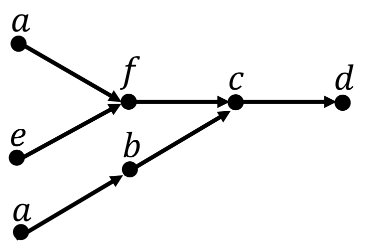


Figure 1: An example of radix tree structure

The function f computes the output recursively from $r = 3$ to 1.

$$\begin{aligned}
f^{(2)}_{(b,c,d)} &= 1 - \prod_{P \in \{(a,b,c,d)\}} (1 - pf^{(3)}_P) = p \\
f^{(2)}_{(f,c,d)} &= 1 - \prod_{P \in \{(a,f,c,d),(e,f,c,d)\}} (1 - pf^{(3)}_P) = 2p - p^2 \\
f^{(1)}_{(c,d)} &= 1 - \prod_{P \in \{(b,c,d),(f,c,d)\}} (1 - pf^{(2)}_P) = 3p^2 - p^3 - 2p^4 + p^5 \\
f^{(0)}_{(d)} &= 1 - \prod_{P \in \{(c,d)\}} (1 - pf^{(1)}_P) = 3p^3 - p^4 - 2p^5 + p^6.
\end{aligned}$$

We use the function f to compute the upper bound $\mathrm{UB}_l(v)$ in each step, instead of using Lemma 1, Equation (2) as in NB-UB. This results in a tighter upper bound since Equation (2) bounds the union of events that node v is influenced by at least one of the walks by treating each walk independent whereas f accounts for the dependencies among the walks. Also, this function ensures that r-nonbacktracking upper bound is smaller than or equal to r'-nonbacktracking upper bound when $r > r'$.

Theorem 4. *For any independent cascade model $IC(G, B)$, a seed set $S_0 \subseteq V$ and an integer $r \geq 2$,*

$$\sigma(S_0) \leq \sum_{v \in V} (1 - \prod_{l=0}^{n-1} (1 - \mathrm{UB}_l(v))) =: \sigma^+_r(S_0) \leq \sigma^+(S_0), \tag{19}$$

where $\mathrm{UB}_l(v)$ *is obtained recursively as in Definition 9.*

Next, we present the r-Nonbacktracking Upper Bound (rNB-UB) algorithm which computes $\mathrm{UB}_l(v)$ and $\mathrm{UB}_l(v, P)$ by message passing. At the l-th iteration, the variables in rNB-UB have the following meanings.

- S_l represents the set of nodes that are used in the computations at the l-th iteration.
- $\mathrm{M}_{\mathrm{curr}}(u) = \{(P, \mathrm{UB}_{l-1}(\cdot, P)) : P \in P_{r-1}(\cdot \to u) \text{ and } u \in S_{l-1}\}$ is the set of messages that node u received in the previous step. The message contains a $(r-1)$-length path ending at node u and the bound associated with the path.
- $\mathrm{M}_{\mathrm{next}}(u) = \{(P, \mathrm{UB}_l(\cdot, P)) : P \in P_{r-1}(\cdot \to u) \text{ and } u \in S_l\}$ is the set messages that node u receive in the current iteration. The message contains a $(r-1)$-length path ending at u and the bound associated with the path.

At the beginning, the first $r - 1$ upper bounds for each node are computed as well as $\mathrm{UB}_{r-1}(s, P)$ for all $s \in S_0$ and $P \in P_{r-1}(S_0 \to \cdot)$ to satisfy the initial condition in Definition 9. For each l-th iteration, the algorithm processes every node u in S_l as follows. First, $f(\mathrm{M}_{\mathrm{curr}}(u))$ computes the upper bound $\mathrm{UB}_l(u)$ as in Equation (16) defined in Definition 10. Second, u transmits a message to its neighbor $v \in N^+(u) \setminus S_0$, and v stores (inserts) the message in $\mathrm{M}_{\mathrm{next}}(v)$ for the next iteration. The message contains 1) the path Q of length $r - 1$ ending at node v which satisfies $P \sim Q$, and 2) $\mathrm{UB}_l(v_2, Q)$, which is computed as in

Algorithm 3 r-nonbacktracking upper bound (rNB-UB)

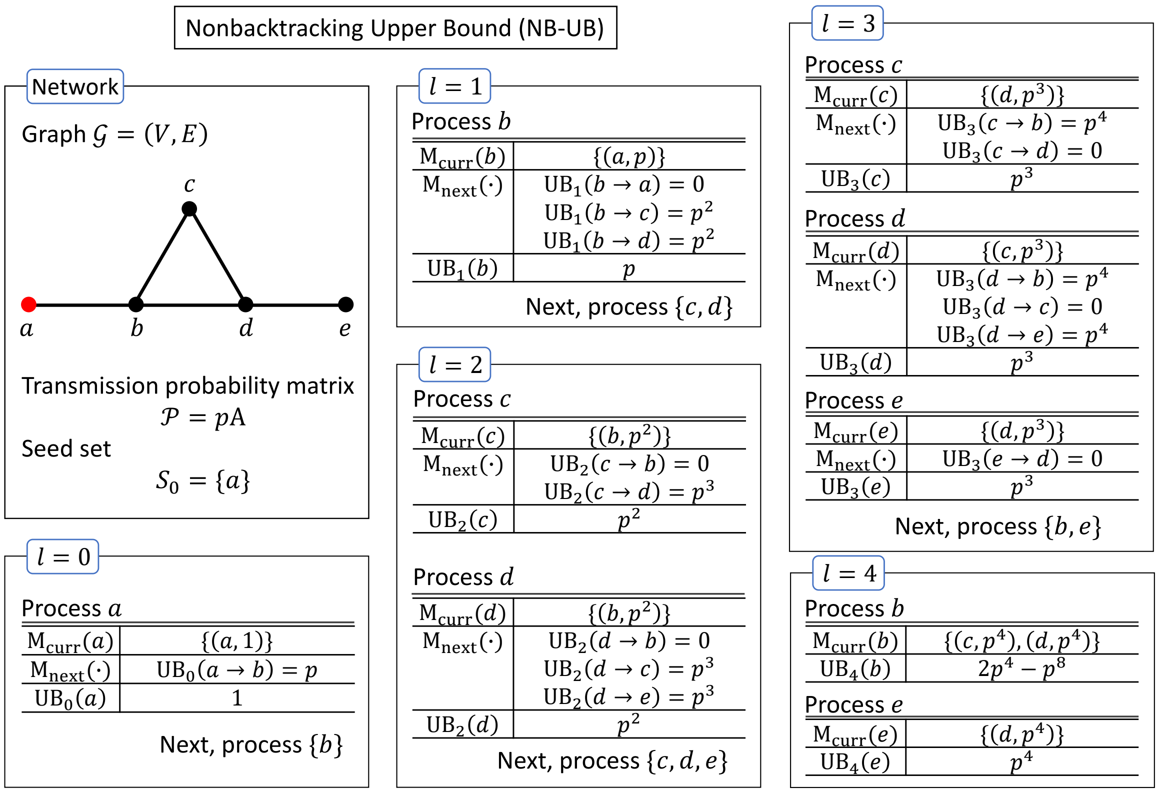
```
**parameter:** non-negative integer $2 \leq r \leq n-1$
**Initialize:** $\text{UB}_l(v) = 0$ for all $0 \leq l \leq n-1$ and $v \in V$
**for** $l = 0$ **to** $r-2$ **do**
    **for** $u \in V$ **do**
        $\text{UB}_l(u) = f(\cup_{P \in P_l(S_0 \to u)}(P, 1)), \forall\, l \leq r-1$
**for** $s \in S_0$ **do**
    **for** $P = (p_1, \ldots, p_r = u) \in P_{r-1}(s \to \cdot)$ **do**
        $\text{UB}_{r-1}(s, P) = 1$
        $\text{M}_{\text{next}}(u)$.insert$(P, \text{UB}_{r-1}(s, P))$
        $S_{r-1}$.insert$(u)$
**for** $l = r-1$ **to** $n-1$ **do**
    **for** $u \in S_l$ **do**
        $\text{M}_{\text{curr}}(u) = \text{M}_{\text{next}}(u)$
        Clear $\text{M}_{\text{next}}(u)$
        $\text{UB}_l(u) = f(\text{M}_{\text{curr}}(u))$
    **for** $u \in S_l$ **do**
        **for** $Q = (q_1, \ldots, q_r = v) \in \{Q' : P \sim Q', P \in P_{r-1}(\cdot \to u)\}$ **do**
            $\text{UB}_l(q_1, Q) = r\texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\text{M}_{\text{curr}}(u))$
            $\text{M}_{\text{next}}(v)$.insert$(Q, \text{UB}_l(q_1, Q))$
            $S_{l+1}$.insert$(v)$
**Output:** $\text{UB}_l(u)$ for all $0 \leq l \leq n-1$, $u \in V$
```

Equation (15), by the function $r\texttt{GenerateOutgoingMsg}_{\texttt{UB}}$. Note that v_2 is the start node of path Q, which is the second node in path P. Finally, the algorithm outputs $\text{UB}_l(u)$ for all $u \in V$ and $l \in \{0, \ldots, n-1\}$, and the upper bound $\sigma_r^+(S_0)$ is computed by Equation (19).

Computational complexity: To compute f, we first generate a radix tree using the input paths then compute the output from the leaves. Generating the radix tree takes $O(rM)$ where $r-1$ is the length of the paths and M is the number of paths since inserting a path in a radix tree takes $O(r)$. Then, we start the computation from the leaves to the root accessing each node once. Therefore, computing f takes $O(rM)$. Note that for every $l \geq r$ and $u \in V$, there are at most $|\deg(u)|V|^{r-2}|$ elements in $\text{M}_{\text{curr}}(u)$, since $\text{M}_{\text{curr}}(u)$ may contain all paths of length $r-1$ ending at node u. Therefore, function f takes $O(\deg(u)|V|^{r-2})$. To find the set $\{Q' : P \sim Q', P \in P_{r-1}(\cdot \to u)\}$, for every $P \in P_{r-1}(\cdot \to u)$, check if adding an out-neighbor of node u to the end of path P forms a length r path. This takes $O(\deg(p_r))$. We pre-compute this process for every path of length $r-1$, which takes $O((|E|+|V|)|V|^{r-1})$. In the algorithm, we compute function f at most $(n-r)n$ times and pre-compute the relationships $P \sim Q$ once, resulting the overall computational complexity $O((|E|+|V|)|V|^{r-1})$.

3.4. Comparisons of the Upper Bounds

In this section, we compare the three types of nonbacktracking upper bounds, NB-UB, tNB-UB, and rNB-UB, in a small independent cascade model. The model $IC(G, B)$ is defined



$l = 0$

Process a

$\mathrm{M}_{\mathrm{curr}}(a)$	$\{(a, 1)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_0(a \to b) = p$
$\mathrm{UB}_0(a)$	1

Next, process $\{b\}$

$l = 1$

Process b

$\mathrm{M}_{\mathrm{curr}}(b)$	$\{(a, p)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_1(b \to a) = 0$
	$\mathrm{UB}_1(b \to c) = p^2$
	$\mathrm{UB}_1(b \to d) = p^2$
$\mathrm{UB}_1(b)$	p

Next, process $\{c, d\}$

$l = 2$

Process c

$\mathrm{M}_{\mathrm{curr}}(c)$	$\{(b, p^2)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_2(c \to b) = 0$
	$\mathrm{UB}_2(c \to d) = p^3$
$\mathrm{UB}_2(c)$	p^2

Process d

$\mathrm{M}_{\mathrm{curr}}(d)$	$\{(b, p^2)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_2(d \to b) = 0$
	$\mathrm{UB}_2(d \to c) = p^3$
	$\mathrm{UB}_2(d \to e) = p^3$
$\mathrm{UB}_2(d)$	p^2

Next, process $\{c, d, e\}$

$l = 3$

Process c

$\mathrm{M}_{\mathrm{curr}}(c)$	$\{(d, p^3)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_3(c \to b) = p^4$
	$\mathrm{UB}_3(c \to d) = 0$
$\mathrm{UB}_3(c)$	p^3

Process d

$\mathrm{M}_{\mathrm{curr}}(d)$	$\{(c, p^3)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_3(d \to b) = p^4$
	$\mathrm{UB}_3(d \to c) = 0$
	$\mathrm{UB}_3(d \to e) = p^4$
$\mathrm{UB}_3(d)$	p^3

Process e

$\mathrm{M}_{\mathrm{curr}}(e)$	$\{(d, p^3)\}$
$\mathrm{M}_{\mathrm{next}}(\cdot)$	$\mathrm{UB}_3(e \to d) = 0$
$\mathrm{UB}_3(e)$	p^3

Next, process $\{b, e\}$

$l = 4$

Process b

$\mathrm{M}_{\mathrm{curr}}(b)$	$\{(c, p^4), (d, p^4)\}$
$\mathrm{UB}_4(b)$	$2p^4 - p^8$

Process e

$\mathrm{M}_{\mathrm{curr}}(e)$	$\{(d, p^4)\}$
$\mathrm{UB}_4(e)$	p^4

Figure 2: The step-wise illustration of NB-UB algorithm on the example network.

on an bidirected graph $G = (V, E)$, where $V = \{a, b, c, d, e\}$, $S_0 = \{a\}$, and every edge has the same transmission probability p.

In Figure 2, we describe how NB-UB algorithm runs on the network $IC(G, B)$. Since $|V| = 5$, the algorithm iterates from $l = 0$ to $l = 4$. On each step, it processes the nodes in the set S_l and passes the messages generated by them. For example, at $l = 3$, the nodes in $S_3 = \{c, d, e\}$ are processed. From previous iteration $l = 2$, node c sent the message $(c, \mathrm{UB}_2(c \to d))$ to d, and node d sent the message $(d, \mathrm{UB}_2(d \to c))$ to c and $(d, \mathrm{UB}_2(d \to e))$ to e. Thus,

$$\begin{aligned}
\mathrm{M}_{\mathrm{curr}}(c) &= \{(d, \mathrm{UB}_2(d \to c))\} = \{(d, p^3)\} \\
\mathrm{M}_{\mathrm{curr}}(d) &= \{(c, \mathrm{UB}_2(c \to d))\} = \{(c, p^3)\} \\
\mathrm{M}_{\mathrm{curr}}(e) &= \{(d, \mathrm{UB}_2(d \to e))\} = \{(d, p^3)\}
\end{aligned}$$

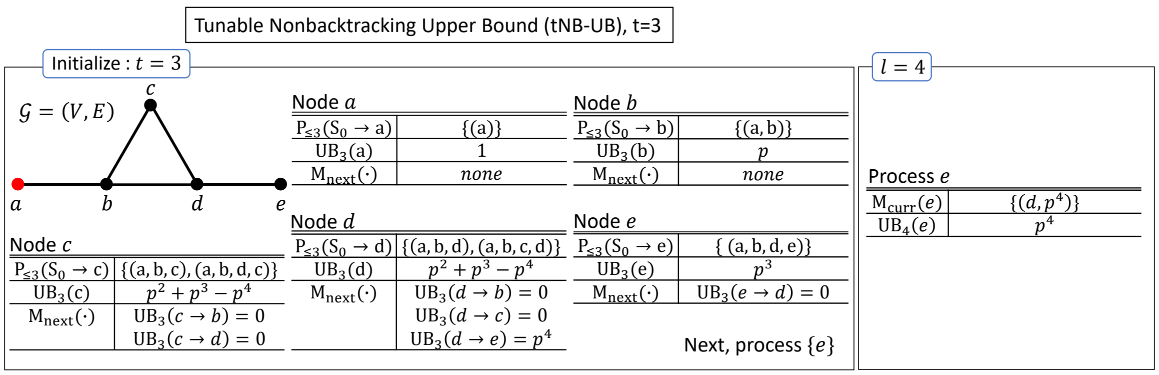
and the nodes c, d, e are processed as follows. First, compute the upper bounds at the step $l = 3$ as

$$\begin{aligned}
\mathrm{UB}_3(c) &= \texttt{ProcessIncomingMsg}_{\mathrm{UB}}(\mathrm{M}_{\mathrm{curr}}(c)) = p^3 \\
\mathrm{UB}_3(d) &= \texttt{ProcessIncomingMsg}_{\mathrm{UB}}(\mathrm{M}_{\mathrm{curr}}(d)) = p^3 \\
\mathrm{UB}_3(e) &= \texttt{ProcessIncomingMsg}_{\mathrm{UB}}(\mathrm{M}_{\mathrm{curr}}(e)) = p^3.
\end{aligned}$$

Next, each node computes the messages for its out-neighbors.

$$
\begin{aligned}
\mathrm{UB}_3(c \to b) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(0, \mathrm{UB}_3(c), B_{cb}) \\
&= B_{cb}\left(1 - \frac{1 - \mathrm{UB}_3(c)}{1 - 0}\right) = p^4 \\
\mathrm{UB}_3(c \to d) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\mathrm{UB}_2(d \to c), \mathrm{UB}_3(c), B_{cd}) \\
&= B_{cd}\left(1 - \frac{1 - \mathrm{UB}_3(c)}{1 - \mathrm{UB}_2(d \to c)}\right) = 0 \\
\\
\mathrm{UB}_3(d \to b) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(0, \mathrm{UB}_3(d), B_{db}) \\
&= B_{db}\left(1 - \frac{1 - \mathrm{UB}_3(d)}{1 - 0}\right) = p^4 \\
\mathrm{UB}_3(d \to c) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\mathrm{UB}_2(c \to d), \mathrm{UB}_3(d), B_{dc}) \\
&= B_{dc}\left(1 - \frac{1 - \mathrm{UB}_3(d)}{1 - \mathrm{UB}_2(c \to d)}\right) = 0 \\
\mathrm{UB}_3(d \to e) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(0, \mathrm{UB}_3(d), B_{de}) \\
&= B_{de}\left(1 - \frac{1 - \mathrm{UB}_3(d)}{1 - 0}\right) = p^4 \\
\mathrm{UB}_3(e \to d) &= \texttt{GenerateOutgoingMsg}_{\texttt{UB}}(\mathrm{UB}_2(d \to e), \mathrm{UB}_3(d), B_{ed}) \\
&= B_{ed}\left(1 - \frac{1 - \mathrm{UB}_3(d)}{1 - \mathrm{UB}_2(d \to e)}\right) = 0. \qquad (20)
\end{aligned}
$$

Then, node c send message $(b, \mathrm{UB}_3(c \to b))$ to b, and node d send messages $(d, \mathrm{UB}_3(d \to b))$ to b and $(d, \mathrm{UB}_3(d \to e))$ to e, concluding the process of the $l = 3$ step.



Node a

$P_{\leq 3}(S_0 \to a)$	$\{(a)\}$
$UB_3(a)$	1
$M_{next}(\cdot)$	*none*

Node b

$P_{\leq 3}(S_0 \to b)$	$\{(a,b)\}$
$UB_3(b)$	p
$M_{next}(\cdot)$	*none*

Node c

$P_{\leq 3}(S_0 \to c)$	$\{(a,b,c),(a,b,d,c)\}$
$UB_3(c)$	$p^2 + p^3 - p^4$
$M_{next}(\cdot)$	$UB_3(c \to b) = 0$ $UB_3(c \to d) = 0$

Node d

$P_{\leq 3}(S_0 \to d)$	$\{(a,b,d),(a,b,c,d)\}$
$UB_3(d)$	$p^2 + p^3 - p^4$
$M_{next}(\cdot)$	$UB_3(d \to b) = 0$ $UB_3(d \to c) = 0$ $UB_3(d \to e) = p^4$

Node e

$P_{\leq 3}(S_0 \to e)$	$\{(a,b,d,e)\}$
$UB_3(e)$	p^3
$M_{next}(\cdot)$	$UB_3(e \to d) = 0$

Process e

$M_{curr}(e)$	$\{(d, p^4)\}$
$UB_4(e)$	p^4

Figure 3: The step-wise illustration of tNB-UB algorithm on the example network.

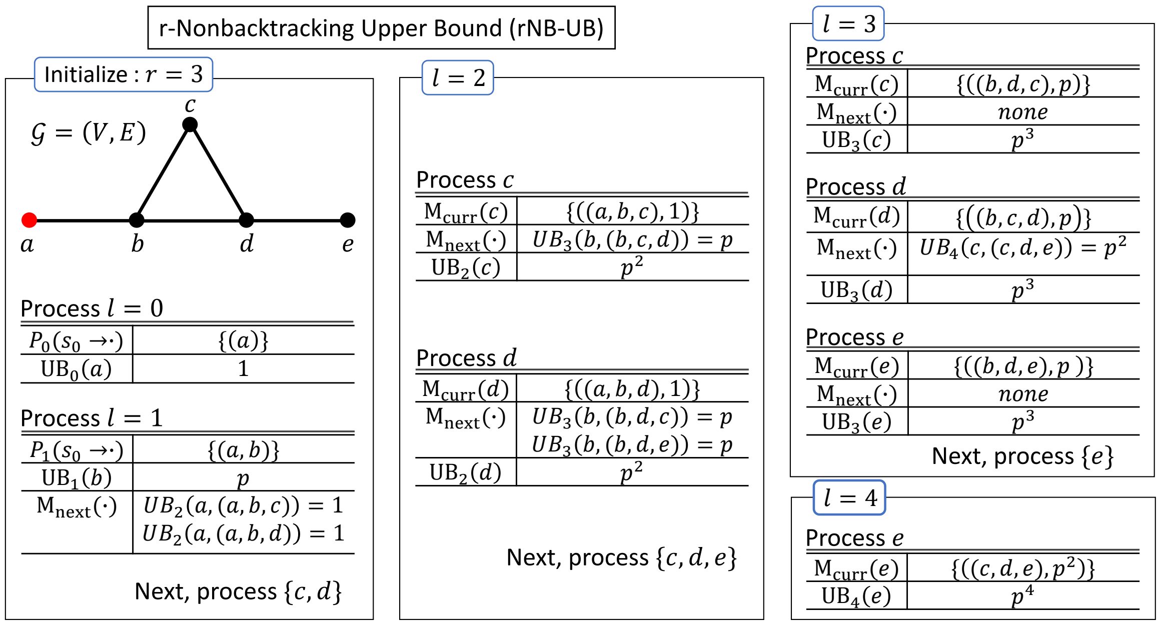
Next, we show the process of tNB-UB in the same network when $t = 3$ in Figure 3. As the first step, tNB-UB finds all paths from the seed a having the length less than or equal to 3. Then, for each node $u \in \{a, b, c, d, e\}$, it computes the exact probability $\mathrm{UB}_3(u)$ that the node is influenced through a path of length less than or equal to 3. For example,

$$
\begin{aligned}
\mathrm{UB}_3(d) &= \mathbb{P}(\text{Path } (a,b,d) \text{ is live or Path } (a,b,c,d) \text{ is live}) \\
&= p^2 + p^3 - p^4.
\end{aligned}
$$

To initialize the process, tNB-UB computes $\mathrm{UB}_3(u \to v)$, the probability that node v is influenced directly by u with a live path of length 4 from the seed a. For node d, there is a path (a, b, c, d) of length 3 from the seed, so we compute $\mathrm{UB}_3(d \to v)$ for its neighbors $\{b, c, e\}$. Notice that there is no path from a that ends with (d, b) with length 4 neither a path from a that ends with (d, c) with length 4. However, (a, b, c, d, e) is a path from a ending with (d, e). Thus,

$$\begin{aligned}
\mathrm{UB}_3(d \to b) &= 0\\
\mathrm{UB}_3(d \to c) &= 0\\
\mathrm{UB}_3(d \to e) &= \mathbb{P}(\text{Path } (a, b, c, d, e) \text{ is live}) = p^4.
\end{aligned}$$

After the initialization, tNB-UB repeats the same process as NB-UB from $l = t + 1$ to $l = n - 1$.



Process $l = 0$

$P_0(s_0 \to \cdot)$	$\{(a)\}$
$\mathrm{UB}_0(a)$	1

Process $l = 1$

$P_1(s_0 \to \cdot)$	$\{(a, b)\}$
$\mathrm{UB}_1(b)$	p
$\mathrm{M}_{\text{next}}(\cdot)$	$UB_2(a, (a, b, c)) = 1$ $UB_2(a, (a, b, d)) = 1$

Process c

$\mathrm{M}_{\text{curr}}(c)$	$\{((a, b, c), 1)\}$
$\mathrm{M}_{\text{next}}(\cdot)$	$UB_3(b, (b, c, d)) = p$
$\mathrm{UB}_2(c)$	p^2

Process d

$\mathrm{M}_{\text{curr}}(d)$	$\{((a, b, d), 1)\}$
$\mathrm{M}_{\text{next}}(\cdot)$	$UB_3(b, (b, d, c)) = p$ $UB_3(b, (b, d, e)) = p$
$\mathrm{UB}_2(d)$	p^2

Process c

$\mathrm{M}_{\text{curr}}(c)$	$\{((b, d, c), p)\}$
$\mathrm{M}_{\text{next}}(\cdot)$	*none*
$\mathrm{UB}_3(c)$	p^3

Process d

$\mathrm{M}_{\text{curr}}(d)$	$\{((b, c, d), p)\}$
$\mathrm{M}_{\text{next}}(\cdot)$	$UB_4(c, (c, d, e)) = p^2$
$\mathrm{UB}_3(d)$	p^3

Process e

$\mathrm{M}_{\text{curr}}(e)$	$\{((b, d, e), p)\}$
$\mathrm{M}_{\text{next}}(\cdot)$	*none*
$\mathrm{UB}_3(e)$	p^3

Process e

$\mathrm{M}_{\text{curr}}(e)$	$\{((c, d, e), p^2)\}$
$\mathrm{UB}_4(e)$	p^4

Figure 4: The step-wise illustration of rNB-UB algorithm on the example network.

In Figure 4, we illustrate rNB-UB in the same example network. For the initialization, rNB-UB computes $\mathrm{UB}_i(v)$ for all $i \in \{0, \dots, r-2\} = \{0, 1\}$ and $v \in \{a, b, c, d, e\}$. Since there is only one length 0 path from the seed a, we have $P = (a)$, $\mathrm{UB}_0(a) = 1$ and $\mathrm{UB}_0(v) = 0, \forall v \in \{b, c, d, e\}$. Similarly, $\mathrm{UB}_1(b) = p$ and $\mathrm{UB}_1(v) = 0, \forall v \in \{a, c, d, e\}$. Then, the algorithm finds all paths from the seed a with length $r - 1 = 2$ and sends messages $(P, \mathrm{UB}_2(a, P) = 1)$ to the end node of the each path. In this network, there are two paths of length 2, (a, b, c) and (a, b, d). Thus, rNB-UB sends $((a, b, c), 1)$ to node c and $((a, b, d), 1)$ to node d. From $l = r-1$ to $n-1$, each node v in set S_l first computes the upper bound $\mathrm{UB}_l(v)$ by the function f in Definition 10. For example, at $l = 2$, nodes in $S_2 = \{c, d\}$ compute the following upper bounds. Since both $\mathrm{M}_{\text{curr}}(c) = \{(a, b, c), 1\}$ and $\mathrm{M}_{\text{curr}}(d) = \{(a, b, d), 1\}$

have only one element each,

$$\begin{aligned}\mathrm{UB}_2(c) &= 1-(1-p(1-(1-p))) = p^2\\ \mathrm{UB}_2(d) &= 1-(1-p(1-(1-p))) = p^2.\end{aligned}$$

Next, each node in set S_l checks whether adding its out-neighbor to the previously received messages' paths forms a path or not. If so, send a message to the neighbor with the updated path and value. For example, node c has the message $((a,b,c),1)$ from $l=1$ step. For node c's two out-neighbors b and d, adding d forms a path but not adding b. Thus, node c sends the message $((b,c,d),p)$ to node d, where (b,c,d) is length 2 sub-path of the path (a,b,c,d) ending at node d. Similarly, d sends the message $((b,d,c),p)$ to node c and $((b,d,e),p)$ to node e, concluding the $l=2$ step.

Now, we compare the three upper bounds, NB-UB, tNB-UB, and rNB-UB, on the example graph $\mathcal{G}$ to the exact influence σ.

$$\begin{aligned}\sigma &= 1+p+(p^2+p^3-p^4)+(p^2+p^3-p^4)+(p^3+p^4-p^5)\\ &= 1+p+2p^2+3p^3-p^4-p^5\\ \sigma^+ &= 1+(p+2p^4-2p^5-p^8+p^9)+(p^2+p^3-p^5)+(p^2+p^3-p^5)+(p^3+p^4-p^7)\\ &= 1+p+2p^2+3p^3+3p^4-4p^5-p^7-p^8+p^9\\ \sigma_t^+ &= 1+p+(p^2+p^3-p^4)+(p^2+p^3-p^4)+(p^3+p^4-p^7)\\ &= 1+p+2p^2+3p^3-p^4-p^7\\ \sigma_r^+ &= 1+p+(p^2+p^3-p^5)+(p^2+p^3-p^5)+(p^3+p^4-p^7)\\ &= 1+p+2p^2+3p^3+p^4-2p^5-p^7\end{aligned}$$

The tunable upper bound σ_t^+ with $t=3$ results in the tightest bound. tNB-UB computes the influence probability for nodes a,b,c and d exactly, since they can only be influenced by a path with length less than or equal to 3. The r-nonbacktracking bound σ_r^+ with $r=3$ avoids all triangles when computing the bound. Thus, it only considers paths rather than walks. However, rNB-UB is still not exact, because it assumes the independence among the events that a node is influenced by a length l path, for all $l \le n-1$. The nonbacktracking bound σ^+ results in the worst bound. The bound not only assumes the independence as in rNB-UB, but also considers all walks that include the triangle formed by nodes b,c and d.

4. Nonbacktracking Lower Bounds

In this section, we introduce a nonbacktracking lower bound on the influence in the independent cascade model. The bound uses nonbacktracking walks and is computed efficiently by message passing algorithms in $O(|V|+|E|)$. Similar to nonbacktracking upper bound, we also present a parameterized version of the lower bound, tunable nonbacktracking lower bound.

Furthermore, from the proposed upper σ^+ and lower σ^- bounds on the expected number of influenced nodes, we can compute an upper bound on the variance given by $(\sigma^+-\sigma^-)^2/4$. This could be used to estimate the number of computations needed by MC simulations. Computing the upper bound on the variance with the proposed bounds can be done in $O(|V|^2+|V||E|)$, whereas computing the variance with MC requires $O(|V|^5+|V|^4|E|)$.

This section is organized as follows. First, sections 4.1 and 4.2 shows the definitions of the proposed lower bounds and efficient algorithms to compute them. Next, section 4.3 shows how each lower bound is computed on a small example network and compares the bounds with the influence.

4.1. Nonbacktracking Lower Bound (NB-LB)

A naive way to compute a lower bound on the influence of a seed set S_0 in a network $IC(G, B)$ is to reduce the network to a (spanning) tree network, by removing edges. Then, since there is a unique path from a node to another, we can compute the influence of the tree network, which is a lower bound on the influence in the original network, in $O(|V|)$. We take this approach of generating a subnetwork from the original network, yet we avoid the significant gap between the bound and the influence by considering the following directed acyclic subnetwork, in which there is no backtracking walk.

Definition 11 (Min-distance Directed Acyclic Subnetwork)**.** *Consider an independent cascade model $IC(G, B)$, where $G = (V, E)$ and $|V| = n$, and a seed set $S_0 \subseteq V$. Let $d(S_0, v) := \min_{s \in S_0} d(s, v)$, i.e., the minimum distance from a seed in S_0 to v. A minimum-distance directed acyclic subnetwork (MDAS), $IC(G', B')$, where $G' = (V', E')$, is obtained as follows.*

- $V' = \{v_1, ..., v_n\}$ *is an ordered set of nodes such that for every $i < j$, $d(S_0, v_i) \leq d(S_0, v_j)$.*
- $E' = \{(v_i, v_j) \in E : i < j\}$, *i.e., E' is obtained from E by removing edges whose source node comes later in the order than its destination node.*
- $B'_{v_i v_j} = B_{v_i v_j}$, *if $(v_i, v_j) \in E'$, and $B'_{v_i v_j} = 0$, otherwise.*

If there are multiple ordered sets of vertices satisfying the condition, we may choose one arbitrarily. While any of the ordered sets satisfying the condition results in a lower bound, the tightness of the bound can varies depending on the ordered set chosen for the computation. This is further explained in 4.3 with an example.

For any $k \in [n]$, let $p(v_k)$ be the probability that $v_k \in V'$ is influenced in MDAS, $IC(G', B')$. Since $p(v_k)$ is equivalent to the probability of the union of the events that an in-neighbor $u_i \in N^-(v_k)$ influences v_k, $p(v_k)$ can be computed by the principle of inclusion and exclusion. Thus, we may compute a lower bound on $p(v_k)$, using Bonferroni inequalities, if we know the probabilities that in-neighbors u and v both influences v_k, for every pair $u, v \in N^-(v_k)$. However, computing such probabilities can take $O(k^k)$. Hence, we present $\mathrm{LB}(v_k)$ which efficiently computes a lower bound on $p(v_k)$ by the following recursion.

Definition 12. *For all $v_k \in V'$, $\mathrm{LB}(v_k) \in [0, 1]$ is defined by the recursion on k as follows.*
Initial condition: *For every $v_s \in S_0$,*

$$\mathrm{LB}(v_s) = 1. \tag{21}$$

Recursion: *For every $v_k \in V' \setminus S_0$,*

$$\mathrm{LB}(v_k) = \sum_{i=1}^{m^*} \left(B'_{u_i v_k} \mathrm{LB}(u_i)(1 - \sum_{j=1}^{i-1} B'_{u_j v_k}) \right), \tag{22}$$

where $N^-(v_k) = \{u_1, \ldots, u_m\}$ is the ordered set of in-neighbors of v_k in $IC(G', B')$ and $m^* = \max\{m' \leq m : \sum_{j=1}^{m'-1} B'_{u_j v_k} \leq 1\}$.

Remark. Since the i-th summand in Equation (22) can use $\sum_{j=1}^{i-2} B'_{u_j v_k}$, which is already computed in $(i-1)$-th summand, to compute $\sum_{j=1}^{i-1} B'_{u_j v_k}$, the summation takes at most $O(\deg(v_k))$. Note that Equation (22) is formulated so that the computational complexity is minimized. However, we may use some other equations to achieve the lower bound property.

Theorem 5 (Nonbacktracking Lower Bound (NB-LB))**.** *For any independent cascade model $IC(G, B)$, a seed set S_0 and its MDAS, $IC(G', B')$,*

$$\sigma(S_0) \geq \sum_{v_k \in V'} \mathrm{LB}(v_k) =: \sigma^-(S_0), \tag{23}$$

where $\mathrm{LB}(v_k)$ is obtained recursively as in Definition 12.

Next, we present Nonbacktracking Lower Bound (NB-LB) algorithm which efficiently computes $\mathrm{LB}(v_k)$. At k-th iteration, the variables in NB-LB represent the followings.

· $\mathrm{M}(v_k) = \{(\mathrm{LB}(v_j), B'_{v_j v_k}) : v_j \text{ is an in-neighbor of } v_k\}$, set of pairs (incoming message from an in-neighbor v_j to v_k, the transmission probability of edge (v_j, v_k)).

Algorithm 4 Nonbacktracking Lower Bound (NB-LB)

```
Input: directed acyclic network $IC(G', B')$
Initialize: $\sigma^- = 0$
Initialize: Insert $(1, 1)$ to $\mathrm{M}(v_i)$ for all $v_i \in S_0$
for $k = 1$ to $n$ do
    $\mathrm{LB}(v_k) = \texttt{ProcessIncomingMsg}_{\texttt{LB}}(\mathrm{M}(v_k))$
    $\sigma^-$ += $\mathrm{LB}(v_k)$
    for $v_l \in N^+(v_k) \setminus S_0$ do
        $\mathrm{M}(v_l)$.insert$((\mathrm{LB}(v_k), B'_{v_k v_l}))$
Output: $\sigma^-$
```

At the beginning, every seed node $s \in S_0$ is initialized such that $\mathrm{M}(s) = \{(1, 1)\}$ in order to satisfy the initial condition, $\mathrm{LB}(s) = 1$. For each k-th iteration, node v_k is processed as follows. First, $\mathrm{LB}(v_k)$ is computed as in the Equation (22), by the function $\texttt{ProcessIncomingMsg}_{\texttt{LB}}$, and added to σ^-. Second, v_k passes the message $(\mathrm{LB}(v_k), B'_{v_k v_l})$ to its out-neighbor $v_l \in N^+(v_k) \backslash S_0$, and v_l stores (inserts) it in $\mathrm{M}(v_l)$. Finally, the algorithm outputs σ^-, the lower bound on the influence.

Computational complexity: Obtaining an arbitrary directed acyclic subnetwork from the original network takes $O(|V| + |E|)$. Next, the algorithm iterates through the nodes $V' = \{v_1, \ldots, v_n\}$. For each node v_k, $\texttt{ProcessIncomingMsg}_{\texttt{LB}}$ takes $O(\deg(v_k))$ and v_k sends messages to its out-neighbors in $O(\deg(v_k))$. Hence, the worst case computational complexity is $O(|V| + |E|)$.

4.2. Tunable Nonbacktracking Lower Bound (tNB-LB)

The first step of tunable NB-LB is the same as NB-LB, which is to order the vertex set as $V' = \{v_1, \ldots, v_n\}$ to satisfy $P(S_0, v_i) \leq P(S_0, v_j)$, for every $i < j$. Next, given a non-negative integer parameter $t \leq n$, we obtain a t-size subnetwork $IC(G[V_t], B[V_t])$ and a new seed set $S_0 \cap V_t$, where $G[V_t]$ is the vertex-induced subgraph which is induced by the set of nodes $V_t = \{v_1, \ldots, v_t\}$, and $B[V_t]$ is the corresponding transmission probability matrix. For each $v_i \in V_t$, we compute the exact probability $p_t(v_i)$ that node v_i is influenced from the new seed set $S_0 \cap V_t$ in the subnetwork $IC(G[V_t], B[V_t])$. Then, we start (non-parameterized) NB-LB algorithm from $k = t + 1$ with the new initial condition: for all $k \leq t$,

$$\mathrm{LB}(v_k) = p_t(v_k). \tag{24}$$

Finally, tNB-LB computes the lower bound as $\sum_{v_k \in V'} \mathrm{LB}(v_k)$.

Algorithm 5 Tunable nonbacktracking lower bound (tNB-LB)

```
**parameter:** non-negative integer $t \leq n$
**Initialize:** $\sigma^- = 0$
**for** $k = 1$ **to** $t$ **do**
    $\mathrm{LB}(v_k) = p_t(v_k)$
    $\sigma^- \mathrel{+}= \mathrm{LB}(v_k)$
    **for** $v_i \in \{N^+(v_k) \cap \{v_j : j > t\}\}$ **do**
        $\mathrm{M}(v_i)$.insert$((\mathrm{LB}(v_k), B'_{v_k v_i}))$
**for** $k = t + 1$ **to** $n$ **do**
    $\mathrm{LB}(v_k) = \texttt{ProcessIncomingMsg}_{\texttt{LB}}(\mathrm{M}(v_k))$
    $\sigma^- \mathrel{+}= \mathrm{LB}(v_k)$
    **for** $v_i \in N^+(v_k) \setminus S_0$ **do**
        $\mathrm{M}(v_i)$.insert$((\mathrm{LB}(v_k), B'_{v_k v_i}))$
**Output:** $\sigma^-$
```

For larger t, the algorithm results in a tighter bound. However, the computational complexity may increase exponentially with respect to t. To avoid such large computational complexity, the algorithm can adopt Monte Carlo simulations on the subnetwork. However, this modification results in probabilistic lower bounds, rather than theoretically guaranteed lower bounds. Nonetheless, this can still give a significant improvement, because the Monte Carlo simulations on a smaller network require less computation to stabilize the estimation.

4.3. Illustration of Nonbacktracking Lower Bounds (NB-LB)

In Figure 5, we show step-wise lower bound computation by NB-LB on a small network $IC(G, B)$ defined on an bidirected graph $G = (V, E)$, where $V = \{a, b, c, d\}$, $S_0 = \{a\}$, and every edge has the same transmission probability p. For each k, Table 1 shows the values of the key variables, $\mathrm{M}(v_k)$, $\mathrm{LB}(v_k)$, and $(\mathrm{LB}(v_k), B'_{v_k v_l})$ for the out-neighbors $v_l \in N^+(v_k) \setminus S_0$, and shows the change in σ^-.

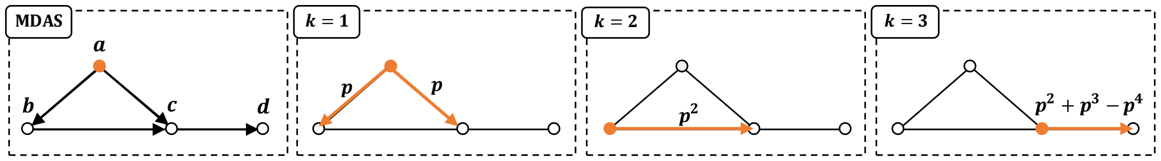


Figure 5: The step-wise illustration of NB-LB on the example network.

	$k=1$	$k=2$	$k=3$	$k=4$
v_k	a	b	c	d
$\mathrm{M}(v_k)$	$\{(1,1)\}$	$\{(1,p)\}$	$\{(1,p),(p,p)\}$	$\{(p+p^2-p^3,p)\}$
$\mathrm{LB}(v_k)$	1	p	$p+p^2-p^3$	$p^2+p^3-p^4$
$N^+(v_k)\setminus S_0$	$\{b,c\}$	$\{c\}$	$\{d\}$	$\varnothing$
$(\mathrm{LB}(v_k), B'_{v_k v_l})$ to v_l	$(1,p)$ to b and c	(p,p) to c	$(p+p^2-p^3,p)$ to d	
σ^-	1	$1+p$	$1+2p+p^2-p^3$	$1+2p+2p^2-p^4$

Table 1: The values of the key variables in NB-LB on the example network in Figure 5.

We obtain MDAS from the network as follows. Since $d(S_0,a)=0$, $d(S_0,b)=d(S_0,c)=1$ and $d(S_0,d)=2$, we order the vertices as $\{v_1=a, v_2=b, v_3=c, v_4=d\}$ so that $d(S_0,v_i)\le d(S_0,v_j)$, for every $i<j$.

NB-LB algorithm processes the nodes $\{v_1=a, v_2=b, v_3=c, v_4=d\}$ sequentially. For example, at $k=3$, node c is processed. During the previous step at $k=1$, node a sent the message $(\mathrm{LB}(a), B'_{ac})$ to node c, and at $k=2$, node b sent the message $(\mathrm{LB}(b), B'_{bc})$ to node c. Thus, node c has the message:

$$\mathrm{M}(c) = \{(\mathrm{LB}(a), B'_{ac}), (\mathrm{LB}(b), B'_{bc})\} = \{(1,p),(p,p)\}.$$

Then, it computes $\mathrm{LB}(c)$ with the function $\texttt{ProcessIncomingMsg}_{\texttt{LB}}$.

$$\begin{aligned}\mathrm{LB}(c) &= \texttt{ProcessIncomingMsg}_{\texttt{LB}}(\mathrm{M}(c))\\ &= B'_{ac}\mathrm{LB}(a) + B'_{bc}\mathrm{LB}(b)(1-B'_{ac}) = p+p^2-p^3.\end{aligned}$$

Recall that $\sigma^- = 1+p$, at the end of the iteration $k=2$. Thus, at $k=3$,

$$\sigma^- = 1+p+\mathrm{LB}(c) = 1+2p+p^2-p^3.$$

Next, since $N^+(c)\setminus S_0 = \{d\}$, node c sends the message $(\mathrm{LB}(c), B_{cd}) = (p+p^2-p^3, p)$ to node d, concluding the process of the $k=3$ step. At $k=4$, the algorithm computes $\mathrm{LB}(d) = p(p+p^2-p^3)$ and the final lower bound:

$$\sigma^- = 1+2p+p^2-p^3+\mathrm{LB}(d) = 1+2p+2p^2-p^4.$$

For this example network, there is another MDAS that satisfies the condition in 11 with the ordered vertices $\{v_1=a, v_2=c, v_3=b, v_4=d\}$. We can also obtain a lower bound using this MDAS, which is $1+2p+2p^2-p^3$. Notice that this value is different from the lower bound we computed above but still is a lower bound for the expected number of the influenced nodes.

Next, we compute tNB-LB on the same network with $t = 3$. First, for $k = 1$ to $k = 3$, the probabilities that nodes a, b, c are influenced are exactly computed as follows.

$$\begin{aligned} \mathrm{LB}(a) &= 1 \\ \mathrm{LB}(b) &= p + p^2 - p^3 \\ \mathrm{LB}(c) &= p + p^2 - p^3 \end{aligned}$$

Then, for $k = 4$, the algorithm computes the message $\mathrm{M}(d) = \{(\mathrm{LB}(c), B'_{cd})\}$ and, in turns, the lower bound $\mathrm{LB}(d) = B'_{cd}\mathrm{LB}(c) = p(p + p^2 - p^3)$. Thus, the final lower bound is

$$\sigma_t^- = 1 + 2p + 3p^2 - p^3 - p^4.$$

Finally, we compare the lower bounds to the influence.

$$\begin{aligned} \sigma &= 1 + 2p + 3p^2 - p^3 - p^4 \\ \sigma^- &= 1 + 2p + 2p^2 - p^4 \\ \sigma_t^- &= 1 + 2p + 3p^2 - p^3 - p^4 \end{aligned}$$

As expected, tNB-LB results in a tighter lower bound than NB-LB. In fact, tNB-LB is the same as the influence since node d can only be influenced through node c whose probability of being influenced is exactly computed in tNB-LB with $t = 3$.

5. Experimental Results

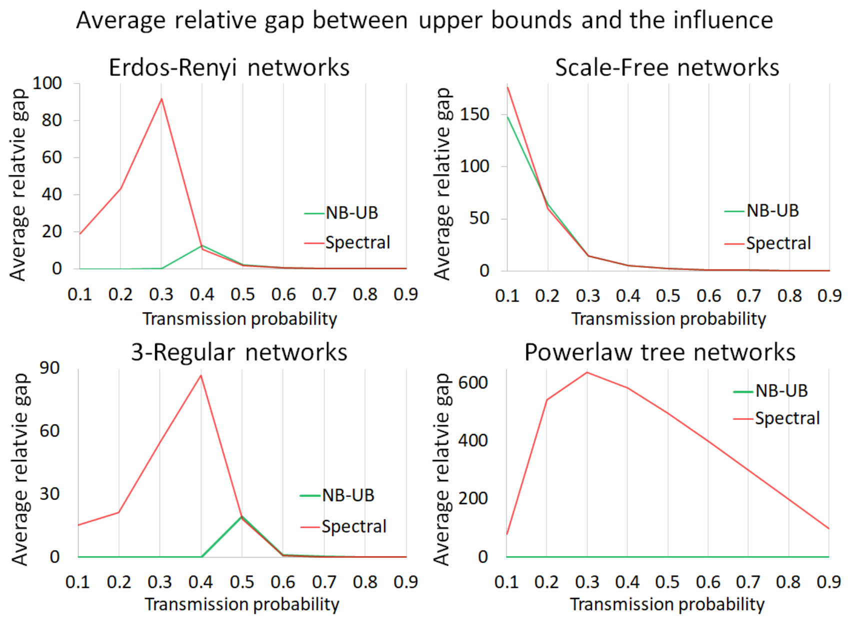
In this section, we evaluate NB-UB and NB-LB in independent cascade models on a variety of classical synthetic networks.

Network Generation. We consider 4 classical random graph models with the parameters shown as follows: Erdos Renyi random graphs with $ER(n = 1000, p = 0.003)$, scale-free networks $SF(n = 1000, \alpha = 2.5)$, random regular graphs $Reg(n = 1000, d = 3)$, and random tree graphs with power-law degree distributions $T(n = 1000, \alpha = 3)$. For each graph model, we generate 100 networks $IC(G, pA)$ and a seed as follows. The bidirected graph G is defined on the largest connected component of a graph drawn from the graph model, the seed node s is a randomly selected vertex, and A is the adjacency matrix of G. The corresponding IC model has the same transmission probability p for every edge.

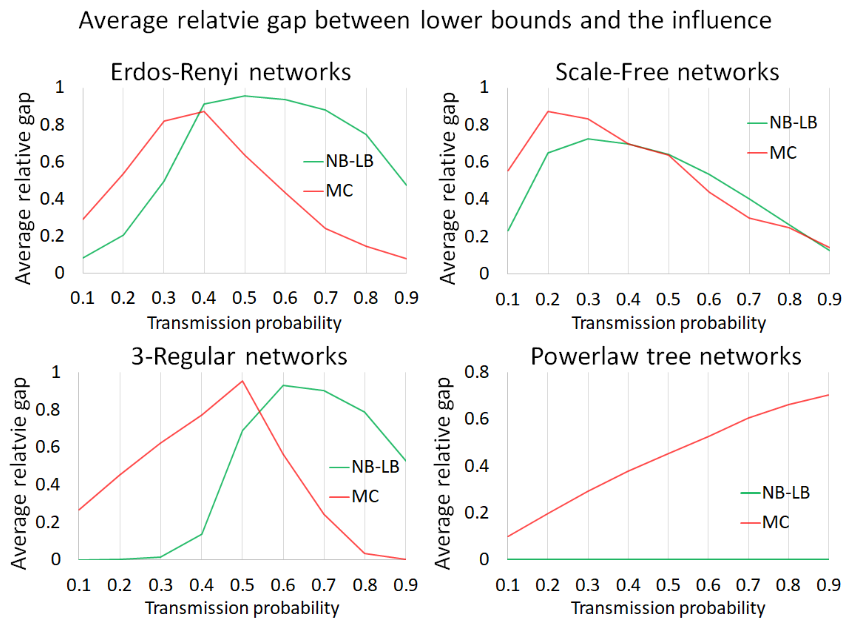
Evaluation of Bounds. For each generated network, we compute the following quantities for each $p \in \{0.1, 0.2, \dots, 0.9\}$.

- σ_{mc}: the estimation of the influence with 10^6 Monte Carlo simulations.
- σ^+: the upper bound obtained by NB-UB.
- σ^+_{spec}: the spectral upper bound by (Lemonnier et al., 2014).
- σ^-: the lower bound obtained by NB-LB.
- σ^-_{prob}: the probabilistic lower bound obtained by 10 Monte Carlo simulations.

There has not been any tight lower bound for general networks. The lower bound introduced in (Khim et al., 2016) is efficient and tight for tree networks. However, for general



(a)



(b)

Figure 6: (a) The average relative gap of the upper bounds: NB-UB and the spectral upper bound in (Lemonnier et al., 2014) for various types of networks.
(b) The average relative gap of the lower bounds: NB-LB and the probabilistic lower bound computed by MC simulations for various types of networks.

networks, since it considers all paths of length $k \in \{0, 1, ..., m\}$ for some $m \in [n]$, the computational complexity is $O(|V|^m)$ which is greater than the computational complexity of the nonbacktracking lower bound for any $m > 2$. Therefore, the probabilistic lower bound is chosen for the experiments. The sample size of 10 is determined to exceed the computational complexity of NB-LB algorithm. In Figure 6, we compare the average relative gap of the bounds for every network model and for each transmission probability, where the true value is assumed to be σ_{mc}. For example, the average relative gap of NB-UB for 100 Erdos Renyi networks $\{\mathcal{N}_i\}_{i=1}^{100}$ with the transmission probability p is computed by $\frac{1}{100}\sum_{i\in[100]}\frac{\sigma^+[\mathcal{N}_i]-\sigma_{mc}[\mathcal{N}_i]}{\sigma_{mc}[\mathcal{N}_i]}$, where $\sigma^+[\mathcal{N}_i]$ and $\sigma_{mc}[\mathcal{N}_i]$ denote NB-UB and the MC estimation, respectively, for the network $\mathcal{N}_i$.

Results. Figure 6 shows that NB-UB outperforms the upper bound in (Lemonnier et al., 2014) for the Erdos-Renyi and random bidirected 3-regular networks, and performs comparably for the scale-free networks. Also, NB-LB gives tighter bounds than the MC bounds on the Erdos-Renyi, scale-free, and random regular networks when the transmission probability is small, $p < 0.4$. NB-UB and NB-LB compute the exact influence for the tree networks since both algorithms avoid backtracking walks.

Next, we show the bounds on exemplary networks.

5.1. Upper Bounds

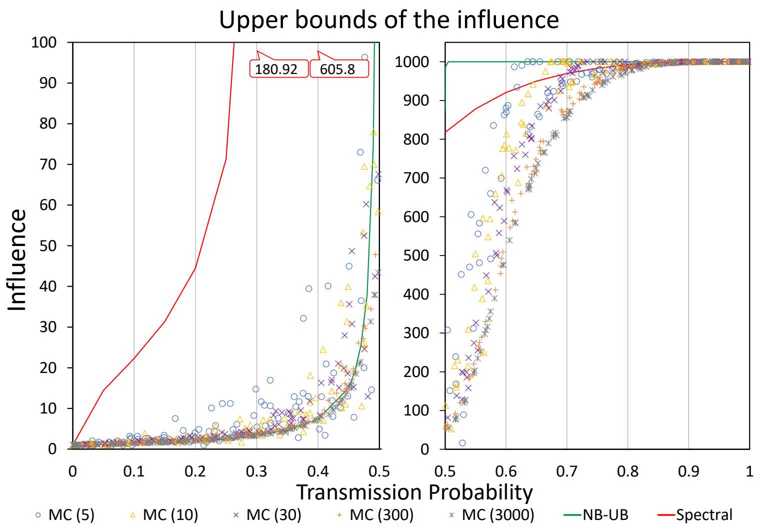


Figure 7: The figure compares various upper bounds on the influence in the 3-regular network in section 5.1. The MC upper bounds are computed with various simulation sizes and shown with the data points indicated with MC(N), where N is the number of simulations. The spectral upper bound in (Lemonnier et al., 2014) is shown in red line, and NB-UB is shown in green line.

In order to illustrate typical behavior of the bounds, we have chosen the network in Figure 7 as follows. First, we generate 100 random 3-regular graphs G with 1000 nodes and assign a random seed s. Then, the corresponding IC model is defined as $IC(G, B = pA)$. For each network, we compute NB-UB and MC estimation. Then, we compute the score for each network, where the score is defined as the sum of the square differences between the upper bounds and MC estimations over the transmission probability $p \in \{0.1, 0.2, \ldots, 0.9\}$. Finally, a graph whose score is the median from all 100 scores is chosen for Figure 7.

In figure 7, we compare 1) the upper bounds introduced (Lemonnier et al., 2014) and 2) the probabilistic upper bounds obtained by Monte Carlo simulations with 99% confidence level, to NB-UB. The MC upper bounds are computed with the various sample sizes $N \in \{5, 10, 30, 300, 3000\}$. It is evident from the figure that a larger sample size provides a tighter probabilistic upper bound. NB-UB outperforms the bound by (Lemonnier et al., 2014) and the probabilistic MC bound when the transmission probability is relatively small. Further, it shows a similar trend as the MC simulations with large sample size.

Next, we show r-nonbacktracking upper bounds on a small world graph $G = SW(n, k, q)$ where $n = 1000$ is the number of nodes in the network, $k = 4$ is the number of neighbors of each node, and $q = 0.01$ is the re-wiring probability. The IC model on the small world graph G is defined as $IC(G, B = pA)$, where A is the adjacency matrix of G, p is the transmission probability for every edge, and the seed node s is chosen arbitrarily.

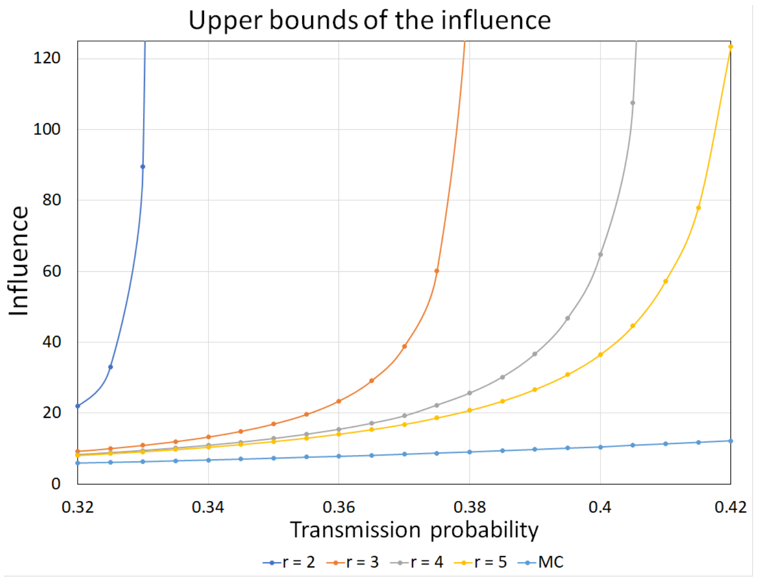


Figure 8: The figure compares r-nonbacktracking upper bounds with various values of r in a small world network. The MC estimation with simulation size 10^6 is computed for comparison.

In Figure 8, we compare the r-nonbacktracking upper bounds with $r \in \{2, 3, 4, 5\}$ to MC estimation. For $p < 0.32$, all upper bounds are very close to the influence, and for $p > 0.42$ the upper bounds with $r \leq 5$ become close to the maximum value, 1000. Therefore, we show the bounds for $p \in [0.32, 0.42]$. Note that when $r = 2$, the r-nonbacktracking upper bound (rNB-UB) is the nonbacktracking upper bound (NB-UB). By avoiding $r = 3$ backtracking walks (triangles) in addition to backtracking walks, the upper bound is significantly improved. The same applies to $r = 4, 5$ nonbacktracking bounds. As r increases, the bounds get tighter and closer to the estimated influence. It is worth noting that while the upper bound results in deterministic, as opposed to probabilistic, upper bound, the computational complexity for $r \geq 4$ exceed the complexity of MC approximation.

5.2. Lower Bounds

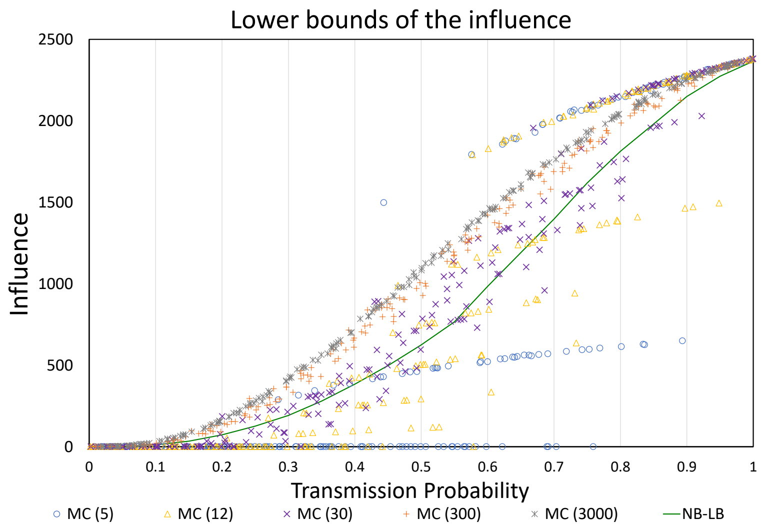


Figure 9: The figure shows lower bounds on the influence of a scale-free network in section 5.2. The probabilistic lower bounds are obtained by MC with various simulation sizes. The data points indicated with MC(N) are obtained by N number of simulations. NB-LB is shown in green line.

In this section, we show lower bounds on an exemplary network. We adopt a similar selection process as described in Section 5.1 but with the scale-free networks, with 3000 nodes and $\alpha = 2.5$.

We compare probabilistic lower bounds obtained by MC with 99% confidence level to NB-LB. The bounds from Monte Carlo simulations are computed with various sample sizes $N \in \{5, 12, 30, 300, 3000\}$, which accounts for a constant, $\log(|V|)$, $0.01|V|$, $0.1|V|$, and $|V|$. NB-LB outperforms the probabilistic bounds by MC with small sample sizes. Recall that the computational complexity of NB-LB is $O(|V| + |E|)$, which is the computational complexity of a constant number of Monte Carlo simulations. In Figure 9, it shows that NB-LB is tighter than the probabilistic lower bounds with the same computational complexity.

6. Conclusion

We propose both upper and lower bounds on the influence in the independent cascade models and provide algorithms to efficiently compute the bounds. The proposed upper bound is monotone and submodular. We extend the results by proposing tunable bounds which can adjust the trade-off between the efficiency and the accuracy and by using r-nonbacktracking walks instead of (2-)nonbacktracking walks. Finally, the tightness and the performance of bounds are shown with the experimental results.

Acknowledgments

The authors thank Colin Sandon for helpful discussions. This research was partly supported by the NSF CAREER Award CCF-1552131, the ARO grant W911NF-16-1-0051, and NSF Center for Science of Information (CSoI) Grant CCF-0939370.

Appendix A. Proof of Theorem 2

We start by defining the following events for the independent cascade model $IC(G, B)$, where $G = (V, E)$ and $|V| = n$, and a seed set $S_0 \subseteq V$.

Definition 13. *For any $u, v \in V$, $l \in \{0, \dots, n-1\}$, and $S \subseteq V$, we define*

$$\begin{aligned} A(v) &= \{v \textit{ is influenced}\} \\ A_l(v) &= \cup_{P \in P_l(S_0 \to v)} \{P \textit{ is live}\} \\ A_l(u \to v) &= \cup_{P \in P_l(S_0 \to u), P \not\ni v} \{P \textit{ is live and edge } (u,v) \textit{ is live}\} \\ A_{l,S}(v) &= \cup_{P \in \{P' \in P_l(S_0 \to v) : P' \not\ni w, \forall w \in S\}} \{P \textit{ is live}\}. \end{aligned}$$

In other words, $A_l(v)$ is the event that node v is influenced by live paths of length l, $A_l(u \to v)$ is the event that v is influenced by node u with live paths of length $l+1$, i.e., there exists a live path of length $l+1$ from a seed to v that ends with edge (u, v), and $A_{l,S}(v)$ is the event that node v is influenced by length l live paths which do not include any node in S. Note that for any pair of events (A, B), we use the notation AB for $A \cap B$.

Lemma 1. *For any $v \in V$,*

$$p(v) \leq 1 - \prod_{l=0}^{n-1} (1 - p_l(v)).$$

For any $v \in V$ and $l \in \{0, \dots, n-1\}$,

$$p_l(v) \leq 1 - \prod_{u \in N^-(v)} (1 - p_l(u \to v)).$$

Proof. Recall that $p(v) = \mathbb{P}(A(v))$, $p_l(v) = \mathbb{P}(A_l(v))$, and $p_l(u \to v) = \mathbb{P}(A_l(u \to v))$.

$$\begin{aligned} p(v) &= \mathbb{P}(\cup_{l=0}^{n-1} A_l(v)) \\ &= 1 - \mathbb{P}(\cap_{l=0}^{n-1} A_l(v)^C) \\ &\leq 1 - \prod_{l=0}^{n-1} \mathbb{P}(A_l(v)^C) \qquad (25) \\ &= 1 - \prod_{l=0}^{n-1} (1 - p_l(v)). \end{aligned}$$

Equation (25) follows from positive correlation among the events $A_l(v)^C$ for all $v \in V$, which can be proved by FKG inequality. Similarly,

$$\begin{aligned} p_l(v) &= \mathbb{P}(\cup_{u \in N^-(v)} A_l(u \to v)) \\ &= 1 - \mathbb{P}(\cap_{u \in N^-(v)} A_l(u \to v)^C) \\ &\leq 1 - \prod_{u \in N^-(v)} \mathbb{P}(A_l(u \to v)^C) \\ &= 1 - \prod_{u \in N^-(v)} (1 - p_l(u \to v)). \qquad (26) \end{aligned}$$

■

Theorem 2. *For any independent cascade model $IC(G,B)$ and a seed set $S_0 \subseteq V$,*

$$\sigma(S_0) \quad \leq \quad \sum_{v\in V}(1 - \prod_{l=0}^{n-1}(1 - \mathrm{UB}_l(v))) =: \sigma^+(S_0), \tag{27}$$

where $\mathrm{UB}_l(v)$ *is obtained recursively as in Definition 7.*

Proof. We provide proof by induction. The initial condition, for $l = 0$, can be easily checked. For every $s\in S_0$, $s^+\in N^+(s)$, $u\in V\setminus S_0$, and $v\in N^+(u)$,

$$\begin{aligned} p_0(s) = 1 &\leq \mathrm{UB}_0(s) = 1 \\ p_0(s\to s^+) = B_{ss^+} &\leq \mathrm{UB}_0(s\to s^+) = B_{ss^+} \\ p_0(u) = 0 &\leq \mathrm{UB}_0(u) = 0 \\ p_0(u\to v) = 0 &\leq \mathrm{UB}_0(u\to v) = 0. \end{aligned}$$

For each $l \leq L$, assume that $p_l(v) \leq \mathrm{UB}_l(v)$ and $p_l(u\to v) \leq \mathrm{UB}_l(u\to v)$ for all $u, v \in V$.

Since $p_l(s) = p_l(s \to s^+) = p_l(s^- \to s) = 0$ for every $l \geq 1$, $s \in S_0$, $s^+ \in N^+(s)$, and $s^-\in N^-(s)$, it is sufficient to show $p_{L+1}(v) \leq \mathrm{UB}_{L+1}(v)$ and $p_{L+1}(u\to v) \leq \mathrm{UB}_{L+1}(u\to v)$ for all $u\in V\setminus S_0$, and $v\in N^+(u)$,

For any $v \in V \setminus S_0$,

$$p_{L+1}(v) = \mathbb{P}(\cup_{u\in N^-(v)} E_{uv} A_{L,\{v\}}(u)),$$

where E_{uv} denotes the event that edge (u, v) is live, i.e., $\mathbb{P}(E_{uv}) = B_{uv}$. Thus,

$$\begin{aligned} p_{L+1}(v) &= 1 - \mathbb{P}(\cap_{u\in N^-(v)}(E_{uv}A_{L,\{v\}}(u))^C) \\ &\leq 1 - \prod_{u\in N^-(v)} (1 - \mathbb{P}(E_{uv}A_{L,\{v\}}(u))) \qquad (28) \\ &= 1 - \prod_{u\in N^-(v)} (1 - p_L(u\to v)) \\ &\leq 1 - \prod_{u\in N^-(v)} (1 - \mathrm{UB}_L(u\to v)) = \mathrm{UB}_{L+1}(v), \qquad (29) \end{aligned}$$

where Equation (28) is obtained by the positive correlation among the events $E_{uv}A_{L,\{v\}}(u)$, and Equation (29) comes from the assumption.

For any $v \in V \setminus S_0$ and $w \in N^+(v)$,

$$\begin{aligned} p_{L+1}(v\to w) &= \mathbb{P}(E_{vw}A_{L+1,\{w\}}(v)) \\ &= B_{vw}\mathbb{P}(A_{L+1,\{w\}}(v)). \qquad (30) \end{aligned}$$

Equation (30) follows from the independence between the events E_{vw} and $A_{L+1,\{w\}}(v)$.
If $w \in N^-(v)$,

$$\begin{aligned} p_{L+1}(v \rightarrow w) &= B_{vw}\mathbb{P}(\cup_{u\in N^-(v)\setminus\{w\}} E_{uv} A_{L,\{v,w\}}(u)) \\ &\leq B_{vw}\left(1 - \prod_{u\in N^-(v)\setminus\{w\}} (1-\mathbb{P}(E_{uv}A_{L,\{v,w\}}(u)))\right) \\ &\leq B_{vw}\left(1 - \prod_{u\in N^-(v)\setminus\{w\}} (1-p_L(u\rightarrow v))\right) \qquad (31) \\ &\leq B_{vw}\left(1 - \prod_{u\in N^-(v)\setminus\{w\}} (1-\mathrm{UB}_L(u\rightarrow v))\right), \end{aligned}$$

Equation (31) holds, since the two events satisfy $E_{uv}A_{L,\{v,w\}}(u) \subseteq E_{uv}A_{L,\{v\}}(u)$.
Recall that, if $w \in N^-(v)$,

$$\begin{aligned} \mathrm{UB}_{L+1}(v\rightarrow w) &= B_{vw}(1 - \frac{1-\mathrm{UB}_{L+1}(v)}{1-\mathrm{UB}_L(w\rightarrow v)}) \\ &= B_{vw}(1 - \prod_{u\in N^-(v)\setminus\{w\}} (1-\mathrm{UB}_L(u\rightarrow v))). \end{aligned}$$

Thus, $p_{L+1}(v\rightarrow w) \leq \mathrm{UB}_{L+1}(v\rightarrow w)$, for all $w \in N^+(v)\cap N^-(v)$.
If $w \notin N^-(v)$,

$$\begin{aligned} p_{L+1}(v\rightarrow w) &= B_{vw}\mathbb{P}(\cup_{u\in N^-(v)} E_{uv}A_{L,\{v,w\}}(u)) \\ &\leq B_{vw}\left(1-\prod_{u\in N^-(v)}(1-\mathrm{UB}_L(u\rightarrow v))\right) \\ &= B_{vw}\mathrm{UB}_{L+1}(v) = \mathrm{UB}_{L+1}(v\rightarrow w). \end{aligned}$$

Hence, $p_{L+1}(v\rightarrow w) \leq \mathrm{UB}_{L+1}(v\rightarrow w)$, for all $w \in N^+(v)$, concluding the proof.
Finally, by Lemma 1,

$$\begin{aligned} \sigma(S_0) &\leq \sum_{v\in V}(1-\prod_{l=0}^{n-1}(1-p_l(v))) \\ &\leq \sum_{v\in V}(1-\prod_{l=0}^{n-1}(1-\mathrm{UB}_l(v))) = \sigma^+(S_0). \end{aligned}$$

■

Appendix B. Proof of Theorem 3

Theorem 3. *For any independent cascade model $IC(G,B)$, the nonbacktracking upper bound $\sigma^+ : 2^V \rightarrow [0,|V|]$ is monotone and submodular.*

Recall that

$$\begin{aligned}
\sigma^+(S_0) &= \sum_{v\in V}(1-\prod_{l\in[n-1]}(1-\mathrm{UB}_l(v)))\\
\mathrm{UB}_l(u) &= 1-\prod_{w\in N^-(u)}(1-\mathrm{UB}_{l-1}(w\to u))\\
\mathrm{UB}_l(u\to v) &= B_{uv}\left(1-\prod_{w\in N^-(u)\setminus\{v\}}(1-\mathrm{UB}_{l-1}(w\to u))\right)
\end{aligned}$$

Since the summation of submodular functions is also submodular, it remains to prove that $(1-\prod_{l\in[n-1]}(1-\mathrm{UB}_l(v)))$ is submodular. Note that B_e and $N^-(v)$ for any $e\in E$ and $v\in V$ is independent of the seed set and that $\mathrm{UB}_0(u\to v)$ for any $u,v\in V$ is constant, so it is equivalent to prove the following statement.

Claim 1. *If $f_i(S)\in[0,1]$ for all $i\in[m]$ is monotone and submodular, then $1-\prod_{i\in[m]}(1-f_i(S))$ is also monotone and submodular.*

Proof. We provide a proof by induction. The initial condition holds for $k=1$.
Assume that $1-\prod_{i\in[k-1]}(1-f_i(S))$ is monotone (non decreasing) and submodular for some $k\le m$. Let $g_{k-1}(A)=1-\prod_{i\in[k-1]}(1-f_i(A))$. Then, for any $A\subseteq B\subseteq V$ and $s\in V$,

$$g_{k-1}(A\cup\{s\})-g_{k-1}(A)\ge g_{k-1}(B\cup\{s\})-g_{k-1}(B) \tag{32}$$

and

$$f_k(A\cup\{s\})-f_k(A)\ge f_k(B\cup\{s\})-f_k(B). \tag{33}$$

By definition,

$$g_k(A)=1-(1-g_{k-1}(A))(1-f_k(A)). \tag{34}$$

g_k is monotone as:

$$\begin{aligned}
g_k(A\cup\{s\}) &= 1-(1-g_{k-1}(A\cup\{s\}))(1-f_k(A\cup\{s\}))\\
&\ge 1-(1-g_{k-1}(A))(1-f_k(A))=g_k(A)
\end{aligned} \tag{35}$$

Next, to prove submodularity, we want to show:

$$\begin{aligned}
&(1-g_{k-1}(A))(1-f_k(A))-(1-g_{k-1}(A\cup\{s\}))(1-f_k(A\cup\{s\}))\\
&\ge(1-g_{k-1}(B))(1-f_k(B))-(1-g_{k-1}(B\cup\{s\}))(1-f_k(B\cup\{s\})).
\end{aligned} \tag{36}$$

To simplify, for any $i\in[m]$ and $S\subseteq V$, let $(1-g_i(S))=g'_i(S)$ and $(1-f_i(S))=f'_i(S)$. Then, Equation(32) and (33) become

$$\begin{aligned}
g'_{k-1}(A)-g'_{k-1}(A\cup\{s\}) &\ge g'_{k-1}(B)-g'_{k-1}(B\cup\{s\})\\
f'_k(A)-f'_k(A\cup\{s\}) &\ge f'_k(B)-f'_k(B\cup\{s\})
\end{aligned}$$

and Equation(36) is equivalent to

$$g'_{k-1}(A)f'_k(A) - g'_{k-1}(A\cup\{s\})f'_k(A\cup\{s\}) \geq g'_{k-1}(B)f'_k(B) - g'_{k-1}(B\cup\{s\})f'_k(B\cup\{s\}).$$

Let $g'_{k-1}(B) - g'_{k-1}(B\cup\{s\}) = \delta_g$ and $f'_k(B) - f'_k(B\cup\{s\}) = \delta_f$. Note that as f and g are non-decreasing, $\delta_g, \geq 0$ and $\delta_f \geq 0$. Then,

$$\begin{aligned}
&g'_{k-1}(A)f'_k(A) - g'_{k-1}(A\cup\{s\})f'_k(A\cup\{s\})\\
&\geq (g'_{k-1}(A\cup\{s\}) + \delta_g)(f'_k(A\cup\{s\}) + \delta_f) - g'_{k-1}(A\cup\{s\})f'_k(A\cup\{s\})\\
&= \delta_f\delta_g + \delta_g f'_k(A\cup\{s\}) + \delta_f g'_{k-1}(A\cup\{s\})\\
&\geq \delta_f\delta_g + \delta_g f'_k(B\cup\{s\}) + \delta_f g'_{k-1}(B\cup\{s\})\\
&= g'_{k-1}(B)f'_k(B) - g'_{k-1}(B\cup\{s\})f'_k(B\cup\{s\})
\end{aligned}$$

■

Appendix C. Proof of Theorem 4

To prove Theorem 4, we first introduce the following events in $IC(G, B)$ where $G = (V, E)$ and $|V| = n$.

Definition 14. *For any $v \in V,\ l \in \{0, \ldots, n-1\}$, we define*

$$\begin{aligned}
A(v) &= \{v \textit{ is influenced}\}\\
A(P) &= \{P \textit{ is live}\}\\
A_l(v) &= \cup_{P\in P_l(S_0\to v)}\{P \textit{ is live}\} = \cup_{P\in P_l(S_0\to v)}A(P).
\end{aligned}$$

For any $2 \leq r \leq n$ and $P \in P_{r-1}(v\to\cdot)$, let $A_l(v, P)$ be the event that node v is influenced by a live path of length $l-(r-1)$ that does not include any node in path P.

Next, we recall Definition 9 and Theorem 4.

Definition 9. *For all $l \in \{0, \ldots, n-1\}$, $u \in V$ and $P \in P_{r-1}(u\to\cdot)$, $\mathrm{UB}_l(u) \in [0, 1]$ and $\mathrm{UB}_l(u, P) \in [0, 1]$ are defined recursively as follows.*
Initial condition: *For every $l \leq r-1$ and $u \in V$,*

$$\mathrm{UB}_l(u) = f(\cup_{P\in P_l(S_0\to u)}(P, 1)).$$

For every $P \in P_{r-1}(s\to\cdot)$ and $s \in S_0$,

$$\mathrm{UB}_{r-1}(s, P) = 1.$$

Recursion: *For every $l \in \{r, \ldots, n-1\}$, $s \in S_0$, $u, x \in V\setminus S_0$, $v \in N^+(u)\setminus S_0$, and $Q \in P_{r-1}(v\to\cdot)$,*

$$\begin{aligned}
&\mathrm{UB}_l(s) = 0, \mathrm{UB}_l(s, P) = 0 \textit{ for any path } P\\
&\mathrm{UB}_l(v, Q) = 1 - \prod_{P\sim Q}(1 - B_{uv}\mathrm{UB}_{l-1}(u, P))\\
&\mathrm{UB}_l(x) = f(\cup_{P\in\{P'\in P_{r-1}(y\to x):\forall y\in V\setminus S_0\}}(P, \mathrm{UB}_l(y, P)))
\end{aligned}$$

Theorem 4. *For any independent cascade model $IC(G,B)$, a seed set $S_0 \subseteq V$ and an integer $r \geq 2$,*

$$\sigma(S_0) \quad \leq \quad \sum_{v\in V}(1 - \prod_{l=0}^{n-1}(1 - \mathrm{UB}_l(v))) =: \sigma_r^+(S_0) \leq \sigma^+(S_0), \tag{37}$$

where $\mathrm{UB}_l(v)$ is obtained recursively as in Definition 9.

Proof. First, let $p(v) = \mathbb{P}(A(v))$, $p(P) = \mathbb{P}(A(P))$, $p_l(v) = \mathbb{P}(A_l(v))$, and $p_l(v,P) = \mathbb{P}(A_l(v,P))$. By Lemma 1, it is sufficient to show $p_l(v) \leq \mathrm{UB}_l(v)$ for all $v \in V$ and $l \in \{0, 1, \ldots, n-1\}$. We provide proof by induction on l.

The initial condition can be easily checked. For every $l \leq r-1$, $u \in V$, $v \in N^+(u)$ and $s \in S_0$,

$$p_l(u) = \mathbb{P}(\cup_{P\in P_l(S_0 \to u)} A(P)) \leq f(\cup_{P\in P_l(S_0 \to u)}(P,1)) = \mathrm{UB}_l(u).$$

For every $P \in P_{r-1}(s \to \cdot)$ and $s \in S_0$,

$$p_{r-1}(s,P) = 1 \leq \mathrm{UB}_{r-1}(s,P) = 1.$$

Next, for each $l \leq L$, assume that $p_l(v) \leq \mathrm{UB}_l(v)$ and $p_l(v,P) \leq \mathrm{UB}_l(v,P)$ for every $v \in V$ and $P \in P_{r-1}(v \to \cdot)$. For every $l \geq r$, $s \in S_0$ and $P \in P_{r-1}(s \to \cdot)$, $p_l(s) = 0$ and $p_l(s,P) = 0$, since a seed cannot be influenced again by other seeds and cannot influence other nodes after the first step $l = 0$. Therefore, to conclude the proof, it is sufficient to show $p_{L+1}(v) \leq \mathrm{UB}_{L+1}(v)$ and $p_{L+1}(v,P) \leq \mathrm{UB}_{L+1}(v,P)$ for every $v \in V \setminus S_0$ and $P \in P_{r-1}(v \to \cdot)$.

For any $u, v \in V \setminus S_0$, $P \in P_{r-1}(u \to \cdot)$, and $Q \in P_{r-1}(v \to \cdot)$ such that $P \sim Q$, let $P = (u = u_1, \ldots, u_k)$ and $Q = (v = v_1, \ldots, v_k)$. Recall that if $P \sim Q$, $(u, v, u_3 = v_2 \ldots, u_k = v_{k-1}, v_k)$ is a path of length r. Then, for any $P \sim Q$, $A_L(u,P)$ contains all events that an in-neighbor u of node v is influenced by a live path that does not contain any node in path $P = (u, v, v_2, \ldots, v_{k-1})$. Notice that the event $A_{L+1}(v,Q)$ is the union of events that an in-neighbor u of node v is influenced by a live path that does not contain any node in path $Q = (v, v_2, \ldots, v_k)$ and edge (u,v) is live. In other words, $A_{L+1}(v,Q) = \cup_{u\in N^-(v)} E_{uv} A_L(u, P \cup Q)$, where $P \cup Q = (u, v, u_3 = v_2 \ldots, u_k = v_{k-1}, v_k)$ and E_{uv} is the event that edge (u,v) is live. Thus, $A_{L+1}(v,Q) \subseteq \cup_{P\sim Q} E_{uv} A_L(u,P)$, which implies

$$p_{L+1}(v,Q) \quad \leq \quad \mathbb{P}(\cup_{P\sim Q} E_{uv} A_L(u,P)).$$

By the positive correlation among the events $E_{uv} A_L(u,P)$,

$$\begin{aligned} p_{L+1}(v,Q) \quad &\leq \quad 1 - \prod_{P\sim Q}(1 - P_{uv}\mathbb{P}(A_L(u,P))) \\ &\leq \quad 1 - \prod_{P\sim Q}(1 - P_{uv}\mathrm{UB}_L(u,P)) = \mathrm{UB}_{L+1}(v,Q), \end{aligned} \tag{38}$$

where Equation (38) follows from the assumption.

For any $x \in V \setminus S_0$,

$$\begin{aligned} p_{L+1}(x) &= \mathbb{P}(\cup_{P \in P_{L+1}(S_0 \to x)} A(P)) \\ &\leq \mathbb{P}(\cup_{P \in \{P' \in P_{r-1}(y \to x): \forall y \in V \setminus S_0\}} A_{L+1}(y, P) A(P)) \qquad (39) \\ &\leq f(\cup_{P \in \{P' \in P_{r-1}(y \to x): \forall y \in V \setminus S_0\}} (P, p_{L+1}(y, P))) \qquad (40) \\ &\leq f(\cup_{P \in \{P' \in P_{r-1}(y \to x): \forall y \in V \setminus S_0\}} (P, \mathrm{UB}_{L+1}(y, P))) \qquad (41) \\ &= \mathrm{UB}_{L+1}(x). \end{aligned}$$

Equation (39) follows from $\cup_{P \in P_{L+1}(S_0 \to x)} A(P) \subseteq \cup_{P \in \{P' \in P_{r-1}(y \to x): \forall y \in V \setminus S_0\}} A_{L+1}(y, P)$ since the right hand side also includes events that y is influenced by a walk rather than a path. Equation (40) holds because the function f computes the upper bound rather than the exact probability of the input events, and Equation (41) comes from the assumption.

Therefore, $p_{L+1}(v, Q) \leq \mathrm{UB}_{L+1}(v, Q)$ and $p_{L+1}(x) \leq \mathrm{UB}_{L+1}(x)$, concluding the proof by induction. Then, by Lemma 1,

$$\begin{aligned} \sigma(S_0) &\leq \sum_{v \in V} (1 - \prod_{l=0}^{n-1} (1 - p_l(v))) \\ &\leq \sum_{v \in V} (1 - \prod_{l=0}^{n-1} (1 - \mathrm{UB}_l(v))) = \sigma_r^+(S_0). \end{aligned}$$

■

Appendix D. Proof of Theorem 5

Theorem 5. *For any independent cascade model $IC(G, B)$ and a seed set S_0 and their MDAS $IC(G', B')$,*

$$\sigma(S_0) \geq \sum_{v_k \in V'} \mathrm{LB}(v_k) =: \sigma^-(S_0), \qquad (42)$$

where $\mathrm{LB}(v_k)$ is obtained recursively as in Definition 12.

Proof. We provide proof by induction. For any $v_k \in V'$, let $A(v_k)$ be the event that node v_k is influenced in MDAS $IC(G', B')$, and for every edge (v_j, v_k), let E_{v_j, v_k} be the event that edge (v_j, v_k) is live, i.e., $\mathbb{P}(E_{v_j, v_k}) = B'_{v_j v_k}$. Recall that $p(v_k) = \mathbb{P}(A(v_k))$.

The initial condition $k = 1$ holds, since $p(v_1) = 1 \geq \mathrm{LB}(v_1) = 1$ (v_1 is a seed). Now, for every $k \leq K$, assume $p(v_k) \geq \mathrm{LB}(v_k)$.

Then, for node v_{K+1},

$$p(v_{K+1}) = \mathbb{P}(\cup_{v_j \in N^-(v_{K+1})} E_{v_j v_{K+1}} A(v_j)).$$

We re-label vertices in $N^-(v_{K+1}) = \{u_1, \dots, u_{m(K+1)}\}$ where $m(K+1) = \text{in-deg}(v_{K+1})$, and let $\mathcal{Q}_{iK+1} = B'_{u_i v_{K+1}}$. Then, for any integer $m \leq m(K+1)$,

$$\begin{aligned}
p(v_{K+1}) &= \mathbb{P}(\cup_{i=1}^{m(K+1)} E_{u_i v_{K+1}} A(u_i)) \\
&\geq \mathbb{P}(\cup_{i=1}^{m} E_{u_i v_{K+1}} A(u_i)) \\
&\geq \sum_{i=1}^{m} \mathbb{P}(E_{u_i v_{K+1}} A(u_i)) - \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathbb{P}(E_{u_i v_{K+1}} A(u_i) E_{u_j v_{K+1}} A(u_j)) \qquad (43) \\
&= \sum_{i=1}^{m} \mathcal{Q}_{iK+1} \mathbb{P}(A(u_i)) - \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathcal{Q}_{iK+1} \mathcal{Q}_{jK+1} \mathbb{P}(A(u_i) A(u_j)) \qquad (44) \\
&\geq \sum_{i=1}^{m} \mathcal{Q}_{iK+1} \mathbb{P}(A(u_i))(1 - \sum_{j=1}^{i-1} \mathcal{Q}_{jK+1}). \qquad (45)
\end{aligned}$$

Equation (43) follows from the principle of inclusion and exclusion. Equation (44) results from the Independence between the event that an edge ending with v_{K+1} is live and the event that a node v_i is influenced where $i < K+1$. Equation (45) holds since $\mathbb{P}(A(u_i)) \geq \mathbb{P}(A(u_i)A(u_j))$.

Now, define $m^* = \max\{m' \leq m(K+1) : \sum_{j=1}^{m'-1} \mathcal{Q}_{jK+1} \leq 1\}$. Then,

$$\begin{aligned}
p(v_{K+1}) &\geq \sum_{i=1}^{m^*} \mathcal{Q}_{iK+1} \mathbb{P}(A(u_i))(1 - \sum_{j=1}^{i-1} \mathcal{Q}_{jK+1}) \\
&\geq \sum_{i=1}^{m^*} \mathcal{Q}_{iK+1} \text{LB}(u_i)(1 - \sum_{j=1}^{i-1} \mathcal{Q}_{jK+1}) \qquad (46) \\
&= \text{LB}(v_{K+1}).
\end{aligned}$$

Equation (46) follows since $1 - \sum_{j=1}^{i-1} \mathcal{Q}_{jK+1} \geq 0$ for all $i \leq m^*$ by the definition of m^*. Thus, $p(v_i) \geq \text{LB}(v_i)$ for all $v_i \in V'$, concluding the induction proof.
Finally,

$$\begin{aligned}
\sigma(S_0) &\geq \sum_{i=1}^{n} p(v_i) \qquad (47) \\
&\geq \sum_{i=1}^{n} \text{LB}(v_i) = \sigma^-(S_0).
\end{aligned}$$

Equation (47) holds since its right hand side equals to the influence of the subnetwork, $IC(G', B')$. ■

References

Emmanuel Abbe and Colin Sandon. Detection in the stochastic block model with multiple clusters: proof of the achievability conjectures, acyclic bp, and the information-computation gap. *arXiv preprint arXiv:1512.09080*, 2015.

Emmanuel Abbe, Sanjeev Kulkarni, and Eun Jee Lee. Nonbacktracking bounds on the influence in independent cascade models. In *Advances in Neural Information Processing Systems 30*, pages 1407–1416. 2017.

Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1347–1357. IEEE, 2015.

Wei Chen and Shang-Hua Teng. Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality. In *Proceedings of the 26th International Conference on World Wide Web*, pages 967–976. International World Wide Web Conferences Steering Committee, 2017.

Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

Moez Draief, Ayalvadi Ganesh, and Laurent Massoulié. Thresholds for virus spread on networks. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 51. ACM, 2006.

Cees M Fortuin, Pieter W Kasteleyn, and Jean Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971.

Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, pages 1420–1443, 1978.

Brian Karrer and Mark EJ Newman. Message passing approach for general epidemic models. *Physical Review E*, 82(1):016101, 2010.

Brian Karrer, MEJ Newman, and Lenka Zdeborová. Percolation on sparse networks. *Physical review letters*, 113(20):208702, 2014.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

Abdelmajid Khelil, Christian Becker, Jing Tian, and Kurt Rothermel. An epidemic model for information diffusion in manets. In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 54–60. ACM, 2002.

Justin T Khim, Varun Jog, and Po-Ling Loh. Computing and maximizing influence in linear threshold and triggering models. In *Advances in Neural Information Processing Systems*, pages 4538–4546, 2016.

Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013.

Eun Jee Lee, S. Kamath, E. Abbe, and S. R. Kulkarni. Spectral bounds for independent cascade model with sensitive edges. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 649–653, March 2016. doi: 10.1109/CISS.2016.7460579.

Remi Lemonnier, Kevin Scaman, and Nicolas Vayatis. Tight bounds for influence in diffusion networks and application to bond percolation and epidemiology. In *Advances in Neural Information Processing Systems*, pages 846–854, 2014.

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

Dunia Lopez-Pintado and Duncan J Watts. Social influence, binary decisions and collective dynamics. *Rationality and Society*, 20(4):399–443, 2008.

Boris Shulgin, Lewi Stone, and Zvia Agur. Pulse vaccination strategy in the sir epidemic model. *Bulletin of Mathematical Biology*, 60(6):1123–1148, 1998.

Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25 (3):545–576, 2012.

Duncan J Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

Jiang Yang and Scott Counts. Predicting the speed, scale, and range of information diffusion in twitter. 2010.