# Influence Maximization Meets Efficiency and Effectiveness: A Hop-Based Approach

Jing Tang
Interdisciplinary Graduate School
Nanyang Technological University
Email: tang0311@ntu.edu.sg

Xueyan Tang
School of Comp. Sci. & Eng.
Nanyang Technological University
Email: asxytang@ntu.edu.sg

Junsong Yuan
School of EEE
Nanyang Technological University
Email: jsyuan@ntu.edu.sg

***Abstract*—Influence Maximization is an extensively-studied problem that targets at selecting a set of initial seed nodes in the Online Social Networks (OSNs) to spread the influence as widely as possible. However, it remains an open challenge to design fast and accurate algorithms to find solutions in large-scale OSNs. Prior Monte-Carlo-simulation-based methods are slow and not scalable, while other heuristic algorithms do not have any theoretical guarantee and they have been shown to produce poor solutions for quite some cases. In this paper, we propose hop-based algorithms that can easily scale to millions of nodes and billions of edges. Unlike previous heuristics, our proposed hop-based approaches can provide certain theoretical guarantees. Experimental evaluations with real OSN datasets demonstrate the efficiency and effectiveness of our algorithms.**

***Index Terms*—Online social networks, influence maximization, hop-based influence estimation.**

## I. Introduction

Information can be disseminated widely and rapidly through Online Social Networks (OSNs) with "word-of-mouth" effects. Viral marketing is such a typical application in which new products or activities are advertised by some influential users in the OSN to other users in a cascading manner [12]. A large amount of recent work [3], [6], [7], [8], [9], [10], [15], [16], [20], [25], [26], [27], [29], [30], [32] has been focusing on *influence maximization* in viral marketing, which targets at selecting a set of initial seed nodes in the OSN to spread the influence as widely as possible. The influence maximization problem was formulated by [16] with two basic diffusion models, namely the *Independent Cascade* (IC) and *Linear Threshold* (LT) models. Although finding the optimal seed set is NP-hard [16], a simple greedy hill-climbing algorithm has a $(1-1/e)$-approximation guarantee due to the submodularity and monotone properties of the influence spread under these models [24]. Follow-up studies have mostly concentrated on efficient implementation of the hill-climbing algorithm for large-scale OSNs. The key difference among various methods lies in how to estimate the influence spread of a seed set.

Computing the exact influence spread on general graphs is #P-hard for both the IC and LT models [6], [8]. Thus, some Monte-Carlo-simulation-based methods [16], [20], [27], [32] estimate the influence spread by reachability tests, while some *reverse influence sampling* methods carry out seed selection [3], [25], [26], [29], [30] by leveraging the concept of reverse reachability. These sampling-based methods can provide theoretical guarantees up to $(1-1/e-\epsilon)$-approximation. However, the sampling-based methods can encounter the efficiency problem even for the dramatically improved versions [3], [25], [26], [27], [29], [30] as they may consume a lot of time/memory to obtain/store one sample. Other heuristic methods [6], [7], [8], [9], [15] conduct rough estimation of the influence spread either by exploiting some related features (such as node degrees) or extracting subgraphs where the influence spread is easier to compute. However, these heuristics do not have any theoretical guarantee and they have been shown to suffer from the effectiveness problem.

To achieve both efficiency and effectiveness, in this paper, we propose a new hop-based approach for the influence maximization problem. Although it belongs to heuristics, unlike other heuristic methods, our method can provide certain theoretical guarantees. As shall be shown by the experimental results, our hop-based methods outperform the existing heuristics and perform as well as the sampling-based methods in terms of the influence spread produced. Meanwhile, our hop-based methods run much faster than the sampling-based methods. For a large OSN with billions of edges, only our hop-based methods and some ineffective heuristics can work with acceptable time and memory usage for various distributions of propagation probabilities. Our contributions are summarized as follows.

1) We propose hop-based influence estimation algorithms for efficiently selecting seed nodes to maximize the influence spread.
2) We develop an upper bounding approach on the influence generated by a seed node to further speed up seed selection.
3) We carry out theoretical analysis for our hop-based algorithms and derive approximation guarantees.
4) We conduct extensive experiments with several real OSN datasets. The results demonstrate the efficiency and effectiveness of our hop-based algorithms.

The rest of this paper is organized as follows. Section II introduces the influence maximization problem and the greedy hill-climbing algorithm. Section III elaborates our algorithm design and analyzes the theoretical guarantees. Section IV presents the experimental study. Section V reviews the related work. Finally, Section VI concludes the paper.

## II. Preliminaries

### A. Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph modeling an OSN, where the nodes $\mathcal{V}$ represent users and the edges $\mathcal{E}$ represent the connections among users (e.g., followships on Twitter). For each directed edge $(u, v) \in \mathcal{E}$, we refer to $v$ as a *neighbor* of $u$, and refer to $u$ as an *inverse neighbor* of $v$.

To facilitate the exposition, we shall mainly focus on the Independent Cascade (IC) model — a representative and most widely-studied diffusion model for influence propagation [3], [6], [7], [15], [16], [20], [25], [26], [27], [29], [30], [32]. Extending our hop-based methods to other diffusion models shall be discussed later. In the IC model, a propagation probability $p_{u,v}$ is associated with each edge $(u, v)$, representing the probability for $v$ to be activated by $u$ through the edge. Let $\mathcal{N}_u$ denote the set of node $u$'s neighbors, i.e., $\mathcal{N}_u = \{v : v \in \mathcal{V}, (u, v) \in \mathcal{E}\}$. Given a set of seed nodes $\mathcal{S}$, the IC diffusion process proceeds as follows. Initially, the seed nodes $\mathcal{S}$ are activated, while all the other nodes are not activated. When a node $u$ first becomes activated, it attempts to further activate its neighbors who are not yet activated. For each such neighbor $v \in \mathcal{N}_u$, $v$ would become activated with probability $p_{u,v}$. This process repeats until no more node can be activated. The *influence spread* of the seed set $\mathcal{S}$, denoted by $\sigma(\mathcal{S})$, is the expected number of nodes activated by the above process. The *influence maximization* problem [16] is to find a set $\mathcal{S}$ of $k$ nodes to maximize $\sigma(\mathcal{S})$, where $k$ is a given parameter. Formally,

$$\max_{|\mathcal{S}|=k} \sigma(\mathcal{S}). \tag{1}$$

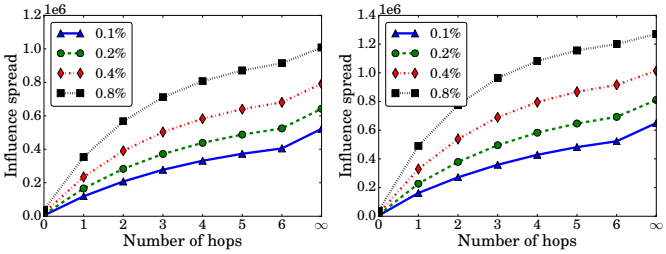
### B. Greedy Heuristic

The influence function $\sigma(\cdot)$ has been proved to be submodular and monotone under the IC model [16]. Thus, a simple greedy hill-climbing algorithm that provides $(1-1/e)$-approximation [24] was proposed for influence maximization as described in Algorithm 1. It starts with an empty seed set $\mathcal{S} = \emptyset$. In each iteration, the greedy heuristic chooses a new seed $u$ from the non-seed nodes $\mathcal{V} \setminus \mathcal{S}$ with largest marginal influence gain $\sigma(\mathcal{S} \cup \{u\}) - \sigma(\mathcal{S})$ and adds $u$ to $\mathcal{S}$. The algorithm stops after selecting $k$ seeds.

**Algorithm 1:** $\boldsymbol{Greedy}(\mathcal{G}, \sigma)$

```
initialize S ← ∅;
while the size of S is smaller than k do
    find u ← arg max_{v∈V\S} {σ(S ∪ {v}) − σ(S)};
    S ← S ∪ {u};
return S;
```

A CELF technique [20] can be used to enhance the efficiency of the greedy algorithm due to the submodularity of influence spread. Specifically, it may not be necessary to evaluate the marginal influence gain for every node of $\mathcal{V} \setminus \mathcal{S}$ in each iteration. Due to the submodularity, the marginal gains can only decrease over iterations. Thus, the marginal gains obtained in the previous iterations can be used as upper bounds for a new iteration. In the new iteration, the nodes can be evaluated in decreasing order of these upper bounds. Once the largest marginal gain evaluated is greater than the upper bound of the next node to evaluate, the evaluation can stop as none of the remaining nodes would be able to produce larger marginal gain.




(a) High degree nodes (b) Top influential nodes

Fig. 1. Influence spread for different hops of propagation on LiveJournal.

## III. Hop-Based Approaches

### A. Hop-Based Influence Estimation under IC Model

We start by studying some empirical results of influence spread tested on a real OSN graph – LiveJournal (5M nodes and 69M edges) [21]. Fig. 1 plots the influence spread within different numbers of hops of propagation (the data points of $\infty$ hops represent the actual influence spread without any hop limit, and the seed set size is represented as a percentage of the entire node set in the OSN). We test two typical seed sets for influence maximization: selecting the nodes with highest degrees and selecting the top influential nodes using a greedy hill-climbing heuristic [16] as described in Section II-B. The propagation probability on each edge $(u, v)$ in the OSN is set to the reciprocal of $v$'s in-degree as widely adopted in previous studies [7], [16], [26], [29]. As seen from Fig. 1, the increase in the influence spread for considering each additional hop of propagation generally decreases with increasing number of hops. The majority of influence spread is produced within the first few hops of propagation. Similar trends have also been observed by several measurement-driven studies on real OSNs [4], [13], [19]. For example, Goel *et al.* [13] showed that less than 10% of the cascades in the diffusion are more than 2 hops away from the seed. These observations motivate us to design hop-based algorithms to efficiently capture the major influence propagation, especially for the first two hops of propagation.

A hop-based algorithm focuses on the influence propagation up to a given number of $h$ hops starting from the initial seed set. For $h = 1$ and $h = 2$, we can efficiently calculate the *exact* influence spread within $h$ hops of propagation and maintain it incrementally when the seed set expands. Corresponding to the denotation of a node $v$'s neighbors as $\mathcal{N}_v$, let $\mathcal{I}_v$ denote $v$'s inverse neighbors, i.e., $\mathcal{I}_v = \{w : w \in \mathcal{V}, (w, v) \in \mathcal{E}\}$. Let $\pi_h^{\mathcal{S}}(v)$ denote the probability for a node $v$ to be activated within $h$ hops of propagation from a seed set $\mathcal{S}$, and let

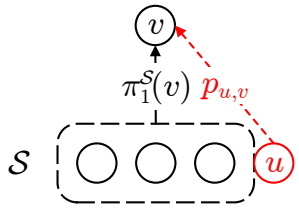



Fig. 2. The effect on $v \in \mathcal{N}_u$ by adding $u$ to the seed set $\mathcal{S}$.

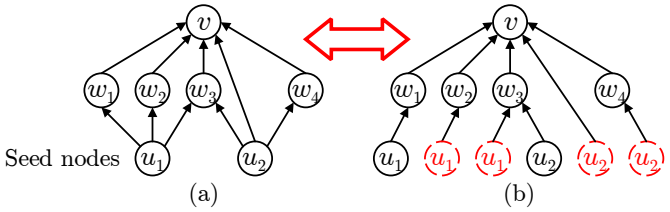



Fig. 3. An example of how a non-seed node $v$ is activated.

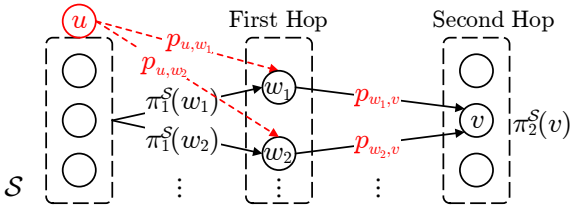



Fig. 4. The effect on $v \in \mathcal{N}_u^2$ by adding $u$ to the seed set $\mathcal{S}$.

$\sigma_h(\mathcal{S})$ denote the influence spread produced within $h$ hops of propagation from $\mathcal{S}$.

**One Hop of Propagation:** We first model one hop of propagation. Obviously, for all the seed nodes $v \in \mathcal{S}$, $\pi_1^{\mathcal{S}}(v) = 1$. With one hop of propagation, for all the non-seed nodes $v \notin \mathcal{S}$, $v$ can only be activated directly by its inverse neighbors $\mathcal{I}_v$ who are seed nodes in $\mathcal{S}$. Since each of such $v$'s inverse neighbors activates $v$ independently, the probability for all of them to fail to activate $v$ is $\prod_{w \in \mathcal{I}_v \cap \mathcal{S}}(1-p_{w,v})$. Consequently, the probability for $v$ to be activated is $1-\prod_{w \in \mathcal{I}_v \cap \mathcal{S}}(1-p_{w,v})$. Thus, for any node $v \in \mathcal{V}$, its one-hop activation probability is given by

$$\pi_1^{\mathcal{S}}(v) = \begin{cases} 1, & \text{if } v \in \mathcal{S}, \\ 1 - \prod_{w \in \mathcal{I}_v \cap \mathcal{S}}(1 - p_{w,v}), & \text{otherwise.} \end{cases} \tag{2}$$

Now, we show how to maintain $\pi_1^{\mathcal{S}}(v)$ when the seed set changes. Suppose that $\pi_1^{\mathcal{S}}(v)$ is known for every node $v \in \mathcal{V}$. If a new seed node $u$ is added to a seed set $\mathcal{S}$, it is clear that the activation probability of the new seed becomes 1, i.e., $\pi_1^{\mathcal{S}\cup\{u\}}(u) = 1$. In addition to $u$, only the one-hop activation probabilities of its neighbors are affected. So, we can incrementally update $\pi_1^{\mathcal{S}\cup\{u\}}(v)$ based on $\pi_1^{\mathcal{S}}(v)$ for each node $v \in \mathcal{N}_u$ (see Fig. 2). The new activation probability $\pi_1^{\mathcal{S}\cup\{u\}}(v)$ is given by

$$\begin{aligned} \pi_1^{\mathcal{S}\cup\{u\}}(v) &= 1 - \prod_{w \in (\mathcal{I}_v \cap \mathcal{S}) \cup \{u\}} (1 - p_{w,v}) \\ &= 1 - \big(1 - \pi_1^{\mathcal{S}}(v)\big) \cdot (1 - p_{u,v}). \end{aligned} \tag{3}$$

Algorithm 2 calculates the increment of one-hop influence spread $\sigma_1(\mathcal{S} \cup \{u\}) - \sigma_1(\mathcal{S})$ efficiently by maintaining $\pi_1^{\mathcal{S}}(v)$ for every node $v$ based on the above equation.

**Algorithm 2:** ***OneHopIncrement***$(\mathcal{G}, \mathcal{S}, u)$

1 $\pi_1^{\mathcal{S}\cup\{u\}}(u) \leftarrow 1$;
2 **for** *each node* $v \in \mathcal{N}_u \setminus \mathcal{S}$ **do**
3 $\quad \pi_1^{\mathcal{S}\cup\{u\}}(v) \leftarrow 1 - \big(1 - \pi_1^{\mathcal{S}}(v)\big) \cdot (1 - p_{u,v})$;
4 **return** $\sum_{v \in \{u\} \cup (\mathcal{N}_u \setminus \mathcal{S})} \big(\pi_1^{\mathcal{S}\cup\{u\}}(v) - \pi_1^{\mathcal{S}}(v)\big)$;

**Two Hops of Propagation:** To better approximate $\sigma(\mathcal{S})$, we next model two hops of propagation. As illustrated in Fig. 3(a), with two hops of propagation, a non-seed node $v$ may be activated directly by a seed node $u_i$ or indirectly via a neighbor $w_j$ of a seed node $u_i$. In the former case, the probability for $v$ to be activated by $u_i$ is $p_{u_i,v}$, which can be rewritten as $p_{u_i,v} \cdot \pi_1^{\mathcal{S}}(u_i)$ since $\pi_1^{\mathcal{S}}(u_i) = 1$. In the latter case, the probability for $v$ to be activated by $w_j$ is $p_{w_j,v} \cdot \pi_1^{\mathcal{S}}(w_j)$. Since the activation probability of each seed node $u_i \in \mathcal{S}$ is 1, Fig. 3(a) is equivalent to Fig. 3(b) in which $v$ is activated independently by all of its inverse neighbors. As a result, the probability for $v$ to be activated is given by $1 - \prod_{w \in \mathcal{I}_v}(1 - p_{w,v} \cdot \pi_1^{\mathcal{S}}(w))$. Thus, for any node $v \in \mathcal{V}$, its two-hop activation probability is given by

$$\pi_2^{\mathcal{S}}(v) = \begin{cases} 1, & \text{if } v \in \mathcal{S}, \\ 1 - \prod_{w \in \mathcal{I}_v} \big(1 - p_{w,v} \cdot \pi_1^{\mathcal{S}}(w)\big), & \text{otherwise.} \end{cases} \tag{4}$$

According to the above equation, we can obtain $\pi_2^{\mathcal{S}}(v)$ based on the $\pi_1^{\mathcal{S}}(w)$'s of its inverse neighbors. When a new seed node $u$ is added, only the nodes within two hops of $u$ are affected. We denote these nodes by $\mathcal{N}_u^2 = \mathcal{N}_u \cup \big(\bigcup_{w \in \mathcal{N}_u} \mathcal{N}_w\big) \setminus \{u\}$. Fig. 4 illustrates the effect of adding $u$ to $\mathcal{S}$ on the activation of a node $v$ in $\mathcal{N}_u^2$. Due to the independence among all two-hop activation paths as discussed above, we can incrementally update the two-hop activation probability by considering the outgoing edges from $u$ one at a time.

*Theorem 1:* The new two-hop activation probability $\pi_2^{\mathcal{S}\cup\{u\}}(v)$ after adding a seed $u$ to $\mathcal{S}$ can be computed by

$$\pi_2^{\mathcal{S}\cup\{u\}}(v) = 1 - \big(1 - \pi_2^{\mathcal{S}}(v)\big) \cdot \prod_{w \in (\mathcal{M}_{u,v} \cup \{u\})} \frac{1 - p_{w,v} \cdot \pi_1^{\mathcal{S}\cup\{u\}}(w)}{1 - p_{w,v} \cdot \pi_1^{\mathcal{S}}(w)}, \tag{5}$$

where $\mathcal{M}_{u,v}$ denotes the set of intermediate nodes connecting $u$ and $v$, i.e., $\mathcal{M}_{u,v} = \{w : (u,w) \in \mathcal{E} \text{ and } (w,v) \in \mathcal{E}\}$.

Due to space limitations, we leave the formal proofs of all theoretical results to the extended version of this paper [28].

Algorithm 3 performs the updates on the activation probabilities of the nodes within two hops of $u$ when $u$ is added as a new seed to $\mathcal{S}$. Lines 1–2 set the one-hop and two-hop activation probabilities of the new seed node $u$ to 1. Lines 3–4 initialize $\pi_2^{\mathcal{S}\cup\{u\}}(v)$ for all the nodes within two hops of

**Algorithm 3:** $TwoHopsIncrement(\mathcal{G}, \mathcal{S}, u)$

1 $\pi_1^{\mathcal{S}\cup\{u\}}(u) \leftarrow 1$;
2 $\pi_2^{\mathcal{S}\cup\{u\}}(u) \leftarrow 1$;
3 **for** *each node* $v \in \mathcal{N}_u^2 \setminus \mathcal{S}$ **do**
4 $\quad \pi_2^{\mathcal{S}\cup\{u\}}(v) \leftarrow \pi_2^{\mathcal{S}}(v)$;
5 **for** *each node* $w \in \mathcal{N}_u \setminus \mathcal{S}$ **do**
6 $\quad \pi_1^{\mathcal{S}\cup\{u\}}(w) \leftarrow 1 - \big(1 - \pi_1^{\mathcal{S}}(w)\big) \cdot (1 - p_{u,w})$;
7 $\quad \pi_2^{\mathcal{S}\cup\{u\}}(w) \leftarrow 1 - \big(1 - \pi_2^{\mathcal{S}\cup\{u\}}(w)\big) \cdot \frac{1 - p_{u,w} \cdot \pi_1^{\mathcal{S}\cup\{u\}}(u)}{1 - p_{u,w} \cdot \pi_1^{\mathcal{S}}(u)}$;
8 $\quad$ **for** *each node* $v \in \mathcal{N}_w \setminus \mathcal{S}$ **do**
9 $\quad\quad \pi_2^{\mathcal{S}\cup\{u\}}(v) \leftarrow 1 - \big(1 - \pi_2^{\mathcal{S}\cup\{u\}}(v)\big) \cdot \frac{1 - p_{w,v} \cdot \pi_1^{\mathcal{S}\cup\{u\}}(w)}{1 - p_{w,v} \cdot \pi_1^{\mathcal{S}}(w)}$;
10 **return** $\sum_{v \in \{u\} \cup (\mathcal{N}_u^2 \setminus \mathcal{S})} \big(\pi_2^{\mathcal{S}\cup\{u\}}(v) - \pi_2^{\mathcal{S}}(v)\big)$;

$u$. Line 6 computes the new one-hop activation probabilities for all of $u$'s neighbors as explained earlier. For each node $v \in \mathcal{N}_u^2 \setminus \mathcal{S}$, lines 7–9 calculate the new two-hop activation probability $\pi_2^{\mathcal{S}\cup\{u\}}(v)$ in an iterative manner according to Theorem 1. In this way, we can save a huge amount of space for storing the intermediate nodes $\mathcal{M}_{u,v}$ for every pair of nodes $u$ and $v$. Finally, the algorithm returns the total increment of two-hop influence spread $\sigma_2(\mathcal{S} \cup \{u\}) - \sigma_2(\mathcal{S})$.

### B. Complexity

We shall refer to the greedy heuristic (Algorithm 1) as the OneHop and TwoHop algorithms respectively when the influence spread is approximated by one hop and two hops of propagation.

**Time Complexity:** The time complexity of Algorithm 2 is $O(1 + |\mathcal{N}_u|)$. Thus, the time complexity of selecting one seed in the OneHop algorithm is $O\big(\sum_{u \in \mathcal{V}}(1 + |\mathcal{N}_u|)\big) = O\big(|\mathcal{V}| + |\mathcal{E}|\big)$. Therefore, the total time complexity of OneHop is $O\big(k(|\mathcal{V}| + |\mathcal{E}|)\big)$. The time complexity of Algorithm 3 is $O\big(1 + |\mathcal{N}_u| + \sum_{w \in \mathcal{N}_u} |\mathcal{N}_w|\big)$. Therefore, the time complexity of selecting one seed in the TwoHop algorithm is $O\big(\sum_{u \in \mathcal{V}}(1 + |\mathcal{N}_u| + \sum_{w \in \mathcal{N}_u} |\mathcal{N}_w|)\big) = O\big(|\mathcal{V}| + |\mathcal{E}| + \sum_{u \in \mathcal{V}} \sum_{w \in \mathcal{N}_u} |\mathcal{N}_w|\big) = O\big(|\mathcal{V}| + |\mathcal{E}| + \sum_{w \in \mathcal{V}}(|\mathcal{I}_w| \cdot |\mathcal{N}_w|)\big)$. Thus, the total time complexity of TwoHop is $O\Big(k\big(|\mathcal{V}| + |\mathcal{E}| + \sum_{w \in \mathcal{V}}(|\mathcal{I}_w| \cdot |\mathcal{N}_w|)\big)\Big)$.

**Space Complexity:** Besides the space used to store the graph, the OneHop algorithm only requires $O(|\mathcal{V}|)$ space to store the one-hop activation probability $\pi_1^{\mathcal{S}}(v)$ for every $v \in \mathcal{V}$ before and after a new seed is added. Similarly, the TwoHop algorithm requires $O(|\mathcal{V}|)$ space to store the one-hop and two-hop activation probabilities $\pi_1^{\mathcal{S}}(v)$ and $\pi_2^{\mathcal{S}}(v)$ for computing the influence increment. Thus, the space complexities of the OneHop and TwoHop algorithms are both $O(|\mathcal{V}|)$.

### C. Further Improvement on Efficiency

Note that selecting the first seed in Algorithm 1 requires calculating the influence spread $\sigma(\{v\})$ for every node $v$ even when the CELF technique [20] is adopted. To avoid such computation, we develop an upper bound on $\sigma(\{v\})$ when hop-based influence estimation is applied.

*Theorem 2:* For each node $v \in \mathcal{V}$, the $h$-hop influence spread $\sigma_h(\{v\})$ satisfies

$$\sigma_h(\{v\}) \leq 1 + \sum_{w \in \mathcal{N}_v} \big(p_{v,w} \cdot \sigma_{h-1}(\{w\})\big). \tag{6}$$

Furthermore, let $\hat{\sigma}_0(\{v\}) = \sigma_0(\{v\}) = 1$ and $\hat{\sigma}_h(\{v\}) = 1 + \sum_{w \in \mathcal{N}_v} \big(p_{v,w} \cdot \hat{\sigma}_{h-1}(\{w\})\big)$, then

$$\sigma_h(\{v\}) \leq \hat{\sigma}_h(\{v\}). \tag{7}$$

Note that when $h = 1$, the upper bound $\hat{\sigma}_1(\{v\}) = 1 + \sum_{w \in \mathcal{N}_v} p_{v,w}$ is the exact 1-hop influence spread of a single seed $\{v\}$, i.e., $\hat{\sigma}_1(\{v\}) = \sigma_1(\{v\})$. Computing $\hat{\sigma}_1(\{v\})$ for a node $v$ has a time complexity of $O(1 + |\mathcal{N}_v|)$. Thus, it takes a time complexity of $O(\sum_{v \in \mathcal{V}}(1 + |\mathcal{N}_v|)) = O(|\mathcal{V}| + |\mathcal{E}|)$ to calculate $\hat{\sigma}_1(\{v\})$ for all nodes $v \in \mathcal{V}$. The time complexity for computing the upper bound $\hat{\sigma}_2(\{v\})$ given in Theorem 2 is $O(1 + |\mathcal{N}_v|)$ after obtaining $\hat{\sigma}_1(\{w\})$ for all nodes $w \in \mathcal{V}$. Thus, the total time complexity for calculating $\hat{\sigma}_2(\{v\})$ for all nodes $v \in \mathcal{V}$ is $O(|\mathcal{V}| + |\mathcal{E}|) + O(\sum_{v \in \mathcal{V}}(1 + |\mathcal{N}_v|)) = O(|\mathcal{V}| + |\mathcal{E}|) + O(|\mathcal{V}| + |\mathcal{E}|) = O(|\mathcal{V}| + |\mathcal{E}|)$, which is much lower than the time complexity for computing the exact two-hop influence spread $\sigma_2(\{v\})$ for all nodes $v \in \mathcal{V}$ using Algorithm 3 which is $O\big(|\mathcal{V}| + |\mathcal{E}| + \sum_{w \in \mathcal{V}}(|\mathcal{I}_w| \cdot |\mathcal{N}_w|)\big)$. On obtaining the upper bounds $\hat{\sigma}_h(\{v\})$, the CELF technique described in Section II-B can then be applied to the first iteration of Algorithm 1 so that only the influence spreads of a subset of nodes in $\mathcal{V}$ need to be calculated for selecting the first seed. We shall show in Section IV-B that the upper bounding approach can dramatically reduce the running time of the two-hop method.

### D. Theoretical Analysis

In this section, we carry out theoretical analysis for our hop-based algorithms. We first show that the influence spread within $h$ hops of propagation is submodular and monotone.

*Theorem 3:* For any $h \geq 1$, the influence spread produced within $h$ hops of propagation is submodular and monotone under the IC model.

Let $\mathcal{S}^*$ denote the optimal seed set for maximizing the actual influence without any hop limit, i.e., $\sigma(\mathcal{S}^*) = \max_{|\mathcal{S}|=k} \sigma(\mathcal{S})$. Then, we can derive the following guarantees for our hop-based methods.

*Theorem 4:* Under the IC model, if $\sigma_h(\mathcal{S})/\sigma(\mathcal{S}) \geq \alpha$ for any seed set $|\mathcal{S}| = k$, the solution $\mathcal{S}_h$ returned by the greedy heuristic (Algorithm 1) with hop-based influence estimation satisfies

$$\sigma(\mathcal{S}_h) \geq \big((1 - 1/e)\alpha\big) \cdot \sigma(\mathcal{S}^*). \tag{8}$$

Theorem 4 indicates that if the ratio $\sigma_h(\mathcal{S})/\sigma(\mathcal{S})$ is lower bounded by $\alpha$, the hop-based methods can provide a multiplicative guarantee of $\big((1 - 1/e)\alpha\big)$. Next, we derive a lower bound on the ratio $\sigma_h(\mathcal{S})/\sigma(\mathcal{S})$ in the class of scale free random graphs which are commonly used to model OSNs [2], [22]. The degree distribution of a scale-free (undirected)

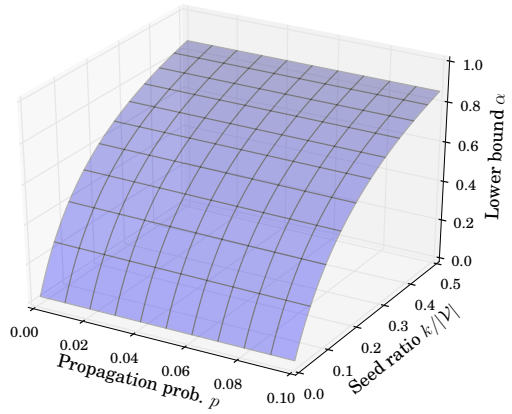



Fig. 5. Lower bound $\alpha$ for different propagation probabilities and seed ratios ($\gamma = 3$).

graph follows a power law. That is, the probability of a node having degree $d$ is $P_0(d) = \frac{d^{-\gamma}}{\sum_{d=1}^{\infty} d^{-\gamma}}$, where $\gamma$ is a given power scale parameter whose typical value is in the range of $2 \leq \gamma \leq 3$. We first analyze the expected number of nodes activated within one hop of propagation, which gives a lower bound on $\sigma_h(\mathcal{S})$ for any $h \geq 1$ since $\sigma_h(\mathcal{S})$ increases with $h$. Next, we derive an upper bound on the expected number of nodes activated $\sigma(\mathcal{S})$. Using the lower bound on $\sigma_h(\mathcal{S})$ and upper bound on $\sigma(\mathcal{S})$, we can derive a lower bound $\alpha$ on the ratio $\sigma_h(\mathcal{S})/\sigma(\mathcal{S})$.

*Theorem 5:* For scale free random graphs with propagation probability $p_{u,v} = p$ for every edge $(u, v) \in \mathcal{E}$ and any seed set $\mathcal{S}$ and any hop of $h \geq 1$, we have

$$\frac{\mathbb{E}[\sigma_h(\mathcal{S})]}{\mathbb{E}[\sigma(\mathcal{S})]} \geq \frac{1 - (1 - k/|\mathcal{V}|)(1 - pk/|\mathcal{V}|)}{1 - (1 - k/|\mathcal{V}|)P_0(1)(1 - pA)}. \quad (9)$$

Fig. 5 shows the lower bound derived in (9) when varying the propagation probability $p$ from 0 to 0.1 and the seed ratio $k/|\mathcal{V}|$ from 0 to 0.5. We can see that the lower bound generally increases with both the seed ratio and propagation probability.

### E. Extension to Linear Threshold Model

The Linear Threshold (LT) model is another basic diffusion model described in [16], which is also widely used [8], [29], [30]. In the LT model, each directed edge $(u, v)$ is associated with a weight $b_{u,v}$, and the total weight of all the incoming edges to each node does not exceed 1. In the diffusion process, each node randomly chooses a threshold between 0 and 1. The diffusion starts from an initial seed set $\mathcal{S}$. A non-seed node becomes activated only when the aggregate weight of the incoming edges from its inverse neighbors that have been activated reaches its threshold.

Our proposed hop-based algorithms can be easily extended to work with the LT model. For one hop of propagation, if $v \in \mathcal{S}$, $\pi_1^{\mathcal{S}}(v) = 1$; otherwise, $\pi_1^{\mathcal{S}}(v) = \sum_{u \in \mathcal{I}_v \cap \mathcal{S}} b_{u,v}$. When adding a new seed $u$ to a seed set $\mathcal{S}$, for each neighbor $v$ of $u$ where $v \notin \mathcal{S}$, the increment of $v$'s activation probability is $\pi_1^{\mathcal{S}\cup\{u\}}(v) - \pi_1^{\mathcal{S}}(v) = b_{u,v}$. Thus, the total increment of influence spread by adding $u$ to $\mathcal{S}$ is $\sigma_1(\mathcal{S} \cup \{u\}) - \sigma_1(\mathcal{S}) = \sum_{v \in \mathcal{N}_u \setminus \mathcal{S}} b_{u,v}$. For two hops of propagation, all the paths from seed nodes to a non-seed node within two hops are acyclic. The increment of influence spread by adding a new seed $u$ to a seed set $\mathcal{S}$ consists of three parts: 1) $1 - \pi_2^{\mathcal{S}}(u)$ from node $u$ itself; 2) $\left(1 - \pi_1^{\mathcal{S}}(u)\right) \cdot \sum_{v \in \mathcal{N}_u \setminus \mathcal{S}} b_{u,v}$ from $u$'s non-seed neighbors; 3) $\sum_{v \in \mathcal{N}_u \setminus \mathcal{S}} \left(b_{u,v} \cdot \sum_{w \in \mathcal{N}_v \setminus (\mathcal{S} \cup \{u\})} b_{v,w}\right)$ from $u$'s neighbors' neighbors which are not seeds. Thus, the total increment of influence spread by adding a new seed $u$ to a seed set $\mathcal{S}$ is $\sigma_2(\mathcal{S}\cup\{u\}) - \sigma_2(\mathcal{S}) = 1 - \pi_2^{\mathcal{S}}(u) + \left(1 - \pi_1^{\mathcal{S}}(u)\right) \cdot \sum_{v \in \mathcal{N}_u \setminus \mathcal{S}} b_{u,v} + \sum_{v \in \mathcal{N}_u \setminus \mathcal{S}} \left(b_{u,v} \cdot \sum_{w \in \mathcal{N}_v \setminus (\mathcal{S} \cup \{u\})} b_{v,w}\right)$. It is easy to show that the influence spread within a fixed number of hops under the LT model is submodular and monotone as well. Thus, our analysis on the hop-based algorithms can also be applied to the LT model.

## IV. Evaluation

### A. Experimental Setup

**Datasets.** We use several real OSN datasets in our experiments [1], [17], [21]. Due to space limitations, we just show the results for three representative datasets: NetHEPT (15K nodes and 32K edges), LiveJournal (5M nodes and 69M edges) and Twitter (42M nodes and 1.5B edges). The results for other datasets are similar.

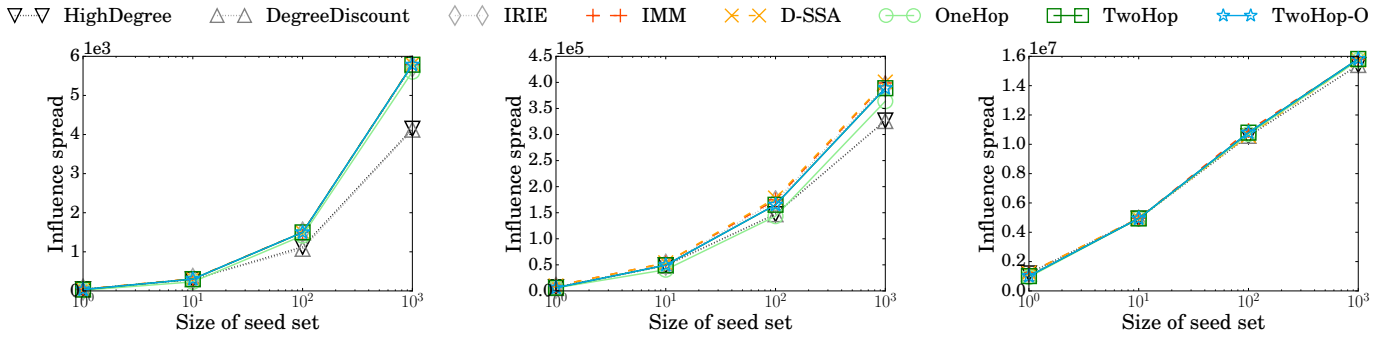
**Algorithms.** We compare our OneHop, TwoHop and TwoHop-O (without the upper bounding technique described in Section III-C) algorithms with the following state-of-the-art algorithms.

- HighDegree: Select the $k$ nodes with highest degrees [16].
- DegreeDiscount: The degree discount heuristic was developed by [7].
- IRIE: IRIE [15] is a state-of-the-art heuristic. We set $\alpha = 0.7$ and $\theta = 1/320$ respectively as suggested in [15].
- IMM: IMM [29] is one of the most advanced sampling-based methods that can provide a $(1 - 1/e - \epsilon)$-approximation guarantee with probability at least $1 - \delta$.
- D-SSA: D-SSA [26] aims to further reduce the number of samples generated compared to IMM while providing the same approximation guarantee. We set $\epsilon = 0.1$ and $\delta = 1/|\mathcal{V}|$ for both IMM and D-SSA according to the default setting in [26].

**Parameter Settings.** For the IC model, we set the propagation probability via the following two models.

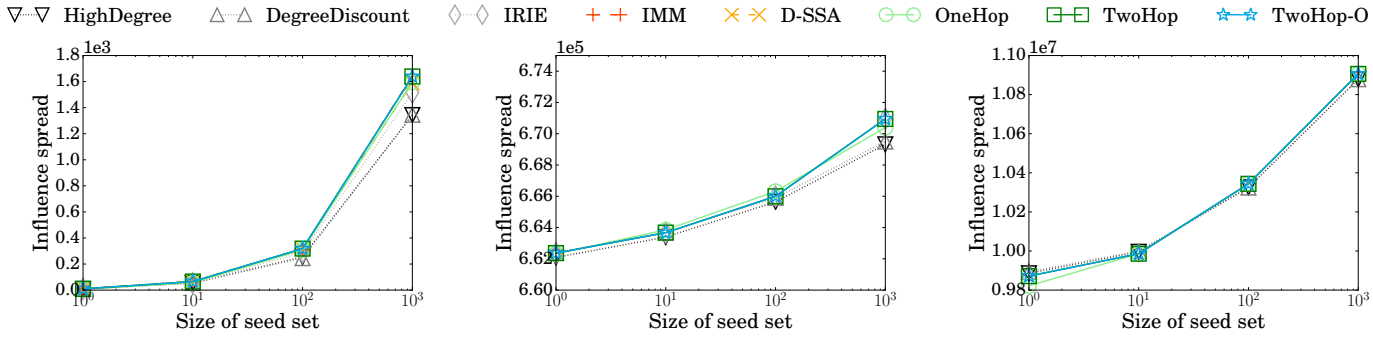
- WC model [7], [16], [26], [29]: $p_{u,v}$ of each edge $(u, v)$ is set to the reciprocal of $v$'s in-degree, i.e., $p_{u,v} = 1/|\mathcal{I}_v|$.
- TRIVALENCY model [6], [15]: $p_{u,v}$ of each edge $(u, v)$ is set by choosing a probability from the set $\{0.1, 0.01, 0.001\}$ at random.

To evaluate the seed sets returned by different algorithms, we estimate the influence spread of each seed set by taking the average measurement of $10,000$ Monte-Carlo simulations. The algorithms are all implemented in C++ and the experiments are carried out on a machine with an Intel Xeon E5-2695 2.4GHz CPU and 64GB memory. We limit the running time of each algorithm up to 100 hours ($3.6 \times 10^5$ seconds).




(a) NetHEPT (b) LiveJournal (c) Twitter (IRIE cannot run)

Fig. 6. Influence spread on various graphs under the WC model.




(a) NetHEPT (b) LiveJournal (IMM and D-SSA cannot run) (c) Twitter (IRIE, IMM and D-SSA cannot run)

Fig. 7. Influence spread on various graphs under the TRIVALENCY model.

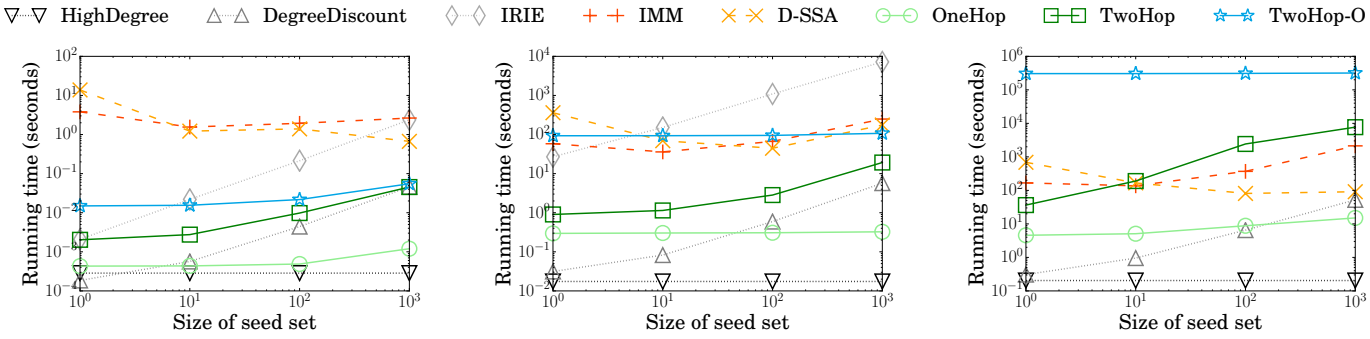
### *B. Results*

**Influence Spread:** Figs. 6 and 7 show the influence spread produced by different algorithms on various graphs when the size of seed set is set to $k = 1, 10, 100, 1000$ under the WC and TRIVALENCY models respectively. Due to out-of-memory reasons and prohibitively long computation times, IRIE *failed* to produce results on the Twitter dataset under both WC and TRIVALENCY models, while both IMM and D-SSA *failed* on the LiveJournal and Twitter datasets under the TRIVALENCY model. From the results obtained, we can make the following observations. Our OneHop, TwoHop and TwoHop-O methods usually generate influence spread as high as that by the IMM and D-SSA methods which can provide the state-of-the-art $(1 - 1/e - \epsilon)$-approximation guarantee. Our methods remarkably outperform both the HighDegree and DegreeDiscount heuristics (by up to $40\%$) on the NetHEPT dataset under both WC and TRIVALENCY models (Figs. 6(a), 7(a)) and on the LiveJournal dataset under the WC model (Fig. 6(b)). These observations demonstrate the effectiveness of our hop-based methods.

**Running Time:** Figs. 8 and 9 show the running times of different algorithms. The OneHop and DegreeDiscount methods run almost at the same speed which can find the top 1000 influential nodes on the Twitter dataset (with billions of edges) within 30 seconds. They are just slightly slower than the HighDegree method and run several orders faster than other methods, including the IRIE, IMM, D-SSA, TwoHop and TwoHop-O methods (note that the y-axis is in logscale). This shows a tradeoff between efficiency and effectiveness for the hop-based methods. Estimating the influence spread with a higher hop limit takes more time but can improve the quality of the seed set chosen. For example, on the LiveJournal dataset under the WC model (Fig. 6(b)), the TwoHop method performs notably better than the OneHop method. If the application is highly time-sensitive, the OneHop method could be preferable to the TwoHop method. Otherwise, the TwoHop method is favoured since its running time is quite acceptable even for very large networks.

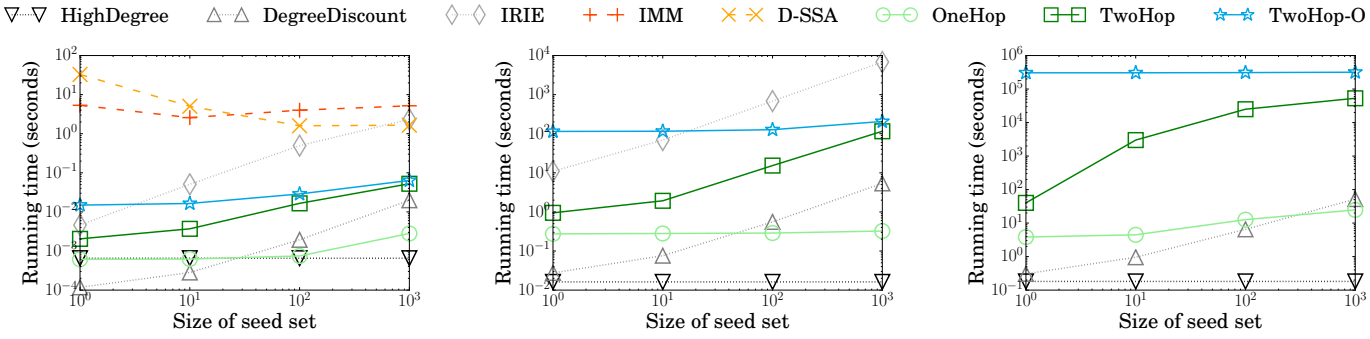
We also observe that while the TwoHop and TwoHop-O methods always produce the same seed set solution, the former runs significantly faster than the latter (by up to $4$ orders of magnitude). This is because TwoHop-O consumes too much time on computing the influence spread of all the single seed sets. This demonstrates the efficiency of our upper bounding approach.

Moreover, we observe that the OneHop and TwoHop methods generally run much faster than the state-of-the-art IRIE, IMM and D-SSA methods. This demonstrates the efficiency of our hop-based methods. Since IMM and D-SSA are reverse influence sampling methods, their running times heavily depend on the sizes of the reverse reachable sets sampled. Under the TRIVALENCY model, all the edges are likely to have substantial propagation probabilities. As a result, high-degree nodes can have many direct reverse reachable neighbors in a




(a) NetHEPT (b) LiveJournal (c) Twitter (IRIE cannot run)

Fig. 8. Running time on various graphs under the WC model.




(a) NetHEPT (b) LiveJournal (IMM and D-SSA cannot run) (c) Twitter (IRIE, IMM and D-SSA cannot run)

Fig. 9. Running time on various graphs under the TRIVALENCY model.

TABLE I
RUNNING TIME (SECONDS) FOR SELECTING 1000 SEEDS IN THE TWITTER GRAPH UNDER THE WC MODEL WITH DIFFERENT PROPAGATION PROBABILITY SCALE FACTORS $f$.

| Method | $f = 1.0$ | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 |
|---|---|---|---|---|---|---|
| HighDegree | 0.20 | 0.20 | 0.21 | 0.21 | 0.20 | 0.21 |
| DegreeDiscount | 52.23 | 51.79 | 51.71 | 52.14 | 52.70 | 51.53 |
| IRIE | – | – | – | – | – | – |
| IMM | 2.13e3 | 3.52e4 | – | – | – | – |
| D-SSA | 91.21 | 3.76e3 | 2.44e4 | – | – | – |
| OneHop | 15.07 | 16.08 | 17.70 | 17.18 | 17.33 | 17.85 |
| TwoHop | 7.70e3 | 7.89e3 | 8.11e3 | 8.25e3 | 7.94e3 | 8.32e3 |

The field with "–" means that the method cannot run under the setting.

sample outcome of influence propagation. Thus, the samples of reverse reachable sets are large and require significant memory and time to compute, which makes IMM and D-SSA intractable on the LiveJournal and Twitter datasets. Under the WC model, for any node, the expected number of its direct reverse reachable neighbors is 1 because the edge propagation probabilities are given by the reciprocal of the node's in-degree. This significantly limits the sizes of reverse reachable sets and favors IMM and D-SSA. However, the running times of IMM and D-SSA are still very sensitive to the propagation probabilities. Table I shows the trends when we scale the propagation probabilities under the WC model by a small factor up to 1.5 (for selecting 1000 seeds in the Twitter graph). It can be seen that the running times of IMM and D-SSA increase rapidly with the scale factor and far exceed our hop-based methods even at a minor factor of 1.2. In contrast, our hop-based methods are much less sensitive to the propagation probabilities.

**Remark on Memory Usage:** Recall that our hop-based algorithms just need $O(|\mathcal{V}|)$ space which is negligible compared to the space $O(|\mathcal{V}| + |\mathcal{E}|)$ required for storing the OSN graph. On the other hand, the IRIE, IMM and D-SSA methods have significantly higher space complexities, e.g., those of IMM and D-SSA are both $O\left(\frac{\left(\ln(1/\delta)+\ln\binom{|\mathcal{V}|}{k}\right)\cdot(|\mathcal{V}|+|\mathcal{E}|)}{\epsilon^2}\right)$ [26], [29]. Thus, they fail to produce results on very large datasets. Our hop-based algorithms never face the out-of-memory problems as long as the memory is large enough to store the OSN graph.

## V. RELATED WORK

Since the $(1 - 1/e - \epsilon)$-approximation greedy algorithm was proposed by [16] for influence maximization, there has been considerable research on improving the efficiency of the greedy algorithm by using heuristics to trade the accuracy of influence estimation for computational efficiency [6], [7], [8], [15], or optimizing the Monte-Carlo simulations for influence estimation [3], [25], [26], [27], [29], [30]. Among them, the DegreeDiscount heuristic [7] roughly estimated influence spread within one-hop neighborhood and the PMIA heuristic [6] used independent propagation paths to construct arborescences for rough influence estimation. However, PMIA is very costly in both time and space compared to a follow-up IRIE algorithm [15]. DegreeDiscount and IRIE as well as advanced

sampling-based methods are all included in our experimental comparison. Leveraging the independency among propagation paths, we have developed efficient hop-based methods to compute the *exact* influence spread within a certain number of hops.

Our hop-based influence estimation is in spirit similar to the time-constrained independent cascade model studied in [5], [11], [23] by concentrating on the diffusion within a fixed number of hops. We make new technological advances by inventing algorithms to compute the exact influence spreads of one-hop and two-hop propagations, which could only be approximately estimated in previous work where the approximation guarantee is difficult to analyze [18]. Our algorithms enable very efficient evaluation of the change in influence spread when a new seed node is added. We also derive an upper bound on the influence spread to further speed up our hop-based algorithms. Our hop-based approaches can be easily applied to many influence-based applications, such as topic-aware influence maximization [31] and community detection via influence maximization [14].

## VI. Conclusion

In this paper, we have proposed lightweight hop-based methods to address the problem of influence maximization in OSNs. We have also developed an upper bounding technique to further speed up the seed selection algorithm. Through analysis, we show that our methods can provide certain theoretical guarantees. Experiments are conducted with real OSN datasets to compare the efficiency and effectiveness of our algorithms with state-of-the-art ones. In terms of solution quality, our hop-based methods are on par with the most advanced IMM and D-SSA methods which can provide the best $(1 - 1/e - \epsilon)$-approximation guarantee and remarkably outperform the HighDegree and DegreeDiscount heuristics for quite some cases. In terms of efficiency, our hop-based methods run much faster than the IRIE, IMM and D-SSA methods for most cases tested. Furthermore, while all these existing methods fail to run on some test cases, our hop-based methods can always execute and find solutions.

## Acknowledgment

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative, and by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2013-T1-002-123.

## References

[1] http://research.microsoft.com/en-us/people/weic/projects.aspx.
[2] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
[3] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proc. SODA*, 2014, pp. 946–957.
[4] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *Proc. WWW*, 2009, pp. 721–730.
[5] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," *Proc. AAAI*, pp. 592–598, 2012.
[6] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. ACM KDD*, 2010, pp. 1029–1038.
[7] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proc. ACM KDD*, 2009, pp. 199–208.
[8] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *Proc. IEEE ICDM*, 2010, pp. 88–97.
[9] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng, "IMRank: Influence maximization via finding self-consistent ranking," in *Proc. ACM SIGIR*, 2014, pp. 475–484.
[10] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, "Sketch-based influence maximization and computation: Scaling up with guarantees," in *Proc. ACM CIKM*, 2014, pp. 629–638.
[11] T. N. Dinh, H. Zhang, D. T. Nguyen, and M. T. Thai, "Cost-effective viral marketing for time-critical campaigns in large-scale social networks," *IEEE/ACM Trans. Networking*, vol. 22, no. 6, pp. 2001–2011, 2014.
[12] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. ACM KDD*, 2001, pp. 57–66.
[13] S. Goel, D. J. Watts, and D. G. Goldstein, "The structure of online diffusion networks," in *Proc. ACM EC*, 2012, pp. 623–638.
[14] F. Jiang, S. Jin, Y. Wu, and J. Xu, "A uniform framework for community detection via influence maximization in social networks," in *Proc. IEEE/ACM ASONAM*, 2014, pp. 27–32.
[15] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in *Proc. IEEE ICDM*, 2012, pp. 918–923.
[16] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. ACM KDD*, 2003, pp. 137–146.
[17] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. WWW*, 2010, pp. 591–600.
[18] J. R. Lee and C. W. Chung, "A fast approximation for influence maximization in large social networks," in *WWW Companion*, 2014, pp. 1157–1162.
[19] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web*, vol. 1, no. 1, 2007.
[20] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. ACM KDD*, 2007, pp. 420–429.
[21] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, 2014.
[22] Y. Li, B. Q. Zhao, and J. C. S. Lui, "On modeling product advertisement in large-scale online social networks," *IEEE/ACM Trans. Networking*, vol. 20, no. 5, pp. 1412–1425, 2012.
[23] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *Proc. IEEE ICDM*, 2012, pp. 439–448.
[24] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
[25] H. T. Nguyen, T. N. Dinh, and M. T. Thai, "Cost-aware targeted viral marketing in billion-scale networks," in *Proc. IEEE INFOCOM*, 2016.
[26] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," in *Proc. ACM SIGMOD*, 2016, pp. 695–710.
[27] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi, "Fast and accurate influence maximization on large networks with pruned monte-carlo simulations," in *Proc. AAAI*, 2014, pp. 138–144.
[28] J. Tang, X. Tang, and J. Yuan, "Influence maximization meets efficiency and effectiveness: A hop-based approach," arXiv preprint, https://arxiv.org/abs/1705.10442, 2017.
[29] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proc. ACM SIGMOD*, 2015, pp. 1539–1554.
[30] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proc. ACM SIGMOD*, 2014, pp. 75–86.
[31] C. Zhang, J. Sun, and K. Wang, "Information propagation in microblog networks," in *Proc. IEEE/ACM ASONAM*, 2013, pp. 190–196.
[32] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo, "UBLF: An upper bound based approach to discover influential nodes in social networks," in *Proc. IEEE ICDM*, 2013, pp. 907–916.