Influence Spread in Large-Scale Social Networks – A Belief Propagation Approach

Huy Nguyen and Rong Zheng

University of Houston, Computer Science Department
4800 Calhoun Rd., Houston, TX, 77204
{hanguyen4,rzheng}@uh.edu

Abstract. Influence maximization is the problem of finding a small set of seed nodes in a social network that maximizes the spread of influence under a certain diffusion model. The Greedy algorithm for influence maximization first proposed by Kempe, later improved by Leskovec 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 this work, we tackle both problems by devising efficient algorithms to compute influence spread and determine the best candidate for seed selection. The fundamental insight behind the proposed algorithms is the linkage between influence spread determination and belief propagation on a directed acyclic graph (DAG). Experiments using real-world social network graphs with scales ranging from thousands to millions of edges demonstrate the superior performance of the proposed algorithms with moderate computation costs.

1 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. Another more recent example is the Hewlett-Packard (HP) TouchPad fire sale event [1]. The company slashed the price of TouchPad by 75% to clear out inventory. Without any mass media advertisement or public announcement, the move inadvertently generated an Internet phenomenon – with Twitter and Facebook users sharing tips on websites where the product was still in stock – and long lines at retailers as consumers jostled to pick up TouchPads.

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. Under a specific diffusion model, the goal of influence maximization (IM) is to find k vertices (seed nodes) in the graph such that the expected number of vertices influenced by the k seeds is maximized [2,3,4]. Kempe *et al.* 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, there are major limitations with this method as previously mentioned. Follow-up works either only addresses one of the deficiencies [5,6] or sacrifices accuracy for less computation time [7].

In this work, 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 previous selected seed need to be evaluated. Experimental study shows that the proposed algorithms can scale up to massive graphs with millions of edges with high accuracy. On real-world social network graphs, the proposed algorithms can achieve influence spread comparable to that by Greedy algorithm and incur significant less computation costs. They also outperform the scheme in [8] in achievable influence spread at the expense of marginal increase in computation time.

The main contributions of this paper are summarized as follows:

– 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.
– A fast algorithm is devised to incrementally update the DAG as more seeds are added, and select candidate seeds.

The rest of this paper is organized as follows. In Section 2, we give a comprehensive review of the existing literature on influence spread maximization. Section 3 presents theoretical results concerning influence spread on DAGs. In Section 4, we devise two heuristics to reduce a general directed graph into a DAG which captures the majority of influence spread. Improvements on seed selection are discussed in Section 5. In Section 6, extensive experiment results are presented. Finally we conclude the paper and discuss future research directions in Section 7.

2 Related Work

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 the CELF, 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 [7,8,9]. 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 [8] and [9], 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 3 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.* [10] 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 [11] 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.* [12] consider a new model that incorporates negativity bias and design an algorithm to compute influence on tree structures.

3 Influence Spread on Directed Acyclic Graphs

In this section, we consider the problem of computing influence spread given a fixed seed set when the underlying social network is a DAG. We first show the problem remains #P-hard, and then establish its equivalence to the computation of marginal probabilities in a Bayesian network.

3.1 Problem Formulation

We consider 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* [4] 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 $S \leq T$. 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.

3.2 Hardness of Computing Influence Spread on DAGs

In [4], 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, $p(x_u | x_{Par(x_u)}) = p(x_u)$. The joint distribution is thus given by,

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

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

$$\sigma(S) = \mathbb{E}(\sigma_C(S)) = \sum_{u \in V} \mathbb{E}(x_u) = \sum_{u \in V} p(u). \tag{2}$$

The second equality is due to the linearity of expectations. To compute $p(u)$, we can sum (1) 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 theorem[1].

Theorem 1. *Computing the influence spread $\sigma(S)$ on a DAG given a seed set S is #P-complete.*

3.3 Estimating $\sigma(\cdot)$ via Belief Propagation

Belief propagation 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 [13]. For *singly-connected* DAGs, where between any two vertices there is only one simple path, the belief propagation (BP) algorithm [14] 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 [13,15,16] have been shown to work well in general. Exact solutions such as junction tree [15] 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 Bayesian Network. For each factor in the graph or a Bayesian node, a conditional probability table (CPT) is constructed. For a node v with the parent set $Par(v) = \{par_1, par_2, \ldots, par_k\}$, its CPT consists of one column for each state and one row for each set of states its parents may assume. In influence spread, each state has two states: active (1) and inactive (0). Thus the number of rows in a CPT is 2^k. An illustrative example of a factor graph and one of its CPT's is given in Figure 1 and 2.

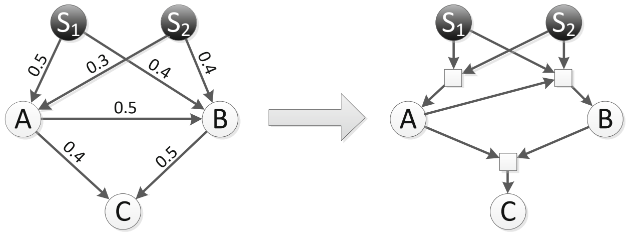


Fig. 1. Converting a DAG into a factor graph

Once the factor graph and CPT's associated with each factor are available, we can apply a suitable BP algorithm to calculate the active probability of each node in the DAG. $\sigma(\cdot)$ can then be determined by (2).

[1] All proofs are omitted due to lack of space but can be found on the full technical report at `http://arxiv.org/abs/1204.4491`

		States of C	
A	B	0	1
0	0	1	0
0	1	0.5	0.5
1	0	0.6	0.4
1	1	0.3	0.7

Fig. 2. CPT of C with two parents A, B

Computation Complexity. The complexity of $\sigma(\cdot)$ calculation is dominated by the execution of the BP algorithm. A variety of BP algorithms exist. In our evaluation, we adopt the Loopy Belief Propagation (LBP) algorithm which was shown to perform well for various problems [17,18]. LBP's complexity to estimate the active probability of a node v is $O(M^d)$, where M is the number of possible labels (states) for each variable ($M = 2$), and d is the number of neighbors of v.

3.4 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,

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

The algorithm is summarized in Algorithm 1. It starts with the seed nodes and proceeds with the topological sorting order. Clearly, the algorithm has a complexity of $O(d \cdot n)$, where d is the maximum in-degree.

Algorithm 1. 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         $p(v) = 1$
      else
5         $p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u, v))$
6     $\sigma(S) = \sigma(S) + p(v)$
  output: $\sigma(S)$
```

4 DAG Construction

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

4.1 Localizing Influence Spread Region

One important observation in [8] 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 $P(u,v) = \{u_1, u_2, \ldots, u_l\}$ of length l from a vertex u to v, with $u_1 = u, u_l = v$ and $u_2, \ldots, u_{l-1}$ are intermediate vertices, define the propagation probability of the path, $p(P)$, as:

$$p(P(u,v)) = \prod_{i=1}^{l-1} p(u_1, u_{i+1}). \tag{3}$$

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:

$$MIP(\mathcal{G}, u, v) = \arg\max_P \{p(P) | P \in \mathcal{P}(\mathcal{G}, u, v)\}. \tag{4}$$

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 $MIP(\mathcal{G}, x, y)$.

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 defined as:

$$MIOA(\mathcal{G}, u, \theta) = \bigcup_{v \in V, p(MIP(\mathcal{G}, u, v)) \geq \theta} MIP(\mathcal{G}, u, v). \tag{5}$$

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.

4.2 Building DAGs

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,

$$r(v) = \min(-\log(p(P(s,v)))), \forall s \in S. \tag{6}$$

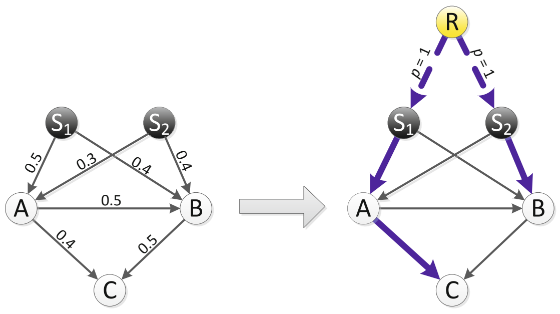
Rank grows as the node is further away from R. We include in $\mathcal{D}_1(\cdot)$ all edges in $\mathcal{G}$ whose end points are in $\mathcal{D}_1(\cdot)$ and go from a node with lower rank to one with higher rank. Clearly, the resulting graph is a DAG. The DAG constructing procedure is illustrated in Figure 3 and summarized in Algorithm 2.

Algorithm 2. Calculate $\mathcal{D}_1(S)$ from a seed set S

```
input : $\mathcal{G}, S, \theta$
1 Build $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$
2 $\mathcal{D}_1(S) = MIOA(\mathcal{G}_R, R, \theta) \backslash R$
3 Calculate $r(v), \forall v \in V_{\mathcal{D}_1}$ (Eq. (6))
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: $\mathcal{D}_1(S)$
```

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 2. A node v is associated with a rank $r(v)$ as in (6). 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 3.

The next proposition provides the relationship between DAGs constructed by Algorithm 2 and 3.



Node	S_1	S_2	A	B	C
r(Node)	0	0	0.301	0.398	0.699

Fig. 3. DAG due to Algorithm 2. 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 3. Calculate $\mathcal{D}_2(S)$ from a seed set S

```
input  : $\mathcal{G}, S, MIOA(\mathcal{G}, v, \theta), \forall v \in V$
1 $\mathcal{D}_2(S) = \bigcup_{\forall s \in S} MIOA(\mathcal{G}, s, \theta)$
2 Calculate $r(v), \forall v \in V_{\mathcal{D}_2}$ (Eq. (6))
3 foreach $(u, v) \in \mathcal{D}_2(S)$ do
4     if $r(u) \geq r(v)$ then
5         $\mathcal{D}_2(S) = \mathcal{D}_2(S) \backslash (u, v)$
output: $\mathcal{D}_2(S)$
```

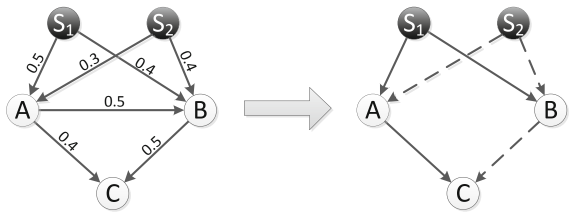
Proposition 1. *Given a fixed influence threshold θ, let $\mathcal{D}_1(\cdot) = (V_{\mathcal{D}_1}, E_{\mathcal{D}_1})$ and $\mathcal{D}_2(\cdot) = (V_{\mathcal{D}_2}, E_{\mathcal{D}_2})$ be the DAGs constructed by Algorithm 2 and Algorithm 3. Then, $V_{\mathcal{D}_1} = V_{\mathcal{D}_2}$ and $E_{\mathcal{D}_2} \subseteq E_{\mathcal{D}_1}$.*

Computation Complexity. The computation complexity of a Dijkstra tree is $O(n^2)$. When a new seed node is added, the worst cast computation time is $O(n^2)$ (if the corresponding $MIOA$ needs to be computed anew). The union operation in DAG 2 takes $O(n-1)$ time, and the edge pruning in DAG 1 and DAG 2 take $O(m)$ and $O(\min(m, k(n-1))$, respectively.

5 Accelerated Greedy Algorithm

In the original Greedy algorithm [4], in each round, a seed node with the maximum increment on influence spread is selected, namely, $v = \max_{v \in V \backslash S}(\sigma(S \cup \{v\}) - \sigma(S))$. We call $\delta_S(v) = \sigma(S \cup \{v\}) - \sigma(S)$ the spread increment of v under S. Initially, when $S = \emptyset$, $\delta_S(v) = \sigma(v)$.

To accelerate the execution of Greedy algorithm, one can try to improve on two aspects, namely, 1) limiting the set of nodes to pick from for the next



Node	S_1	S_2	A	B	C
r(Node)	0	0	0.301	0.398	0.699

Fig. 4. DAG due to Algorithm 3. 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). $\theta = 0.0001$.

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 CELF.

Recall in Section 4.1, 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:

$$PS(\mathcal{G}, v, \theta) = \{s \in V | MIOA(\mathcal{G}, s, \theta) \cap MIOA(\mathcal{G}, v, \theta) \neq \emptyset\}\,. \tag{7}$$

$PS(\mathcal{G}, v, \theta)$ can be computed efficiently just once at the beginning when all $MIOA(\mathcal{G}, v, \theta)$'s are available. To this end, we summarize the complete procedure to determine the optimal seed set in Algorithm 4.

6 Evaluation

In this section, we evaluate the performance of the proposed algorithms. Large scale social networks are used to evaluate the maximum influence spread of different algorithms. 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) [8]:** a very fast heuristic that builds a tree-like structure model on which influence is spread. We set the influence threshold $\theta = 1/160$.

Algorithm 4. Accelerated Greedy Algorithm

```
input : network graph $\mathcal{G}(V, E)$ and seed set size $k$
// initialization
1  $S = \emptyset, \sigma_0 = 0, \theta =$ 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 1)
6      $\delta(v) = \sigma(v)$
7      $\delta_{old}(v) = 0$
8  build $PS(\mathcal{G}, v, \theta), \forall v \in V$
// main loop
9  for $i = 1, \ldots, k$ do
       // select the i'th seed
10     $u = \arg\max_{v \in V \setminus S}(\delta(v))$
11     $S = S \cup \{u\}$
12     $\sigma_0 = \sigma(S)$
13     $\delta_{old}(v) = \delta(v), \forall v \in V \setminus S$
       // update incremental influence spread
14     $\delta_{max} = 0$
15     foreach $v \in PS(\mathcal{G}, u, \theta) \setminus S$ do
16         if $\delta_{old}(v) > \delta_{max}$ then
17             build $\mathcal{D}(S \cup \{v\})$ (Algorithm 2 or 3)
18             calculate $\sigma(S \cup \{v\})$ (LBP or Algorithm 1)
19             $\delta(v) = \sigma(S \cup \{v\}) - \sigma_0$
20             if $\delta(v) > \delta_{max}$ then
21                 $\delta_{max} = \delta(v)$
output: selected seed set $S$
```

– **Greedy:** The Greedy approach from [4] 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.

We do not compare with other heuristics such as SP1M, SPM [19], PageRank [20], Random, DegreeDiscountIC [12] or Betweenness centrality [21] since they have been reported in previous studies [8,4,6] to be either unscalable or have poorer performance.

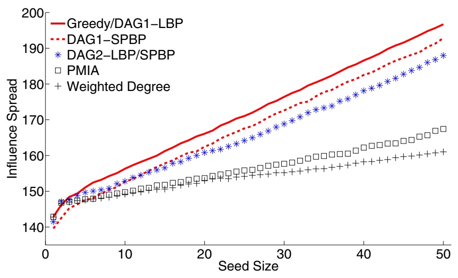
We have implemented the proposed algorithms in C++. All experiments are conducted on a workstation running Ubuntu 11.04 with an Intel Core i5 CPU and 2GB memory. In order to implement LBP algorithm, we use libDAI [22] and Boost [23] libraries. We find out through the implementation that constructing the CPT can be very costly when the in-degree of a node is high, and thus only include the parents with highest 10 influence probabilities in the factor graph. 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 [24] and [25] to compare the experimented algorithms. Details are summarized in Table 1.

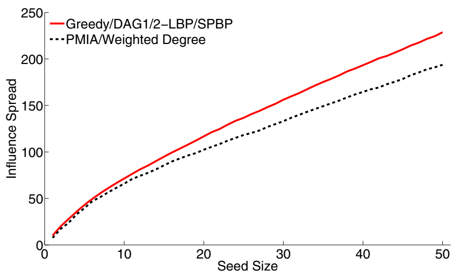
Probability Generation Model. Two models that have been used in previous work [4,8,12,6] are: 1) the WC model where $p(u, v) = 1/d(v)$ where $d(v)$ is the in-degree of v and 2) the TRIVALENCY model where $p(u, v)$ is assigned

Table 1. Network datasets

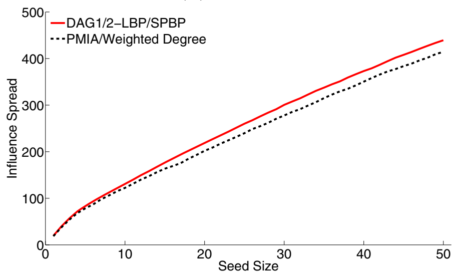
Name	*Email*	*p2p-Gnutella*	*soc-Slashdot*	*Amazon*
Nodes	447	6,301	82,168	262,111
Edges	5,731	20,777	948,464	1,234,877
Density	0.04	1e–03	1.6e–03	2.6e–05
Max Degree	195	97	5064	425
Mean Degree	25.64	6.59	23.09	9.42
Description	Email exchanged in a research lab during a year	Gnutella peer to peer network from August 2002	Slashdot social network from February 2009	Amazon product purchasing network from March 2003



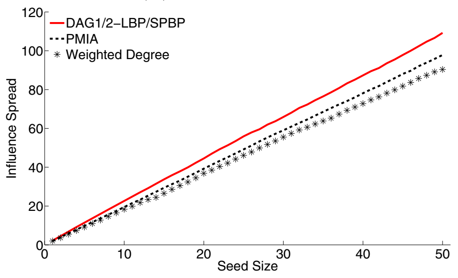
(a) *Email*



(b) *p2p-Gnutella*



(b) *soc-Slashdot*

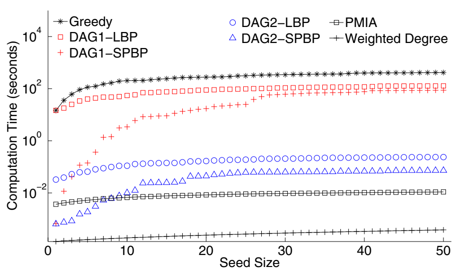


(c) *Amazon*

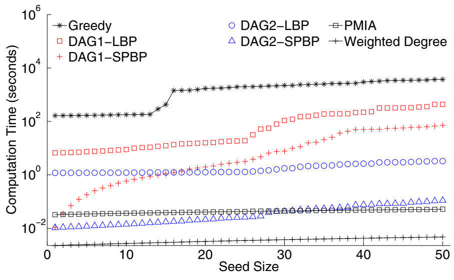
Fig. 5. Influence spread of the best seed sets on 4 datasets

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 TRIVALENCY model assigns a similar probability to all edges. In our evaluation, we consider the RANDOM model where $p(u, v)$ is randomly selected in the range [0.001, 0.1].

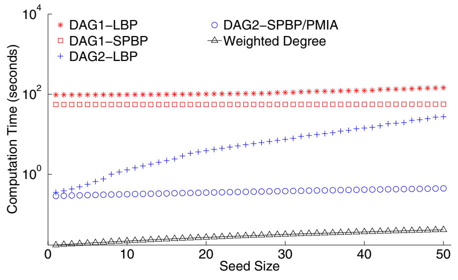
Influence Spread and Running Time. Figure 5 shows the influence spread generated by the best seed sets in different algorithms as the seed size changes. Since Greedy does not scale with large datasets, we only run Greedy on *Email* and *p2p-Gnutella*. In this set of experiments, the influence spread from the seed



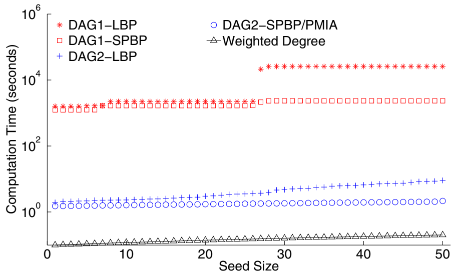
(a) *Email*



(b) *p2p-Gnutella*

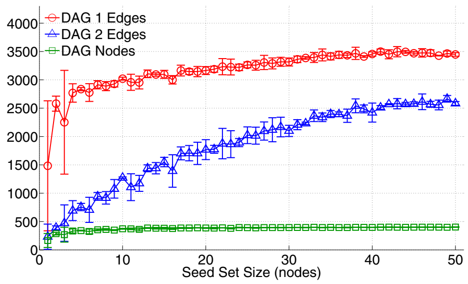


(b) *soc-Slashdot*

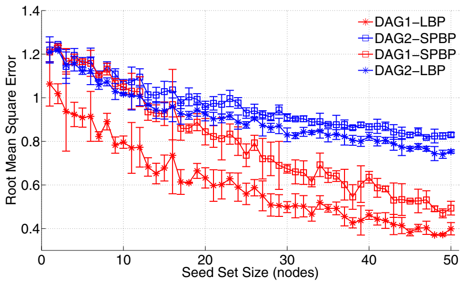


(c) *Amazon*

Fig. 6. Computation time on 4 datasets



(a) Number of nodes and edges in DAG



(b) RMSE

Fig. 7. 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.

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–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 [8]. 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.

Comparison of the Two DAG Models. To understand the behavior of the proposed algorithms, we conduct further experiments on *Email* dataset as it gives the most performance difference between the experimented algorithms.

Figure 7 (a) gives the number of vertices and edges as the result of the two DAG models with varying size of seed sets. Since both have the same number of vertices, only one curve is shown. It it clear that DAG 1 is much bigger from DAG 2 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,

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

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 7(b) shows that DAG 1 methods have smaller RMSE since they are based on a denser graph. More edges clearly help increase quality of the seed selection process. In the context of LBP vs. SPBP, LBP is slightly better since SPBP get rid of the state correlation between nodes. DAG 1 and LBP can help produce better inference result, but entails more computation complexity. The results are consistent with those in Figure 5(a).

Summary. From the conducted experiments, Weighted Degree gives the best efficiency in terms of spread/complexity. However, there are cases (*Email* dataset) in which Weight Degree performs poorly. Our proposed schemes works well 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.

7 Conclusion

In this paper, we considered the IM problem on social networks where the objective is to find a set k of nodes that can maximize the influence spread. We established the linkage between influence spread computation and BP on a Bayesian network. With 2 DAG models and 2 BP algorithms, 4 methods are proposed offering the flexibility between computation time and accuracy. Simulations using real-world social network graphs show that the proposed schemes achieve higher influence spread compared to the best known solutions. Interestingly, DAG 2 model, although being much smaller than DAG 1, gives a good approximation result that is comparable to DAG 1 with only a marginal computation cost. Result also exhibits the dependency of algorithm performance over the experimented network. Thus suggesting an interesting research direction to study the impact of graph structure in IM problem.

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

References

1. Dignan, L.: HP's touchpad fire sale: The fallout (2011), `http://www.zdnet.com/blog/btl/hps-touchpad-fire-sale-the-fallout/55594`
2. Domingos, P., Richardson, M.: Mining the network value of customers. In: Procs. of KDD 2001, pp. 57–66. ACM, New York (2001)
3. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of KDD 2002, pp. 61–70. ACM, New York (2002)
4. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD, KDD 2003, pp. 137–146. ACM, New York (2003)
5. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD, KDD 2007, pp. 420–429. ACM, New York (2007)
6. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proc. of WWW 2011, pp. 47–48. ACM, New York (2011)
7. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 199–208. ACM, New York (2009)
8. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of KDD 2010, pp. 1029–1038. ACM, New York (2010)

9. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: Proceedings of ICDM 2010, pp. 88–97. IEEE Computer Society, Washington, DC (2010)
10. Even-Dar, E., Shapira, A.: A note on maximizing the spread of influence in social networks. Inf. Process. Lett. 111, 184–187 (2011)
11. Sylvester, J.: Maximizing diffusion on dynamic social networks. Science (2009)
12. Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., Yuan, Y.: Influence maximization in social networks when negative opinions emerge and propagate. In: SDM, pp. 379–390. SIAM/Omnipress (2011)
13. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations, San Francisco, CA, USA, pp. 239–269 (2003)
14. Pearl, J.: Reverend Bayes on inference engines: A distributed hierarchical approach. In: Proceedings of the AAAI 1982, Pittsburgh, PA, pp. 133–136 (1982)
15. Lauritzen, S.L., Spiegelhalter, D.J.: Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. Journal of the Royal Statistical Society. Series B (Methodological) 50(2) (1988)
16. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy Belief Propagation for Approximate Inference: An Empirical Study. In: Procs. of Uncertainty in AI, pp. 467–475 (1999)
17. Frey, B.J., Koetter, R., Petrovic, N.: Very loopy belief propagation for unwrapping phase images, vol. 14, pp. 737–743. MIT Press (2001)
18. McEliece, R., MacKay, D., Cheng, J.F.: Turbo decoding as an instance of pearl's ldquo;belief propagation rdquo; algorithm. IEEE Journal on Selected Areas in Communications 16(2), 140–152 (1998)
19. Kimura, M., Saito, K.: Tractable Models for Information Diffusion in Social Networks. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 259–271. Springer, Heidelberg (2006)
20. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. 30, 107–117 (1998)
21. Freeman, L.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1979)
22. Mooij, J.M.: libDAI: A free and open source C++ library for discrete approximate inference in graphical models. Journal of Machine Learning Research 11, 2169–2173 (2010)
23. Boost.org: Boost c++ libraries, `http://www.boost.org/`
24. Leskovec, J.: Stanford large network dataset collection, `http://snap.stanford.edu/data/`
25. Wiki, I.: Social network generation, `http://www.infovis-wiki.net/index.php/Social_Network_Generation`