

On Budgeted Influence Maximization in Social Networks

Huy Nguyen

Department of Computer Science
University of Houston
Houston, TX 77204
hanguyen5@uh.edu

Rong Zheng

Department of Computing and Software
McMaster University
Hamilton, ON, L8S 4K1, Canada
rzheng@mcmaster.ca

Abstract—Given a budget and arbitrary cost for selecting each node, the budgeted influence maximization (BIM) problem concerns selecting a set of seed nodes to disseminate some information that maximizes the total number of nodes influenced (termed as influence spread) in social networks at a total cost no more than the budget. Our proposed seed selection algorithm for the BIM problem guarantees an approximation ratio of $(1 - 1/\sqrt{e})$. The seed selection algorithm needs to calculate the influence spread of candidate seed sets, which is known to be #P-complex. Identifying the linkage between the computation of marginal probabilities in Bayesian networks and the influence spread, we devise efficient heuristic algorithms for the latter problem. Experiments using both large-scale social networks and synthetically generated networks demonstrate superior performance of the proposed algorithm with moderate computation costs. Moreover, synthetic datasets allow us to vary the network parameters and gain important insights on the impact of graph structures on the performance of different algorithms.

Index Terms—Budgeted influence maximization, social network, information diffusion, belief propagation.

I. INTRODUCTION

The social network of interactions among a group of individuals plays a fundamental role in the spread of information, ideas, and influence. Such effects have been observed in real life, when an idea or an action gains sudden widespread popularity through “word-of-mouth” or “viral marketing” effects. For example, free e-mail services such as Microsoft’s Hotmail, later Google’s Gmail, and most recently Google’s Google+ achieved wide usage largely through referrals, rather than direct advertising.

In viral marketing, one important question is given limited advertisement resources, which set of customers should be targeted such that the resulting influenced population is maximized. Consider a social network modeled as a graph with vertices representing individuals and edges representing connections or relationship between two individuals. The influence maximization (IM) problem tries to find a seed set S with cardinality $|S| = k$ in the graph such that the expected number of nodes influenced by S is maximized [1], [2], [3]. With the cardinality constraint, the submodularity nature of the influence spread renders a greedy algorithm with $(1 - 1/e)$ approximate ratio that in each round picks the seed with maximum influence spread and runs for k rounds. However the assumption of equal costs for all seed nodes seldom holds in practice. With the proliferation of influence score services

such as Klout and PeerIndex¹, one can easily measure his influence in the social sphere and use that to negotiate the price for services he provides. The higher the influence score of a user, the more costly it is to persuade him.

We consider in this paper a generalized version of the IM problem, namely, the budgeted influence maximization (BIM) problem: given a fixed budget b and a random cost function c , find a seed set S which fits the budget $\sum_{s_i \in S} c(s_i) \leq b$ and maximizes the number of influenced nodes. Clearly, BIM is more relevant in practice as there is typically a price associated with initializing the dissemination of information. With the budget constraint, we prove that direct application of the simple greedy algorithm may result in unbounded performance gap.

In this paper, we present a seed selection algorithm that can attain an approximation guarantee of $(1 - 1/\sqrt{e})$ (~ 0.394). One critical component of the seed selection process is the determination of influence spread of a set of seeds. Exact computation of influence spread is proven to be of #P-complete [3]. Thus, efficient algorithms need to be devised. More specifically, we first establish the linkage between influence spread computation and belief propagation on a Bayesian network (modeled as a directed acyclic graph [DAG]), where the marginal conditional dependency corresponds to the influence probabilities. Belief propagation has been extensively studied in literatures, and thus many exact or approximation algorithms can be leveraged to estimate the influence spread. For a general graph that contains loops, we propose two approximation algorithms that prune some edges in the graph to obtain a DAG that captures the bulk of influence spread. To reduce the number of candidate seed nodes, we localize the influence spread region such that at each round, only nodes that are affected by the previously selected seed need to be evaluated. Empirical study shows that the proposed algorithms can scale up to large-scale graphs with millions of edges with high accuracy. On real-world social network graphs, our methods achieve influence spread comparable to that by Greedy algorithm [3] and incur significant less computation costs. In the unit-cost IM problem, the proposed methods outperform PMIA [4] in achievable influence spread at the expense of marginal increase in computation time. In the BIM problem, the proposed methods outperform CELF [5] in term of scalability and performance on dense graphs. We further

¹<http://www.klout.com> and <http://www.peerindex.com>

study the effect of network structures on the performance of the algorithms.

The main contributions of this paper are summarized as follows:

- We propose a greedy algorithm for BIM with a constant approximation ratio.
- We cast the problem of inference spread computation on a DAG as an instance of belief propagation on a Bayesian network.
- We prove the #P-hardness of inference spread computation on a DAG.
- Two heuristics are proposed to construct DAGs from a general graph that capture the bulk of influence spread.
- We provide important insights on the impact of graph structures on performance of different algorithms.

The rest of this paper is organized as follows. In Section II, we give a comprehensive review of the related literatures. Section III presents the seed selection algorithm and proves its performance bound. Theoretical results concerning influence spread on DAGs are in Section IV. In Section V, we devise two heuristics to reduce a general directed graph into a DAG which captures the majority of influence spread. From the presented theoretical results, we have the main algorithm in Section VI. In Section VII, extensive experiment results are presented. Finally we conclude the paper and discuss future research directions in Section VIII.

II. RELATED WORK

Kempe *et al.* in [3] are the first to formulate the IM problem. The authors proved the submodularity of the influence spread function and suggested a greedy scheme (henceforth referred to as Greedy algorithm) with an incremental oracle that identifies, in each iteration, a new seed that maximizes the incremental spread. The approach was proven to be a $(1-1/e)$ -approximation of the IM problem. However, Greedy suffers from two sources of computational deficiency: 1) the need to evaluate many candidate nodes before selecting a new seed in each round, and 2) the calculation of the influence spread of any seed set relies on Monte-Carlo simulations.

In an effort to improve Greedy, Leskovec *et al.* [5] recognized that not all remaining nodes need to be evaluated in each round and proposed the “Cost-Effective Lazy Forward” (CELf) scheme. Experimental results demonstrate that CELf optimization could achieve as much as 700-time speed-up in selecting seeds. However, even with CELf mechanism, the number of candidate seeds is still large. Recently, Goyal *et al.* proposed CELf++ [6] that has been shown to run from 35% to 55% faster than CELf. However, the improvement comes at the cost of higher space complexity to maintain a larger data structure to store the look-ahead marginal gains of each node.

Chen *et al.* devises several heuristic algorithms for influence spread computation [7], [4], [8]. In Degree Discount [7], the expected number of additional vertices influenced by adding a node v in the seed set is estimated based on v ’s one hop neighborhoods. It also assumes that the influence probability is identical on all edges. In [4] and [8], two approximation algorithms, PMIA and LDAG are proposed to compute the

maximum influence set under IC and LT models, respectively. In LDAG, it has been proven that under the LT model, computing influence spread in a DAG has linear time complexity, and a heuristic on local DAG construction is provided to further reduce the compute time. We have proven in Section IV that computing influence spread in a DAG under the IC model remains #P-hard. The marked difference between the two results arises from the fact that in the LT model, the activation of incoming edges is coupled so that in each instance, only one neighbor can influence the node of interest in an equivalent random graph model.

Another line of work explores diffusion models beyond LT and IC. Even-Dar *et al.* [9] argue that the most natural model to represent diffusion of opinions in a social network is the probabilistic voter model where in each round, each person changes his opinion by choosing one of his neighbors at random and adopting the neighbor’s opinion. Interestingly, they show that the straightforward greedy solution, which picks the nodes in the network with the highest degree, is optimal. Sylvester [10] studies the spread maximization problem on dynamic networks and examines the use of dynamic measures with Greedy algorithm on both LT and IC models. Chen *et al.* [11] consider a new model that incorporates negativity bias and design an algorithm to compute influence on tree structures.

Inapproximability results of problems related to IM have also been investigated in literature. MINSEED is the problem of finding the minimized seed set size to activate all or a portion of vertices. Chen [12] proves that under LT model with a general threshold, MINSEED can not be approximated within a ratio of $O(2^{\log^{1-\varepsilon} n})$, for any fixed $\varepsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$. In the case when the threshold equals two, the author proves that it is as hard as the case with a general threshold, even for constant degree graphs. Ackerman *et al.* [13] cast MINSEED and IM as maximization problems making them amenable to optimization techniques. However, since the number of variables and constraints grow in $O(n^2)$ and $O(n^3)$ respectively – n being the number of vertices in the graph – this approach is only tractable in small-size problems. MINTIME is the problem of finding a target size k such that all or a portion of vertices are activated in the minimum possible time (in terms of spread time or hop count). With a given coverage threshold η , Goyal *et al.* [14] prove that under both IC and LT model, the greedy algorithm can produce the result covering $\eta - \varepsilon$ vertices ($\varepsilon > 0$) in min time, with seed size $|S| \leq k(1 + \ln(\eta/\varepsilon))$. Ni *et al.* [15] investigate the MINTIME problem by proposing a new spread model and proving various timing bounds on the proposed model.

Literatures on epidemiology are also related to the IM problem that identifies nodes that can initiate viral propagation to most part of the network. Under the proposed model, the authors of [16] proved that the epidemic threshold for a network is exactly the inverse of the largest eigenvalue of its adjacency matrix. In a follow-up work [17], the authors used the previously defined epidemic threshold to quantify the vulnerability of a given network and devised a fast algorithm to choose the best k nodes to be immunized (removed) so as to minimize network vulnerability. [18] considered the immu-

nization problem on dynamic networks. The key differences between work on viral immunization literatures and IM lie in the spreading model adopted (e.g.: SIS [susceptible-infected-susceptible] or SIR [susceptible-infected-recovered] vs. IC or LT) and whether the dynamics in the evolution of influence are of interest.

Most existing work on the IM problem only considers cardinality constraints. CELF [5] is the only applicable approach to the BIM problem. We will later show in our evaluation that the proposed methods outperform CELF in term of running time (several orders of magnitude faster) and performance on dense networks.

This article is an extended version of our conference paper in [19]. We modified our main algorithm to solve the BIM problem and prove its approximation factor. We added detailed algorithm description, complexity analysis, and report more comprehensive results regarding algorithm performance on different real datasets. We also conducted new experiment sets on synthetic networks and provide results on the impact of graph structure on different IM algorithms which, to the best of our knowledge, has never been studied before.

III. THE BUDGETED INFLUENCE MAXIMIZATION PROBLEM

In this section, we consider the BIM problem with the objective to select the seed set that maximizes influence spread given a fixed budget and arbitrary node costs.

A. Problem Formulation

Consider the network a directed graph $\mathcal{G} = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges. For every edge $(u, v) \in E$, $p(u, v)$ denotes the probability of influence being propagated on the edge. In this paper, we adopt the Independent Cascade (IC) model. Given a seed set $S \subseteq V$, the IC model works as follows. Let $S_t \subseteq V$ be the set of node (newly) activated at time t , with $S_0 = S$ and $S_t \cap S_{t-1} = \emptyset$. At round $t + 1$, every node $u \in S_t$ tries to activate its neighbors in $v \in V \setminus \bigcup_{0 \leq i \leq t} S_i$ independently with probability $p(u, v)$. The influence spread of S , denoted by $\sigma(S)$, is the *expected* number of activated nodes given seed set S .

Kempe *et al.* [3] proved two important properties of the $\sigma(\cdot)$ function: 1) $\sigma(\cdot)$ is *submodular*, namely, $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all $v \in V$ and all subsets S and T with $S \subseteq T \subseteq V$; 2) $\sigma(S)$ is *monotone*, i.e. $\sigma(S) \leq \sigma(T)$ for all set $S \subseteq T$. For any given spread function $\sigma(\cdot)$ that is both submodular and monotone, the problem of finding a set S of size k that maximizes $\sigma(S)$ can be approximated by a simple greedy approach.

Budgeted Influence Maximization: In BIM, each node u is associated with an arbitrary cost $c(u)$. The goal is to select a seed set $S \subseteq V$ such that the total cost of this set is less than a budget b . Denote by $c(S)$ the total cost of a set, i.e., $c(S) = \sum_{u \in S} c(u)$. Budgeted IM (BIM) can be formulated as an optimization problem:

$$\begin{aligned} \max_{S \subseteq V} \quad & \sigma(S) \\ \text{s.t.} \quad & c(S) \leq b \end{aligned} \quad (1)$$

TABLE I: Notations

\mathcal{G}, V, E	the directed graph, its set of vertices and edges
n, m	the number of nodes, edges in \mathcal{G}
k, b	the budget in term of node count and cost
$p(u, v)$	the propagation probability from u to v
$p(v)$	the activation probability of the node v
$c(v)$	the cost of the node v
$Par(v)$	the set of parents of the node v
S	the selected seed set
θ	the influence threshold
$\sigma(S)$	the influence spread of the set S
$\delta(v)$	the incremental influence spread of selecting v as a seed node
$\mathcal{D}(S)$	the directed acyclic graph from \mathcal{G} on which influence is spread given the seed set S

Algorithm 1: Naive Greedy

input : $G = (V, E), b$

```

1  $S = \emptyset$ 
2 repeat
3    $\delta(v) = (\sigma(S \cup v) - \sigma(S))/c(v), \forall v \in V$ 
4    $u = \arg \max_{v \in V} \delta(v)$ 
5   if  $c(S \cup u) \leq b$  then
6      $S = S \cup u$ 
7    $V = V \setminus u$ 
until  $V = \emptyset$ ;
output:  $S$ 
```

When $c(u) \equiv 1, \forall u \in S$, BIM degenerates to the original IM problem. Thus, we call IM the unit-cost BIM. Since IM is NP-hard, it is easy to see that BIM is NP-hard as well. Key notations used in this paper are summarized in Table I.

B. The Seed Selection Algorithm

First, we consider an intuitive greedy strategy that selects at each step a node u that maximizes the spread gained over cost ratio if the cost of u is less than the remaining budget. We hereby refer to this scheme as the *Naive Greedy* approach. Let r be the number of iterations executed and S_r be the seed set at step r . Note that $|S_r| \leq r$. At step $r + 1$, Naive Greedy calculates the incremental spread-cost ratio.

$$\delta(v) = (\sigma(S \cup v) - \sigma(S))/c(v), \forall v \in V \setminus S. \quad (2)$$

The algorithm chooses u if $u = \arg \max_{v \in V, c(S_r \cup v) \leq b} \delta(v)$. The algorithm terminates when no budget remains, or no node can be added to S . Naive Greedy is summarized in Algorithm 1.

We first observe that Naive Greedy can have unbounded approximation ratio. Consider a network containing $l + 1$ nodes $V = \{u, v_1, v_2, \dots, v_l\}$. Every pair in v_1, v_2, \dots, v_l is connected by an edge with influence probability one, while u is an isolated node. Let the cost $c(u) = 1 - \varepsilon$, $c(v_i) = l, \forall i = 1, \dots, l$ and the budget $b = l$. The optimal solution will pick any node v_i and achieve an influence spread of l . In contrast, Naive Greedy picks u since it has the maximum

Algorithm 2: Improved Greedy

input : $G = (V, E), b$

- 1 $S_1 = \text{result of Naive Greedy}$
- 2 $s_{max} = \arg \max_{v \in V} \sigma(v)$
- 3 $S = \arg \max(\sigma(S_1), \sigma(s_{max}))$

output: S

influence-cost ratio $1/(1-\epsilon) > 1$. The resulting influence spread is 1. Thus, the approximation ratio for Naive Greedy is l .

Next, we show that Naive Greedy can be modified to achieve a constant approximation ratio. This algorithm is an adaptation of an algorithm first proposed by Khuller *et al.* [20]. We assume that there is no node with a cost greater than the budget b , as it will never be a feasible solution to BIM. Let S_1 be the seed set selected by Naive Greedy, we consider another candidate solution s_{max} , which is the node that has the largest influence. We compare the spread of S_1 and s_{max} , then output the one with higher influence spread. The process is illustrated in Algorithm 2.

Theorem 1: Algorithm 2 provides a $(1 - 1/\sqrt{e})$ -approximation for the BIM problem.

By considering the candidate solution with the maximum influence spread, Algorithm 2 guarantees the approximation ratio within a constant factor, while Algorithm 1 is unbounded. Note that Algorithm 2 is different from CELF presented by Leskovec *et al.* in [5]. CELF runs Naive Greedy on the budgeted and the unit-cost (by setting all costs to one) versions of the problem, and selects the set with the maximum influence spread. While finding the seed set to maximize IM consumes more time than what it takes to select a single node with the largest spread, CELF can only guarantee a looser bound of $\frac{1}{2}(1 - 1/e)$ (~ 0.316).

Complexity: Let T be the maximum time needed to calculate the value of $\sigma(S), \forall S \subseteq V$. Algorithm 1 runs in $O(n^2T)$ time where n is the number of nodes (i.e. $n = |V|$). Finding S_1 therefore costs $O(n^2T)$. s_{max} can be determined in $O(nT)$ time. Algorithm 2 therefore runs in $O(n^2T)$ time. Note in [20] that Greedy with partial enumeration heuristic can achieve an approximation guarantee of $(1 - 1/e)$. However, the improvement is attained at the expense of much higher computation complexity of $O(n^4d)$ [21].

Algorithm 2 calls $\sigma(\cdot)$ as a subroutine. The efficiency of $\sigma(\cdot)$ computation is thus critical to the overall running time of the algorithm. In the following sections, we develop efficient algorithms for approximating the spread function $\sigma(\cdot)$. We first consider the special case when the network is a directed acyclic graph (DAG). Then, we provide two DAG construction algorithms from a general network graph. Finally, some techniques to further optimize the execution of Algorithm 2 is presented.

IV. DETERMINING INFLUENCE SPREAD ON DAG

Given a seed set, estimating value of the $\sigma(\cdot)$ from that seed set was proven to be a #P-complete problem [3]. We show in this section that under the IC model, calculation of $\sigma(\cdot)$ remains #P-complete even when the underlying network graph is a DAG. Then we establish the equivalence between computing $\sigma(\cdot)$ on a DAG and the computation of marginal probabilities in a Bayesian network.

A. Hardness of Computing Influence Spread on DAGs

In [3], Kempe *et al.* proposed an equivalent process of influence spread under the IC model, where at the initial stage, an edge (u, v) in \mathcal{G} is declared to be *live* with probability $p(u, v)$ resulting in a subgraph of \mathcal{G} . A node u is active if and only if there is at least one path from some node in S to u consisting entirely of *live edges*. In general graphs, the influencer-influencee relationship may differ in one realization to another for bi-directed edges. In a DAG, on the other hand, such relationship is fixed and is independent of the outcome of the coin flips at the initial stage (other than the fact that some of the edges may not be present). Let $x_u, u \in V$ denotes the binary random variable of the active state of node u , namely, $\mathbb{P}(x_u = 1) = p(u)$. For each node v in S , $\mathbb{P}(x_v = 1) = 1$. If a node $u \notin S$ does not have any parent in \mathcal{G} then $\mathbb{P}(x_u = 1) = 0$. From \mathcal{G} , the conditional probability $p(x_u | x_{Par(u)})$ is uniquely determined by the edge probability, where $x_{Par(u)}$ denotes the states of the parents of node u . In other words, influence spread can be modeled as a Bayesian network. If node u does not have any parent, $p(x_u | x_{Par(u)}) = p(x_u)$. The joint distribution is thus given by,

$$p(x_1, x_2, \dots, x_n) = \prod_{i=1}^n p(x_i | x_{Par(x_i)}). \quad (3)$$

Given the outcome of coin flips C , $\sigma_C(S) = \sum_{u \in V} x_u$. Therefore,

$$\sigma(S) = \mathbb{E}(\sigma_C(S)) = \sum_{u \in V} \mathbb{E}(x_u) = \sum_{u \in V} p(u). \quad (4)$$

The second equality is due to the linearity of expectations. To compute $p(u)$, we can sum (3) over all possible configurations for $x_v, v \in V \setminus u$. Clearly, such a naive approach has complexity that is exponential in the network's treewidth. In fact, the marginalization problem is known to be #P-complete on a DAG. However, since computing influence spread on a DAG can be reduced to a special instance of the marginalization problem, it remains to be shown if the former problem is #P-complete. The main result is summarized in the following theorem².

Theorem 2: Computing the influence spread $\sigma(S)$ on a DAG given a seed set S is #P-complete.

²All proofs are presented in the Appendix

B. Estimating $\sigma(\cdot)$ via Belief Propagation

Belief propagation (BP) is a message passing algorithm for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes [22]. For *singly-connected* DAGs, where between any two vertices there is only one simple path, the BP algorithm in [23] computes the exact solution with $O(n)$ complexity. For multi-connected DAGs, where multiple simple paths may exist between two vertices, belief propagation and many of its variants [22], [24], [25] have been shown to work well in general. Exact solutions such as junction tree [24] may incur the worst case complexity exponential to the number of vertices due to the need to enumerate all cliques in the DAG.

BP algorithms take as input a factor graph or a description of the underlying Bayesian Network. In the context of influence spreading, each node only has two states: active and inactive. BP algorithms calculate the probability of each node in either states. $\sigma(\cdot)$ can then be determined by summing up the probability of nodes being active.

Computation complexity: The complexity of $\sigma(\cdot)$ calculation is dominated by the execution of the BP algorithm. A variety of BP algorithms exist. In this work, we adopt the Loopy Belief Propagation (LBP) algorithm which was shown to perform well for various problems [26], [27]. LBP takes $O(M^d)$ to estimate the active probability of a node, where M is the number of possible labels (states) for each variable ($M = 2$), and d is the maximum in-degree. We denote by n_0 the number of vertices in a DAG. Thus, the complexity of LBP is $O(n_0 2^d)$.

C. A Single Pass Belief Propagation Heuristic for $\sigma(\cdot)$ Estimation

Calculating $\sigma(\cdot)$ with LBP produces highly accurate results, but the computation time remains to be high when the graph is multi-connected. The main complexity arises from the fact that the activation of parents of a node may be correlated in a multi-connected graph. Thus, in computing the activation probability of the node, one needs to account for the joint distribution of its parent nodes. Next, we propose a single pass belief propagation (SPBP) algorithm that ignores such correlation in determining $\sigma(\cdot)$. Note that the heuristic is exact when the graph is singly-connected.

Let $\mathcal{D}(\cdot)$ be the input DAG. Consider a node $v \in \mathcal{D}(\cdot)$. Given the activation probabilities of its parents $Par(v)$, we approximate $p(v)$ as,

$$p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u, v)). \quad (5)$$

The algorithm is summarized in Algorithm 3. It starts with the seed nodes and proceeds with the topological sorting order. The total complexity is $O(n_0 d)$. Clearly, SPBP is much faster than LBP.

V. DAG CONSTRUCTION

In general, real social networks are not DAGs (with the exception of advisor-advisee and parent-child relationship, for

Algorithm 3: Single-Pass Belief Propagation (SPBP)

input : $\mathcal{D}(S)$

```

1  $\sigma(S) = 0;$ 
2 foreach  $v \in \mathcal{D}(S)$  do
3   if  $v \in S$  then
4      $p(v) = 1$ 
5   else
6      $p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u, v))$ 
7    $\sigma(S) = \sigma(S) + p(v)$ 

```

output: $\sigma(S)$

instance, which exhibit a natural hierarchy). To apply the BP algorithm in computing influence spread, one needs to selectively prune edges and reduce the graph to a DAG. Clearly, there are many ways to do so. The challenge is to find a DAG that approximates well the original graph in influence spread. In this section, we introduce two DAG construction algorithms, both retaining important edges where influences are likely to travel.

A. Localizing Influence Spread Region

One important observation in [4] is that the influence of a seed node diminishes quickly along a path away from the seed node. In other words, the “perimeter” of influence or the *influence region* of a seed node is in fact very small. One way to characterize the *influence region* of a node v is through the union of maximum influence paths defined next.

Definition 1: (Path Propagation Probability)

For a given path $P(u, v) = \{u_1, u_2, \dots, u_l\}$ of length l from a vertex u to v , with $u_1 = u, u_l = v$ and u_2, \dots, u_{l-1} are intermediate vertices, define the propagation probability of the path, $p(P)$, as:

$$p(P(u, v)) = \prod_{i=1}^{l-1} p(u_i, u_{i+1}). \quad (6)$$

$p(P(u, v))$ can be thought as the probability that u will influence v if u is selected as a seed node. Obviously, the longer the path length l , the smaller the chance that u can spread its influence to v .

Definition 2: (Maximum Influence Path)

Denote by $\mathcal{P}(\mathcal{G}, u, v)$ the set of all paths from u to v in \mathcal{G} . The maximum influence path $MIP(\mathcal{G}, u, v)$ from u to v is defined as:

$$MIP(\mathcal{G}, u, v) = \arg \max_P \{p(P) | P \in \mathcal{P}(\mathcal{G}, u, v)\}. \quad (7)$$

Ties are broken in a predetermined and consistent way such that $MIP(\mathcal{G}, u, v)$ is always unique, and any sub-path in $MIP(\mathcal{G}, u, v)$ from x to y is also the $MIP(\mathcal{G}, x, y)$. In order to localize the influence region of nodes and reduce the complexity, we only consider influence spread on maximum influence paths.

Definition 3: (Maximum Influence Out-Arborescence)

For a graph \mathcal{G} , an influence threshold θ , the maximum influence out-arborescence of a node $u \in V$, $MIOA(\mathcal{G}, u, \theta)$, is

Algorithm 4: Calculate $\mathcal{D}_1(S)$ from a seed set S

input : \mathcal{G}, S, θ

- 1 Build $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$
- 2 $\mathcal{D}_1(S) = MIOA(\mathcal{G}_R, R, \theta) \setminus R$
- 3 Calculate $r(v), \forall v \in V_{\mathcal{D}_1}$ (Eq. (9))
- 4 **foreach** $(u, v) \in V_{\mathcal{G}_R}$ **do**
- 5 **if** $r(u) < r(v)$ **and** $(u, v) \in E$ **then**
- 6 $\mathcal{D}_1(S) = \mathcal{D}_1(S) \cup (u, v)$

output: $\mathcal{D}_1(S)$

defined as:

$$MIOA(\mathcal{G}, u, \theta) = \bigcup_{v \in V, p(MIP(\mathcal{G}, u, v)) \geq \theta} MIP(\mathcal{G}, u, v). \quad (8)$$

$MIOA(\mathcal{G}, u, \theta)$ is defined as the union of MIP 's from u to all other nodes in V . MIP 's with propagation probabilities less than a threshold θ are not included to reduce the size of $MIOA$. One can think of $MIOA(\mathcal{G}, u, \theta)$ as a *local region* where u can spread its influence to. $MIOA(\mathcal{G}, u, \theta)$ can be computed by first finding the Dijkstra tree rooted at u with edge weight $-\log(p(u, v))$ for edge (u, v) , and then removing the paths whose cumulative weights are too high. By tuning the parameter θ , influence regions of different sizes can be obtained. For a single node, its $MIOA$ is clearly a tree. For multiple seed nodes, we build upon the idea of local influence region and propose two algorithms.

B. Building DAGs from a Seed Set

DAG 1: We observe that any DAG has at least one topological ordering. Conversely, given a topological ordering, if only edges going from a node of low rank to one with high rank are allowed, the resulting graph is a DAG.

To obtain the topological ordering given seed set S , we first introduce a (virtual) super root node R that is connected to all seed nodes with edge probability 1. Let $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$ where $V_{\mathcal{G}_R} = V \cup \{R\}$ and $E_{\mathcal{G}_R} = E \cup \{(R, S_k) | \forall S_k \in S\}$. We build $MIOA(\mathcal{G}_R, R, \theta)$ by calculating a Dijkstra tree from R . After removing R and its edges from $MIOA(\mathcal{G}_R, R, \theta)$, we obtain a singly connected DAG $\mathcal{D}_1 = (V_{\mathcal{D}_1}, E_{\mathcal{D}_1})$ on which BP algorithms can be directly applied and used to estimate the influence spread from S . However, $\mathcal{D}_1(\cdot)$ is very sparse (with $n - k$ edges) since many edges are removed.

We then augment $\mathcal{D}_1(\cdot)$ with additional edges. Note that $MIOA(\mathcal{G}_R, R, \theta)$ provides a topology ordering. More specifically, let the rank of node v be the sum weight of the shortest path from R , namely,

$$r(v) = \min(-\log(p(P(s, v)))), \forall s \in S. \quad (9)$$

Rank grows as the node is further away from R . We include in $\mathcal{D}_1(\cdot)$ all edges in \mathcal{G} whose end points are in $\mathcal{D}_1(\cdot)$ and go from a node with lower rank to one with higher rank. Clearly, the resulting graph is a DAG. The DAG constructing procedure is illustrated in Figure 1 and summarized in Algorithm 4.

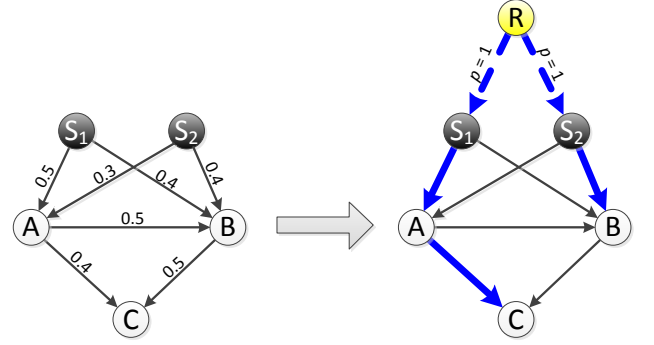


Fig. 1: DAG due to Algorithm 4. S_1 and S_2 are seed nodes. Edges in $MIOA(\mathcal{G}_R, R, \theta)$ are in bold. (S_1, B) , (S_2, A) , (A, B) , and (B, C) are added into $\mathcal{D}_1(S)$ to improve inference accuracy. $\theta = 0.0001$.

Algorithm 5: Calculate $\mathcal{D}_2(S)$ from a seed set S

input : $\mathcal{G}, S, MIOA(\mathcal{G}, v, \theta), \forall v \in V$

- 1 $\mathcal{D}_2(S) = \bigcup_{s \in S} MIOA(\mathcal{G}, s, \theta)$
- 2 Calculate $r(v), \forall v \in V_{\mathcal{D}_2}$ (Eq. (9))
- 3 **foreach** $(u, v) \in \mathcal{D}_2(S)$ **do**
- 4 **if** $r(u) \geq r(v)$ **then**
- 5 $\mathcal{D}_2(S) = \mathcal{D}_2(S) \setminus (u, v)$

output: $\mathcal{D}_2(S)$

DAG 2: In the second algorithm, we first compute the $MIOA$ from each seed node and take the union of $MIOA(\mathcal{G}, s, \theta), \forall s \in S$. Denote the resulting graph $\mathcal{D}_2(S) = (V_{\mathcal{D}_2}, E_{\mathcal{D}_2})$. Note that $\mathcal{D}_2(S)$ is not necessary a DAG as there could be circles. To break the cycles, certain edges need to be removed. We adopt a similar approach as in Algorithm 4. A node v is associated with a rank $r(v)$ as in (9). Only edges that connect a lower ranked node to higher ranked node are retained. Clearly, the resulting graph is a DAG. The approach is summarized in Algorithm 5.

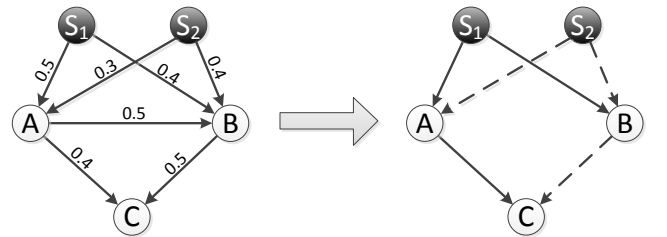


Fig. 2: DAG due to Algorithm 5. S_1 and S_2 are seed nodes. $\mathcal{D}_2(S)$ is the union of $MIOA(\mathcal{G}, S_1, \theta)$ (solid edges) and $MIOA(\mathcal{G}, S_2, \theta)$ (dashed edges). $\theta = 0.0001$.

The next proposition provides the relationship between DAGs constructed by Algorithm 4 and 5.

Proposition 1: Given a fixed influence threshold θ , let $\mathcal{D}_1(\cdot) = (V_{\mathcal{D}_1}, E_{\mathcal{D}_1})$ and $\mathcal{D}_2(\cdot) = (V_{\mathcal{D}_2}, E_{\mathcal{D}_2})$ be the DAGs constructed by Algorithm 4 and Algorithm 5. Then, $V_{\mathcal{D}_1} = V_{\mathcal{D}_2}$ and $E_{\mathcal{D}_2} \subseteq E_{\mathcal{D}_1}$.

Computation complexity: Building the Dijkstra tree from a source node takes $O(n_0 \log n_0)$, where n_0 is the maximum number of vertices in the resulting DAG. Calculating the node rank $r(\cdot)$ takes $O(n_0)$, the union operation in DAG 2 takes $O(n_0 - 1)$, and the edge augmenting and pruning in DAG 1 and DAG 2 takes $O(m_0)$ and $O(\min(m_0, k(n_0 - 1)))$, respectively, where m_0 is the maximum number of edges in a DAG and k is the seed set cardinality.

Thus, the running time of DAG 1 and DAG 2 are $O(n_0 \log n_0)$ and $O(n_0)$, respectively. Note that DAG 2 calculation requires the availability of $MIOA(\mathcal{G}, v, \theta), \forall v \in V$ first, which can be built at the initialization stage at the cost of $O(nn_0 \log n_0)$. Assuming that k is small and θ is properly selected, we have $n_0 \ll n$.

VI. OPTIMIZATION OF SEED SELECTION

In each round of Naive Greedy, a seed node with the maximum incremental spread-cost ratio is selected, namely, $v = \max_{v \in V \setminus S} \delta(v)$. Recall that $\delta(v) = (\sigma(S \cup v) - \sigma(S))/c(v)$ is the spread increment ratio of v under S . Initially, when $S = \emptyset$, $\delta(v) = \sigma(v)/c(v)$. Evaluating $\delta(v)$ at each iteration for all $v \in V$ dominates the overall computation complexity.

To accelerate the execution of Naive Greedy, one can try to improve on two aspects, namely, 1) limiting the candidate set of nodes to pick from for the next seed, and 2) reducing the complexity of computing the spread increments. CELF algorithm [5] eliminates many nodes from being evaluated. We focus on the second aspect. The proposed mechanism can be used in conjunction with the idea from CELF.

Recall in Section V-A, we use $MIOA$ to localize the influence region of a node. Consider for now that influence from a node can only reach nodes in its $MIOA$. Then, we make the following claim.

Proposition 2: Given the current seed set S , adding u to S will not change the spread increment of v , namely, $\delta_S(v) = \delta_{S \cup u}(v)$ if $MIOA(\mathcal{G}, u, \theta)$ and $MIOA(\mathcal{G}, v, \theta)$ have no common vertex.

As a result of Proposition 2, each time we select a new seed, only the influence increments of nodes that have overlapping influence regions with the newly selected seed need to be re-evaluated. Formally, we define the set of Peer Seeds (PS) of a vertex $v \in V$ as follow:

$$PS(\mathcal{G}, v, \theta) = \{s \in V \mid MIOA(\mathcal{G}, s, \theta) \cap MIOA(\mathcal{G}, v, \theta) \neq \emptyset\}. \quad (10)$$

$PS(\mathcal{G}, v, \theta)$ can be computed efficiently just once at the beginning when all $MIOA(\mathcal{G}, v, \theta)$'s are available.

Combining the ideas of 1) limiting the region to be re-evaluated using PS , 2) limiting the set of nodes to pick from (adopted from CELF), and 3) picking nodes w.r.t its cost and the remaining budget (Algorithm 2), we have the complete procedure to determine the optimal seed set in Algorithm 6. Figure 3 gives the block diagram of the proposed algorithm.

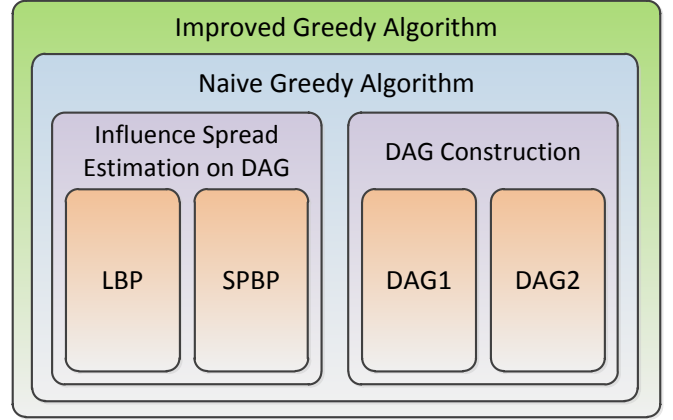


Fig. 3: The building blocks of our proposed algorithm. Details are presented in the previous sections.

The seed selection algorithm proceed as follow: In the initialization phase (lines 1 – 8), $MIOA$'s and PS 'es are constructed. The second candidate solution s_{max} can be determined in $O(n)$ time (line 9). S_1 is computed by executing the loop in lines 10 – 26. Each node in V is ranked by its incremental spread-cost ratio and can be added to S_1 just once. The node with the highest ratio is included in S_1 if it does not violate the budget b (line 12), and the corresponding nodes will be re-evaluated (lines 18 – 24). The procedure terminates once all nodes were considered, or no more budget remains (line 26). Finally, the algorithm compares the spread of S_1 , s_{max} and returns the solution with the larger spread.

Computation complexity: Recall that we denoted by n_0 the largest number of vertices, and by d the largest in-degree of a node in a DAG. For each node $v \in V$ in the initialization phase, building $MIOA(\mathcal{G}, v, \theta)$ takes $O(n_0 \log n_0)$, and estimating $\sigma(v)$ takes $O(n_0 d)$ using SPBP and $O(n_0 2^d)$ using LBP, respectively. Thus, depending on the algorithm used, the running time of initialization is $O(nn_0(\log n_0 + d))$ or $O(nn_0(\log n_0 + 2^d))$.

Let k be the number of seeds selected in the main loop (lines 10 – 26) and v_0 be the cardinality of the largest set of peer seeds, namely, $v_0 = \max_{v \in V} \{|PS(\mathcal{G}, v, \theta)|\} = O(n_0)$. Therefore, nodal influence spread is updated $O(kn_0)$ times. Note that this is much less than the number of updates required by Algorithm 1 ($O(n^2)$) as we do not naively re-evaluate every node. Each time when the influence spread is updated, we need to rebuild the DAG (line 20 – takes $O(n_0 \log n_0)$ with DAG 1 or $O(n_0)$ with DAG 2) and calculate the influence spread (line 21 – takes $O(n_0 2^d)$ with LBP or $O(n_0 d)$ with SPBP). The total computation complexity for different combinations of algorithms is summarized as follows:

	DAG 1	DAG 2
LBP	$n_0(n + kn_0)(\log n_0 + 2^d)$	$n_0(2^d(kn_0 + n) + n \log n_0)$
SPBP	$n_0(n + kn_0)(\log n_0 + d)$	$n_0(n(\log n_0 + d) + kn_0(1 + d))$

Clearly, combining DAG 1 and LBP incurs the highest complexity while the combination of DAG 2 and SPBP is the

Algorithm 6: The Proposed Algorithm

input : network graph $\mathcal{G}(V, E)$ and budget b

// initialization

1 $S = S_1 = s_{max} = \emptyset, \sigma_0 = 0, \theta = \text{influence threshold}$

2 **foreach** $v \in V$ **do**

3 build $MIOA(\mathcal{G}, v, \theta)$

4 $\mathcal{D}(v) = MIOA(\mathcal{G}, v, \theta)$

5 calculate $\sigma(v)$ (LBP or Algorithm 3)

6 $\delta(v) = \sigma(v)/c(v)$

7 $\delta_{old}(v) = 0$

8 build $PS(\mathcal{G}, v, \theta), \forall v \in V$

// select s_{max}

9 $s_{max} = \arg \max_{v \in V} \sigma(v)$

// select S_1

10 **while true do**

 // select a new seed

11 $u = \arg \max_{v \in V \setminus S} (\delta(v))$

12 **if** $c(S_1 \cup u) \leq B$ **then**

13 $S_1 = S_1 \cup \{u\}$

14 $\sigma_0 = \sigma(S)$

15 $\delta_{old}(v) = \delta(v), \forall v \in V \setminus S_1$

16 $b = b - c(u)$

 // update incremental influence spread

17 $\delta_{max} = 0$

18 **foreach** $v \in PS(\mathcal{G}, u, \theta) \setminus S_1$ **do**

19 **if** $\delta_{old}(v) > \delta_{max}$ **then**

20 build $\mathcal{D}(S_1 \cup \{v\})$ (Algorithm 4 and 5)

21 calculate $\sigma(S_1 \cup \{v\})$ (LBP or Algorithm 3)

22 $\delta(v) = (\sigma(S_1 \cup \{v\}) - \sigma_0)/c(v)$

23 **if** $\delta(v) > \delta_{max}$ **then**

24 $\delta_{max} = \delta(v)$

25 $V = V \setminus u$

26 **if** $V = \emptyset$ **or** $b = 0$ **then**

 break

27 $S = \arg \max (\sigma(S_1), \sigma(s_{max}))$

output: selected seed set S

fastest. From the analysis, it is easy to see that the computation complexity depends on n_0 and d . The proposed approach is more efficient with smaller n_0 and d ; that is, when the graph is sparse and the edge propagation probabilities are small, both are likely true in social networks.

VII. EVALUATION

In this section, we evaluate the performance of the proposed framework. First, an illustrative example is provided to highlight the difference in the two DAG construction models, and spread computation methods. Next, we present the implementation details and experimental setup. Finally, we present the results on 1) performance on real-world social networks and 2) impact of network structures using synthetic graphs.

A. An Illustrative Example

Here, we consider a small scale network as shown in Figure 4(a). Figure 4(b) and (c) show the DAG constructed by the two models, and the activation probabilities by the

two methods. DAG 1 retains all the edges in the network (since the original graph is in fact a DAG), while DAG 2 has fewer edges. When LBP is used to compute the influence spread (the numbers on top next to each node), DAG 1 yields higher activation probability compared to DAG 2 for node B and node C since A has a large influence to B (0.5), which is not considered in DAG 2. In both DAGs, ignoring the possible correlation among parent nodes in SPBP, the activation probabilities tend to be bigger. Interestingly, though DAG 2 is a multi-connected graph, the activation probabilities computed by both methods are identical. Upon a close examination, we find that even though the graph is multi-connected, the activations of A and B are in fact independent since both are direct descendents of seed nodes with activation probability one.

B. Experiment Setup

The algorithms and implementation: In addition to the two DAG models and two methods to compute influence spread (a total of 4 combinations DAG1-LBP, DAG1-SPBP, DAG2-LBP, and DAG2-SPBP), we make comparison with the following algorithms:

- **PMIA**(θ) [4]: a very fast heuristic algorithm that builds a tree-like structure model on which influence is spread. θ is the influence threshold. We will set $\theta = 1/160$ in all experiments as it was reported to yield the best performance. The PMIA implementation provided by the authors is optimized for IM, and thus its performance for BIM is excluded.
- **Greedy/CELF**: The greedy approach from [3] with CELF optimization in [5]. The number of simulation rounds for each $\sigma(\cdot)$ estimation is 10,000.
- **Weighted Degree**: The simple heuristic that selects k seeds that have maximum total out-connection weight. Weighted Degree has been reported to be working very well in practice.

We do not compare with other heuristics such as SP1M, SPM [28], PageRank [29], Random, DegreeDiscountIC [7] or Betweenness centrality [30] since they have been reported in previous studies [4], [3] to be either unscalable or have poorer performance.

We have implemented the proposed algorithms in C++. All experiments are conducted on a workstation running Ubuntu 11.04 with an Intel Core i5 CPU and 2GB memory. In order to implement LBP algorithm, we use libDAI [31] and Boost [32] libraries. We find out through the implementation that running LBP on networks with high in-degree nodes is very costly. Therefore when running LBP, we prune incoming edges on high in-degree nodes such that only ten edges with the highest propagation probabilities are retained. The implementation of PMIA is obtained from its authors. Note that with code optimization, the running time of our algorithms can be further reduced.

Datasets: We use four real-world network datasets from [33] and [34] to compare performance of different algorithms. The four datasets were selected so as they are representative of the

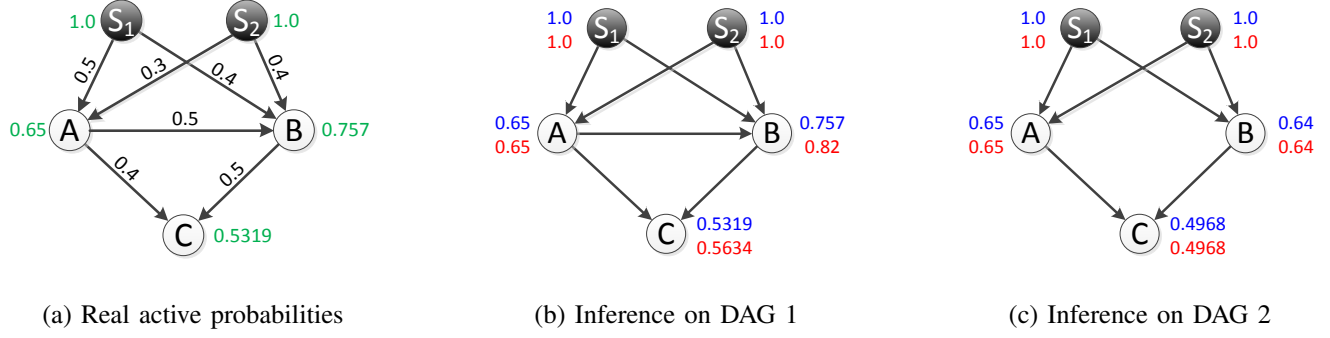


Fig. 4: Inference result on 2 DAG models. The real active probabilities are in green, LBP results are on top, in blue, and SPBP results are below, in red.

TABLE II: Network datasets

Name	Type	Nodes	Edges	Density	Max Degree	Mean Degree	Description
<i>Email</i>	Email exchange network	447	5,731	0.04	195	25.64	Email communication within a research lab during a year
<i>p2p-Gnutella</i>	P2P network	6,301	20,777	1e-03	97	6.59	Gnutella peer to peer network from August 8 2002
<i>soc-Slashdot</i>	Social network	82,168	948,464	1.6e-03	5,064	23.09	Slashdot social network from February 2009
<i>Amazon</i>	Product co-purchasing network	262,111	1,234,877	2.6e-05	425	9.42	Amazon product co-purchasing network from March 2 2003

structural features of large-scale social networks, and are of different scales – from several thousands to millions of edges. The first one is an email exchange network in a research lab, denoted by *Email*. Each researcher is a vertex and an email from a researcher u to v constitutes an edge. The second network, denoted by *p2p-Gnutella* is a snapshot of the Gnutella peer-to-peer file sharing network from August 2002. Nodes represent hosts in the Gnutella network and edges represent connections between the Gnutella hosts. The third network comes from Slashdot.org, a technology-related news website, denoted by *soc-Slashdot*. In 2002, Slashdot introduced the Slashdot Zoo feature that allows users to tag each other as friends or foes. The network contains friend/foe links between Slashdot users obtained in February 2009. Finally, *Amazon* dataset is the product co-purchasing network collected by crawling Amazon website on March 2, 2003. Details of the datasets are summarized in Table II.

In addition to real social networks, we modified DIGG [35] source code and generated scale-free networks with different network densities and node out-degree distributions. The purpose which allows us to study the impact of graph structures and network property on the algorithm performance.

Probability generation models: Two models that have been used in previous work [3], [4] are: 1) the Weighted Cascade (WC) model where $p(u, v) = 1/d(v)$ where $d(v)$ is the in-degree of v and 2) the Trivalency (TV) model where $p(u, v)$ is assigned a small value for any $(u, v) \in E$. We argue that both models are not truthful reflections of the probability model in practice. The WC model assign a very high probability for a connections to nodes with small number of incoming connections while the TV model assigns a similar probability to all edges. In the evaluation, we consider two additional

models: 1) Random (RA) where $p(u, v)$ is randomly selected in the range $[0.001, 0.2]$. RA is useful when no prior information regarding the influence is available; and 2) Power Law (PL) where $p(u, v)$ follows the power law distribution with the density function $p(x) = \alpha/x^\beta$, with x be the propagation probability between two random edges $p(u, v)$. Parameters $\alpha = 0.05$ and $\beta = 0.9$ are selected so that $p(u, v)$ has the mean value 0.1 in the range $[0.001, 0.2]$.

C. Real Social Networks

Unit-cost version of BIM: BIM with unit-cost is the traditional IM problem where the seed set size k is fixed. In this experiment, we run 7 algorithms: Greedy, PMIA, Weighted Degree, and the 4 proposed methods on 4 datasets presented in Table II. k varies from 1 to 50, and we adopt the RA probability generation model.

Figure 5 shows the influence spread generated by the best seed sets in different algorithms as the seed size changes. Since Greedy does not scale with large datasets, we only run Greedy on *Email* and *p2p-Gnutella*. The influence spread from the seed set selected by each algorithm is determined by 10,000 rounds of Monte Carlo simulations on the original graphs.

In Figure 5(a), the performance of DAG1-LBP and Greedy (known to be within a constant ratio of the optimal) are not distinguishable (and thus are represented in one curve). The influence spread of DAG1-SPBP and DAG2-LBP/SPBP are shortly behind, all outperforming PMIA and Weighted Degree. We observe on *Email* dataset (a small but dense network) that both the structure of the DAG (DAG 1 vs. DAG 2) as well as the BP algorithm used (LBP vs. SPBP) will affect performance of the proposed methods. In contrast, as shown in Figure 5(b) – (d), the influence spreads of the four approaches DAG1/2–

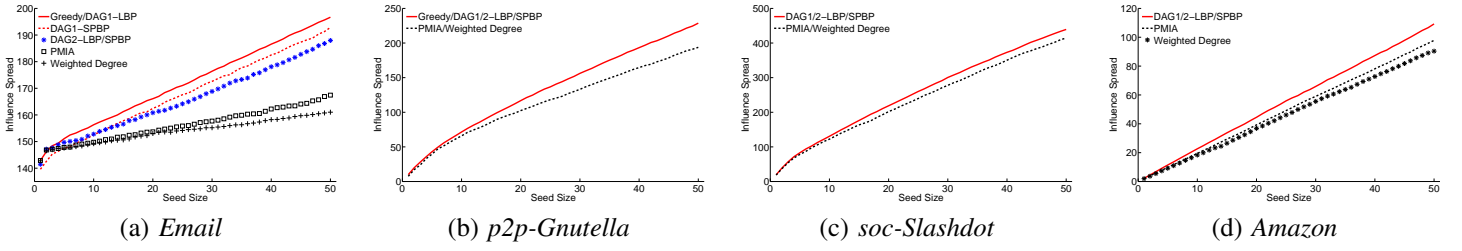


Fig. 5: Influence spread with node unit-cost on 4 datasets. DAG 1 results are in red curves, DAG 2 are in blue curves, and other methods are in black curves.

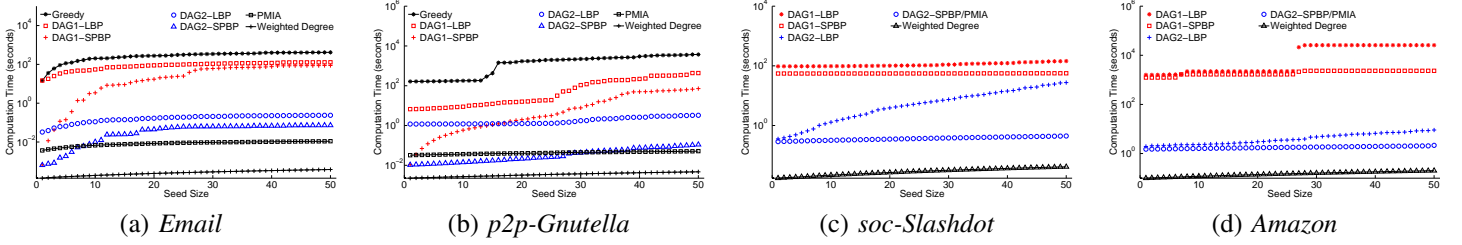


Fig. 6: Computation time with node unit-cost on 4 datasets.

LBP/SPBP are identical for sparser networks, and is the same as Greedy in *p2p-Gnutella* dataset.

In terms of running time, Weighted Degree is the fastest. Among the four proposed approaches, DAG2-SPBP is the fastest, next are DAG2-LBP, DAG1-SPBP, and finally DAG1-LBP. DAG2-SPBP and PMIA have comparable order in running time with DAG2-SPBP being 30-40% slower than PMIA in most cases. Again, this may be primarily attributed to the lack of code optimization in our proposed methods.

Interestingly, influence spread on *Amazon* grows linearly with the seed size. Our result matches with that in [4]. This can be explained by the sheer scale of the network, and thus the small number of selected seeds are likely to have non-overlapping influence regions.

General cost version of BIM: In this set of experiments, we compare only 4 algorithms: Greedy/CELF, Weighted Degree, and DAG1/DAG2-SPBP on 4 datasets presented in Table II. We also omit the two methods that use LBP (DAG1/DAG2-LBP) from the result since they have comparable performance as the SPBP approaches. The budget $b = \{10, 15, 25, 40, 60, 100\}$, and the RA probability generation model is used. Nodal costs are selected uniformly in $[1.0, 3.0]$.

Results in Figures 7 and 8 are similar to that in Figures 5 and 6. In most cases, DAG1 has better performance compared to DAG2. Notably, DAG1-SPBP outperforms Greedy/CELF on *p2p-Gnutella* dataset. Figure 8 shows that the proposed methods are several orders of magnitude faster than Greedy/CELF. Weighted Degree while being the fastest algorithm, does not perform nearly as well as the others on a dense graph (*Email*).

Comparison of Influence Spread on Two DAG Models:

To understand the behavior of the proposed algorithms, we conduct further experiments on *Email* dataset as it gives the largest performance difference between the algorithms.

Figure 9(a) gives the number of vertices and edges as the

result of the two DAG models with varying sizes of seed sets. Since both have the same number of vertices, only one curve is shown. It is clear that DAG1 is much denser than DAG2 due to the inclusion of more edges. As the seed set grows, the gap becomes smaller.

We use Root Mean Square Error (RMSE) to compare the activation probabilities on nodes. RMSE is defined as,

$$RMSE(p, p') = \sqrt{\frac{\sum_{v \in V} (p'(v) - p(v))^2}{n} / \frac{\sum_{v \in V} p(v)}{n}},$$

where $p'(\cdot)$ is the inferred result from the propose algorithms. The ground truth $p(\cdot)$ is determined by Monte Carlo simulations. When $p'(v) = p(v), \forall v \in V$ then $RMSE(p, p') = 0$.

Figure 9(b) shows that DAG1 has smaller RMSE since it constructs a denser graph. More edges clearly improves the quality of the seed selection process. In the comparing LBP and SPBP, LBP is slightly better since SPBP ignores the correlation among node states. The combination of DAG1 and LBP yields the best inference result, but incurs higher computation complexity. The results are consistent with those in Figure 5(a).

D. Synthetic Networks

In this section, we conduct three sets of experiment with 5 methods: CELF, PMIA, Weighted Degree and DAG1/2-SPBP. Synthetically generated networks are used to study the impact of network structures and probability generation models on performance of the algorithms. To isolate the effects of network properties, we only consider the unit cost BIM problem.

Impact of network density: Results from Figure 5 and 7 indicate that our proposed methods perform best on dense networks (*Email* and *p2p-Guntella*). To further validate this observation, we generate 4 networks with 20k, 50k, 100k, and 200k edges using DIGG [35]. The number of vertices is fixed at 5,000. Seed set size $k = 50$ and probability model is RA. We

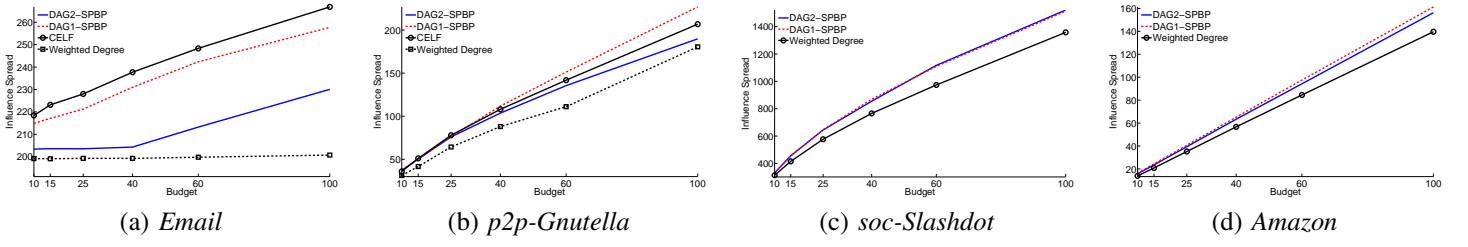


Fig. 7: Influence spread with random node costs on 4 datasets.

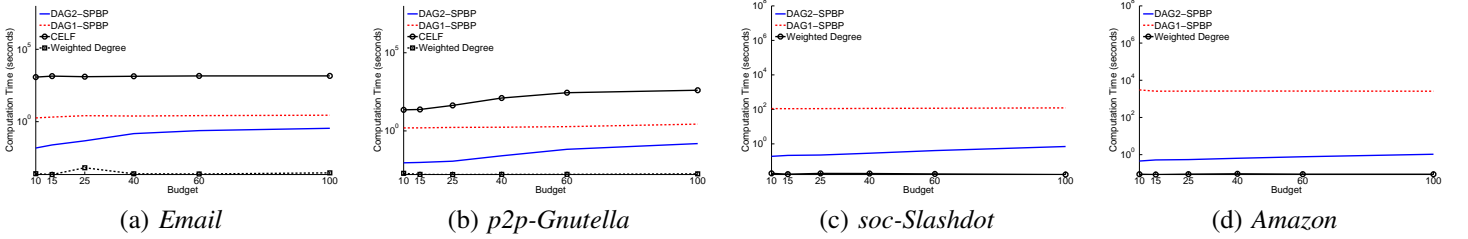
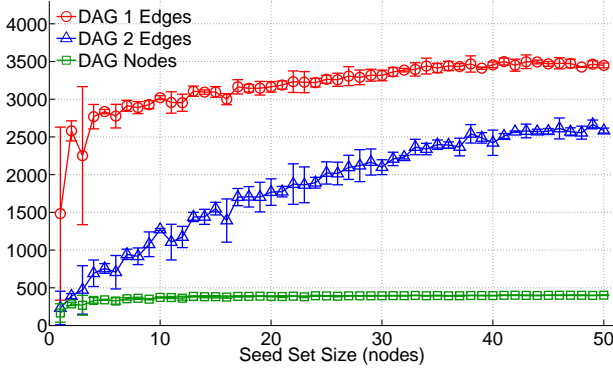
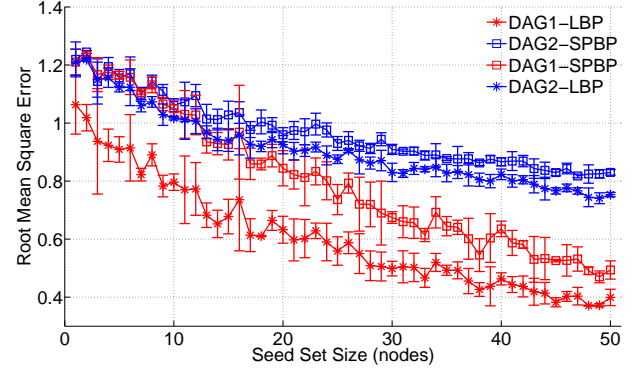


Fig. 8: Computation time with random node costs on 4 datasets.



(a) Number of nodes and edges in DAG



(b) RMSE

Fig. 9: Size of DAGs and RMSE of activation probabilities. Results are averages of 50 runs with different seed selections and symmetric error bars indicate standard deviations.

evaluate the spread ratio of various algorithms, defined as the ratio of the spread attained to that by Greedy/CELF algorithm. From Figure 10(a), as the network density increases, the performance gap between the proposed algorithms and existing algorithms including CELF increases. CELF relies on many rounds of simulations to determine the spread. For dense networks, more rounds of simulations are needed to produce a spread estimation that is close enough to the *ground truth*. As a result, with a fixed number of simulation rounds, CELF has worse performance at high network densities. We also observe that PMIA, which was designed to take advantage of network sparsity; and Weighted Degree, which only uses local node information, do not perform well on densely connected graphs.

Impact of probability generation model: In this set of experiments, we run 5 algorithms on a synthetic network with 5,000 nodes and 50,000 edges. Each algorithm selects a seed set with size $k = 50$ under 4 propagation probability models: RA, TV, PL and WC. All models give similar performance except Weighted Degree on WC model. Recall that WC

generates the propagation probabilities based on the in-degree of nodes, thus strong connections are established between nodes with low in-degree. Weighted Degree can't "see" those strong ties beyond the local edges, and therefore, has the worst performance.

Impact of node out-degree distribution: It is known that node out-degree in real social networks follows the power-law distribution [36]. Let y be the percentage of nodes with degree x , then we have $y \sim \alpha/x^\beta$. α and β can be seen as the intercept and the (negative) slope when degree sequence is plotted on a log-log scale. While varying α only scales the distribution up or down, changing β alters the "shape" of the distribution. More specifically, a high value of β means the node out-degree distribution exhibits larger skew. The network in this case contains few "hubs" that are connected to many other nodes. On the other hand, a small β means that the distribution is fat-tailed and the max out-degree in the network is not much larger than the average out-degree. We run 5 algorithms to solve the unit-cost BIM problem on 4 generated

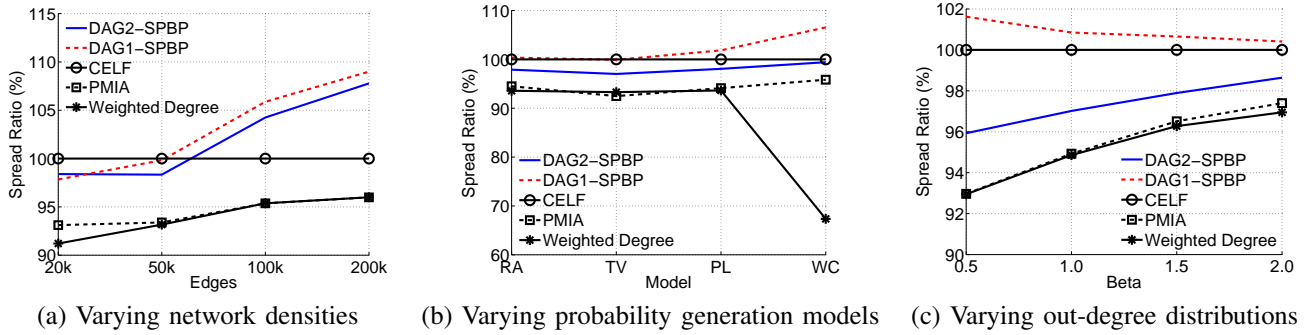


Fig. 10: Algorithm performance on different network conditions.

networks with $\beta = \{0.5, 1.0, 1.5, 2.0\}$. The network size is 5,000 and α is adjusted accordingly such that the total number of edges is roughly 50,000. We see from Figure 10(c) that the performance gap among the algorithms reduces with larger β . This is because with a large degree distribution skewness, nodes with high out-degree (hub) will almost certainly be one of the best seed candidates (unless their costs are too high, which is not this case). Simple algorithms such as Weighted Degree can easily identify such hub nodes. On the other hand, when the network is more “flattened”, more sophisticated algorithms are necessary.

E. Summary

From the experiments results, Weighted Degree gives the best efficiency in terms of spread/complexity. However, its performance degrades significantly on dense networks or more heavy tailed power law graphs. The same conclusion is applied to PMIA. Even though being faster than our algorithms, PMIA shows little improvement in term of attainable spread compared to Weighted Degree, except under the WC model. Our proposed schemes surpass the others in all the experimented datasets. They also offer more application flexibility: one would apply the best performed algorithm (DAG1-LBP) on static networks (e.g.: network of connections between co-workers) to identify the most influential nodes, or apply the fastest algorithm (DAG2-SPBP) on rapidly changing communities (e.g.: network of connections between people in a social group) to obtain immediate result.

VIII. CONCLUSION

While recent researches focus on solving the IM problem, we considered in this paper the BIM problem, which is a generalization of the former one. The study on real world datasets and synthetic datasets with controllable network parameters provides convincing evidences the proposed algorithms have superior performance. Furthermore, we gain some insights on the choice of algorithms in trading computation complexity with performance given the network structure.

IX. ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under grants CNS-1117560 and CNS-0832084.

REFERENCES

- [1] P. Domingos and M. Richardson, “Mining the network value of customers,” in *Proceedings of the seventh ACM SIGKDD KDD '01*. New York, NY, USA: ACM, 2001, pp. 57–66.
- [2] M. Richardson and P. Domingos, “Mining knowledge-sharing sites for viral marketing,” in *Proceedings of the eighth ACM SIGKDD KDD '02*. New York, NY, USA: ACM, 2002, pp. 61–70.
- [3] D. Kempe, J. Kleinberg, and E. Tardos, “Maximizing the spread of influence through a social network,” in *Proceedings of the KDD '03*. New York, NY, USA: ACM, 2003, pp. 137–146.
- [4] W. Chen, C. Wang, and Y. Wang, “Scalable influence maximization for prevalent viral marketing in large-scale social networks,” in *Proceedings of the KDD '10*. New York, NY, USA: ACM, 2010, pp. 1029–1038.
- [5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, “Cost-effective outbreak detection in networks,” in *Proceedings of the 13th ACM SIGKDD KDD '07*. New York, NY, USA: ACM, 2007, pp. 420–429.
- [6] A. Goyal, W. Lu, and L. V. Lakshmanan, “Celf++: optimizing the greedy algorithm for influence maximization in social networks,” in *Proceedings of the WWW '11*. New York, NY, USA: ACM, 2011, pp. 47–48.
- [7] W. Chen, Y. Wang, and S. Yang, “Efficient influence maximization in social networks,” in *Proceedings of the KDD '09*. New York, NY, USA: ACM, 2009, pp. 199–208.
- [8] W. Chen, Y. Yuan, and L. Zhang, “Scalable influence maximization in social networks under the linear threshold model,” in *Proceedings of the 2010 IEEE ICDM '10*, 2010, pp. 88–97.
- [9] E. Even-Dar and A. Shapira, “A note on maximizing the spread of influence in social networks,” *Inf. Process. Lett.*, vol. 111, pp. 184–187, January 2011.
- [10] J. Sylvester, “Maximizing diffusion on dynamic social networks,” *Science*, 2009.
- [11] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan, “Influence maximization in social networks when negative opinions may emerge and propagate,” in *SDM*. SIAM / Ominpress, 2011, pp. 379–390.
- [12] N. Chen, “On the approximability of influence in social networks,” in *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '08. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008, pp. 1029–1037.
- [13] E. Ackerman, O. Ben-Zwi, and G. Wolfvitz, “Note: Combinatorial model and bounds for target set selection,” *Theor. Comput. Sci.*, vol. 411, pp. 4017–4022, October 2010.
- [14] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, “Approximation analysis of influence spread in social networks,” *CoRR*, vol. abs/1008.2005, 2010.
- [15] Y. Ni, L. Xie, and Z.-Q. Liu, “Minimizing the expected complete influence time of a social network,” *Inf. Sci.*, vol. 180, pp. 2514–2527, July 2010.
- [16] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, “Epidemic thresholds in real networks,” *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 4, pp. 1:1–1:26, Jan. 2008.
- [17] H. Tong, B. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. Chau, “On the vulnerability of large graphs,” in *IEEE 10th ICDM*, dec. 2010, pp. 1091–1096.
- [18] B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos, “Virus propagation on time-varying networks: theory and immunization algorithms,” in *Proceedings of the 2010 ECML PKDD: Part III*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 99–114.

- [19] H. Nguyen and R. Zheng, "Influence spread in large-scale social networks — a belief propagation approach," in *Proceedings of the ECML PKDD'12*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 515–530.
- [20] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," in *Inf. Proc. Letters*, vol. 70, no. 1, pp. 39–45, 1999.
- [21] C. Chekuri and A. Kumar, "Maximum coverage problem with group budget constraints and applications," in *Proceedings of the 7th APPROX*, 2004, pp. 72–83.
- [22] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 239–269.
- [23] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conf. on AI*, Pittsburgh, PA, 1982, pp. 133–136.
- [24] S. L. Lauritzen and D. J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 50, no. 2, 1988.
- [25] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy Belief Propagation for Approximate Inference: An Empirical Study," in *In Proceedings of Uncertainty in AI*, 1999, pp. 467–475.
- [26] B. J. Frey, R. Koetter, and N. Petrovic, *Very loopy belief propagation for unwrapping phase images*. MIT Press, 2001, vol. 14, p. 737743.
- [27] R. McEliece, D. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of pearl's ldquo;belief propagation rdquo; algorithm," in *JSAC*, vol. 16, no. 2, pp. 140–152, feb 1998.
- [28] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *Knowledge Discovery in Databases: PKDD 2006*, 2006, vol. 4213, pp. 259–271.
- [29] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 107–117, April 1998.
- [30] L. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.
- [31] J. M. Mooij, "libDAI: A free and open source C++ library for discrete approximate inference in graphical models," *Journal of Machine Learning Research*, vol. 11, pp. 2169–2173, Aug. 2010.
- [32] Boost c++ libraries. [Online]. Available: <http://www.boost.org/>
- [33] J. Leskovec. Stanford large network dataset collection. [Online]. Available: <http://snap.stanford.edu/data/>
- [34] I. Wiki. Social network generation. [Online]. Available: http://www.infovis-wiki.net/index.php/Social_Network_Generation
- [35] L. Dignan. (2006) Dynamic graph generator. [Online]. Available: <http://diggs.cs.tufts.edu/>
- [36] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, vol. 51, no. 4, pp. 661–703, Nov. 2009.
- [37] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979. [Online]. Available: <http://link.aip.org/link/?SMJCAT/8/410/1>
- [38] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," Carnegie Mellon University - School of Computer Science, Tech. Rep. CMU-CALD-05-103, June 2005.

APPENDIX

Proof of Theorem 1:

Proof: The proof is an adaption of the proof in [4] and Valiant's original proofs of the #P-completeness of the s - t connectedness in a direct graph [37]. First, we define a few problems that are known or to be proven to be #P-complete.

Definition 4: (SAT)

Input: $F = c_1 \wedge c_2 \wedge \dots \wedge c_r$, where $c_i = (y_{i1} \vee y_{i2})$ and $y_{ij} \in X$,
Output: $|\{(x, t) | t = (t_1, t_2, \dots, t_n) \in \{1, 2\}^n; \text{ for } 1 \leq i \leq r, x \text{ make } y_{i,k} \text{ true for } k = t_i.\}|$

Definition 5: (S-SET CONNECTEDNESS on DAG)

Input: A DAG $\mathcal{D} = (V, E); s \in V; V' \in V$.

Output: Number of subgraphs of \mathcal{D} in which for each $u \in V'$, there is a (directed) path from s to u .

Definition 6: (S-T CONNECTEDNESS on DAG)

Input: A DAG $\mathcal{D} = (V, E); s, t \in V$.

Output: Number of subgraphs of \mathcal{D} in which there is a directed path from s to t .

To prove Theorem 1, we first establish the following lemma.

Lemma 1: $SAT' \leq_p$ S-T CONNECTEDNESS on DAG.

Proof: Given F construct a DAG $\mathcal{D} = (V, E_1 \cup E_2)$ where $V = \{c_1, c_2, \dots, c_{r+1}, x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n, s\}$, $E_1 = \{(x_i, c_j) | x_i \text{ appears in clause } c_j \text{ in } F\} \cup \{(x_n, c_{r+1}), (\bar{x}_n, c_{r+1})\}$, and $E_2 = \{(x_i, x_{i+1}), (\bar{x}_i, x_{i+1}), (\bar{x}_i, \bar{x}_{i+1}), (x_i, \bar{x}_{i+1}) | 1 \leq i \leq n\} \cup \{(s, x_1), (s, \bar{x}_1)\}$. The direction of each edge follows the order of the pairs. \mathcal{D} is a DAG as edges only go from x 's of smaller index to larger ones, and from x 's to c 's. Note the \mathcal{D} is multi-connected. The rest of the proof follows that in [37]. ■

Theorem 1 can then be proved using the same argument as in [4] with the exception that the reduction is from the S-T CONNECTEDNESS on DAGs due to Lemma 1. ■

Proof of Proposition 1:

Proof: In both algorithms, a node v is not included in the DAG if and only if $r(v) > \theta$. Thus, $V_{\mathcal{D}_1} = V_{\mathcal{D}_2}$.

To show $E_{\mathcal{D}_2} \subseteq E_{\mathcal{D}_1}$, it suffices to show that $\forall (u, v) \in E_{\mathcal{D}_2}$, $(u, v) \in E_{\mathcal{D}_1}$. Since $(u, v) \in E_{\mathcal{D}_2}$, $(u, v) \in E$ and $r(u) \leq r(v)$. Therefore, according to Algorithm 2, $(u, v) \in E_{\mathcal{D}_2}$. Clearly, the converse is not true as some edges in $E_{\mathcal{D}_1}$ may not be part of the MIOA from any seed node. ■

Proof of Proposition 2:

Proof: It is easy to see that by limiting the spread from u in MIOA(\mathcal{G}, u, θ), then $p(w), \forall w \in MIOA(\mathcal{G}, v, \theta)$ will not be affected by the inclusion of u in the seed set. ■

Proof of Theorem 2:

First we establish the following lemma. Let r be the number of iterations executed by the repeat loop in Algorithm 1. Let S be the current seed set and S^* be the optimal seed set. Without loss of generality, we may renumber nodes that was added to S follow the chronicle order $S = \{u_1, u_2, \dots, u_l\}$. Let $S_i = \bigcup_{j=1}^i u_j$ and let j_i be the index of the iteration in which u_i was considered.

Lemma 2: After each iteration $j_i, i = 1, \dots, l + 1$, the following holds:

$$\sigma(S_i) \geq \left[1 - \prod_{k=1}^i \left(1 - \frac{c(k)}{b} \right) \right] \sigma(S^*). \quad (11)$$

Proof: The proof of Lemma 2 was first presented by Khuller *et al.* in [20] for the budgeted maximum coverage problem, which is a special case of BIM where all the active edge probabilities are 1. Later, it was extended by Krause *et al.* (Lemma 3 in [38]) for general submodular functions. ■

Now we're in position to prove Theorem 2:

Proof: (Adapted from [20]) We prove Theorem 1 by case analyzing Algorithm 2.

- **Case 1:** If there exist at least a node $u \in V$ which has spread greater than $\frac{1}{2}\sigma(S^*)$, then u or any other nodes which possess a greater spread, will be selected as the second candidate S_2 . Algorithm 2 will therefore guarantee at least $\frac{1}{2}\sigma(S^*)$.

- **Case 2:** If there is no such node.

– Case 2.1: If $c(S) < \frac{1}{2}b$, then we have $c(u) > \frac{1}{2}b, \forall u \notin S$ since there is no more node that can be added to S without violating the budget constrain. W.l.o.g, we assume $S \neq S^*$. Therefore, $S^* \setminus S$ contains at most 1 node v , otherwise $c(S^*) > b$. By submodularity definition we have,

$$\begin{aligned} \sigma(S^* \cap S) + \sigma(v) &\geq \sigma((S^* \cap S) \cup v) + \sigma((S^* \cap S) \cap v) \\ &\geq \sigma(S^*) + \sigma(\emptyset) \\ &\geq \sigma(S^*). \end{aligned}$$

By assumption, we have $\sigma(v) < \frac{1}{2}\sigma(S^*)$, therefore $\sigma(S^* \cap S) \geq \frac{1}{2}\sigma(S^*)$. It follows that $\sigma(S) \geq \frac{1}{2}\sigma(S^*)$.

– Case 2.2: If $c(S) \geq \frac{1}{2}b$. We first observe that for $a_1, \dots, a_n \in \mathbb{R}$ and $\sum_{i=1}^n a_i \geq \alpha A$, the function,

$$\prod_{i=1}^n \left(1 - \frac{a_i}{A}\right)$$

is maximized when $a_i = \frac{\alpha A}{n}$. By Lemma 2, we have,

$$\begin{aligned} \sigma(S_i) &\geq \left[1 - \prod_{k=1}^i \left(1 - \frac{c(k)}{b}\right)\right] \sigma(S^*) \\ &\geq \left[1 - \left(1 - \frac{1}{2i}\right)^i\right] \sigma(S^*) \\ &\geq \left(1 - \frac{1}{\sqrt{e}}\right) \sigma(S^*). \end{aligned}$$

Thus, in the worst case, Algorithm 2 provides a $(1 - 1/\sqrt{e})$ -approximation. ■