On Approximation of Real-World Influence Spread

Yu Yang, Enhong Chen, Qi Liu, Biao Xiang, Tong Xu, and Shafqat Ali Shad

Dept. of Computer Science, Univ. of Science and Technology of China,
Hefei 230026, China
{ryanyang,feiniaol,bxiang,tongxu,shafqat}@mail.ustc.edu.cn,
cheneh@ustc.edu.cn

Abstract. To find the most influential nodes for viral marketing, several models have been proposed to describe the influence propagation process. Among them, the *Independent Cascade (IC) Model* is most widely-studied. However, under IC model, computing influence spread (i.e., the expected number of nodes that will be influenced) for each given seed set has been proved to be #P-hard. To that end, in this paper, we propose *GS* algorithm for quick approximation of influence spread by solving a linear system, based on the fact that propagation probabilities in real-world social networks are usually quite small. Furthermore, for better approximation, we study the structural defect problem existing in networks, and correspondingly, propose enhanced algorithms, *GSbyStep* and *SSSbyStep*, by incorporating the *Maximum Influence Path* heuristic. Our algorithms are evaluated by extensive experiments on four social networks. Experimental results show that our algorithms can get better approximations to the IC model than the state-of-the-arts.

1 Introduction

Recently, viral marketing has drawn more and more attention from both industrial and research fields [15]. This kind of marketing is based on the word-of-mouth effect in social networks, i.e., one may be influenced by his neighbors. According to a survey [11], 83% of people prefer consulting family, friends or an expert over traditional advertising before trying a new restaurant, 71% of people do the same before buying a prescription drug, and so on. Thus, companies believe that viral marketing should be one of the most effective marketing strategy.

Unlike traditional marketing strategies, viral marketing only targets a few influential individuals in a social network. For example, if a company employs a viral marketing strategy to promote sales performance, it only needs to choose a small number of individuals and persuade them to be the initial users(by offering them discounted or free products etc.). Due to the word-of-mouth effect, these initial users, also called seed users, will influence their friends to use this product. And once the friends are influenced, they will try to influence their friends and so on. Eventually, a much larger group of people compared to the number of initial users will adopt this product.

Since different seed users usually lead to different results of word-of-mouth propagation, an important step for a successful viral marketing is to precisely evaluate the expected number of influenced individuals (influence spread) for each seed user set. Along this line, several influence models have been proposed [8][9][12]. Among these models, *Independent Cascade(IC) Model* is a simple and most widely used one that describes the information propagation in a social network as a stochastic process according to certain probabilistic rules [9][12]. However, Wei Chen *et al.* have proved that computing influence spread of a set of seed users, under IC model, is a #P-hard problem [7]. As an alternative, Monte Carlo simulation, which is very time-consuming, is employed to approximately calculate influence spread. For example, for a moderate sized social network, we usually have to run Monte Carlo simulation for more than ten thousands times to obtain a good estimation of the true influence spread. Thus, the problem of efficiently computing influence spread has become a bottleneck for the deployment of viral marketing.

To that end, in this paper, we provide a novel study on approximating influence spread under IC model, mainly based on the observation that the influence propagation probabilities in real-world social networks are usually quite small. Specifically, our main contributions can be summarized as follows:

– To address the above efficiency problem, we show that by solving a linear system we can approximately calculate each node (individual, or user)'s probability of being influenced, and thus we can quickly compute influence spread of a given seed set.
– We point out that *SteadyStateSpread* algorithm [1] is also an approximation for the real influence spread under IC model when the propagation probabilities are small, and this was not illustrated by Aggarwal, *et al.* in [1].
– For better approximation, we discover the structural defect problem (as illustrated by Definition 2 in Section 4.1) existing in networks, and further propose enhanced algorithms by incorporating the Maximum Influence Path(MIP) heuristic [7] to both our algorithm and *SteadyStateSpread.*
– We evaluate our algorithms on four real networks. Experimental results prove our discoveries and show that the proposed algorithms can get good approximations of the real influence spread. Moreover, the nodes ranking obtained by our algorithms are very similar to the true ranking results.

2 Related Work

In general, related work can be grouped into two categories. The first category includes the most relevant work on influence models. In the second category, we introduce the related work on computing social influence spread.

Influence Models. Domingos *et al.* first proposed to mine social networks for marketing [8]. However, their models are essentially descriptive, and prior works [1][7][12][13] following [8] focus more on the models that explicitly represent the step-by-step dynamics of influence.

Among existing influence models, *Independent Cascade(IC) Model* [9][12] is one of the most widely-studied models. In IC model, a social network is represented by a directed graph $G(V, E)$. Each node $v \in V$ denotes an individual in the social network. Each edge (u, v) represents the relationship between u and v, and is assigned with a real number p_{uv} ($p_{uv} \in [0, 1]$) which is the probability that v is influenced by u through the edge in the next step after u is activated. In the beginning, there is a set of users (seed set) who are already activated, i.e. influenced, denoted by S. Let S_t represents the set of nodes that become activated at time t, and $S_0 = S$. At time $t + 1$, each node $u \in S_t$ will try to activate its inactivated neighbor v with probability p_{uv}. If there are no newly activated nodes at time $t+1$, the propagation process ends. The *influence spread* of S, which is denoted by $\sigma(S)$, is the expected number of nodes being activated in the whole propagation process.

Two kinds of IC model, namely *Uniform IC Model* and *Weighted Cascade(WC) Model* are defined in [12]. In uniform IC model, each link shares the same propagation probabilities, while according to WC model, the propagation probability through edge(u,v) equals to $weight(u, v)/indegree(v)$.

Influence Spread Computation. In [12], Kempe, *et al.* exploited running Monte Carlo simulation for a large number of times to evaluate the influence spread of a given seed set under IC model.

Further, Wei Chen *et al.* [7] proved that computing influence spread based on IC model is #P-hard. Given the fact that the influence of a node is always local, they proposed the Maximum Influence Path(MIP) heuristic. Specifically, they defined the *propagation probability* of a path $P =< u = n_1, n_2, ..., n_m = v >$ $(u \in S)$, $pp(P)$, as

$$pp(P) = \prod_{i=1}^{m-1} p_{n_i n_{i+1}}$$

and the maximum influence path from S to v, which is denoted by $MIP(S, v)$, is defined as

$$MIP(S, v) = arg \max_{P} \{pp(P)|P \in Path(S, v)\}$$

in which $Path(S, v)$ is the set of all paths from S to v. By using the MIP heuristic, they regard the probability of v being influenced by seed set S as $pp(MIP(S, v))$. Their experiments showed that by using this heuristic one can accelerate the influence maximization algorithm drastically. However, the influence spread calculated by the MIP heuristic maybe very different with the true influence spread because it abandons too many possible paths.

In contrast, Kimura and Saito proposed the shortest-path based influence models and provided efficient algorithms to compute influence spread under these models in [13]. However, their algorithms are only suitable for the uniform IC model where propagation probabilities of links are all the same, which is seldom the case in reality.

Recently, Aggarwal *et al.* [1] proposed the *SteadyStateSpread* method to compute *flow authority* by solving a system of nonlinear equations,

$$p_v = \begin{cases} 1 & \text{if } v \in S \\ 1 - \prod_{u \in N(v)} (1 - p_{uv} * p_u) & \text{if } v \notin S \end{cases} \tag{1}$$

in which $N(v)$ is the set of all v's in-neighbors and p_v indicates the probability of v being influenced. As we will explain later, Eq.(1) can be used to approximately compute the real influence spread under IC model when propagation probabilities through links are small. However, it does not strictly hold for some situations(e.g., the situation illustrated in Section 4.1). More importantly, there are some difficulties in solving systems of nonlinear equations, such as the convergence problem [6] and the multiple solutions problem [10]. Though Aggarwal *et al.* [1] gave a simple iterative algorithm for solving Eq.(1), they did not theoretically prove that their algorithms can converge or Eq.(1) has only one solution.

3 Approximating Influence Spread by Linear System

In this section, we first illustrate the observation that influence propagation probabilities in real-world social networks are usually quite small. Based on this fact, we then show the way to approximate influence spread, and represent the approximation by a linear system. At last, we propose a simple iterative algorithm to solve the linear system and return the influence spread for each seed set.

3.1 Preliminary Observation

Though there exists the effect of peer-to-peer influence, in real scenarios of information diffusion, propagation probabilities between people are very small. For example, according to [5], in Facebook, the average probability that an individual will share a Web link is about 2% when there are 6 of his friends who shared the same link before. In LiveJournal, the average probability that an individual will join a community is no more than 2% even though 50 of his/her friends have already joined that community [4]. Moreover, influence is not the only reason that leads to phenomena of friends sharing same Web links and joining same communities. Prior works, [2][3][16], showed that correlations between friends in social networks are also an important factor that contributes to homophily phenomena. Thus, the actual propagation probability caused by the effect of influence is even smaller.

To facilitate the following discussion, we first define *small propagation probabilities* mathematically.

Definition 1 (Small Propagation Probabilities). *Given a network $G = (V, E)$, if for $\forall v \in V$, and for $\forall u \in N(v)$, $p_{uv} < weight(u, v)/indegree(v)$, we say the propagation probabilities in G are small. Note that $indegree(v) = \sum_{u \in N(v)} weight(u, v)$.*

From Definition 1 we can see that if the small propagation probabilities condition holds in a social network, then for $\forall v \in V$, $\sum_{u \in N(v)} p_{uv} < 1$ because $p_{uv} < weight(u, v)/indegree(v)$. The further explanations of these two formulas can be easily observed from the real world social networks: First, besides the influence coming from the direct neighbors in the specific social network, each node v's activity is also impacted by the information from other sources (e.g., influence coming from other friends that are not included in the current network). Second, when influence propagates in the network, there is information lost (or decay) in each node, and this has been widely observed and adopted by models from other domains, such as the PageRank method [14].

3.2 Approximation of Real-World Influence Probability

In this subsection we consider approximating influence spread under IC model for social networks, where propagation probabilities are small. We use $\sigma(S)$ to denote influence spread of a seed set S. It is obvious that $\sigma(S) = \sum_v p_v$, in which p_v denotes the probability that v will be influenced, also called the *influence probability* of v. Since $\forall v \in S$, $p_v = 1$, the core step of computing $\sigma(S)$ is to compute p_v for each $v \notin S$. In the following, we show that the influence probability of $v(v \notin S)$ can be well approximated by a linear equation that $p_v = \sum_{u \in N(v)} p_u * p_{uv}$.

Let $v(t)$ denotes the event that v becomes influenced at time t. Note that $p\{v(0)\} = 1$ if $v \in S$. For each node $v \notin S$, $p\{v(t)\}$ can be computed by

$$p\{v(t)\} = \sum_{W \subseteq N(v)} p\{v(t)|W(t-1), \tilde{v}(t-1)\} p\{W(t-1), \tilde{v}(t-1)\} \tag{2}$$

where $N(v)$ is the set of in-neighbors of v, W is a subset of $N(v)$, $W(t-1)$ indicates the event that all nodes in set W become influenced at time $t-1$ and $\tilde{v}(t-1)$ denotes the event that v is still not influenced after time $t-1$. Worth noting that $p\{v(t)\} = p\{v(t), \tilde{v}(t-1)\}$[1]. It is obvious that

$$p\{v(t)|W(t-1), \tilde{v}(t-1)\} = 1 - \prod_{u \in W} (1 - p_{uv}) \tag{3}$$

Following [8] and [13], given the marginals $\{p\{u(t-1)\};u \in W\}$ and $p\{\tilde{v}(t-1)\}$, we approximate $p\{W(t-1), \tilde{v}(t-1)\}$ by the maximal entropy estimation:

$$p\{W(t-1), \tilde{v}(t-1)\} = p\{\tilde{v}(t-1)\} \prod_{u \in N(v)} p\{u(t-1)\}^{h_u} (1 - p\{u(t-1)\})^{1-h_u} \tag{4}$$

where h_u is an indicator that if $u \in W$, h_u=1, otherwise h_u=0. Combining Eq.(2), Eq.(3) and Eq.(4), and after some algebraic transformations, we have

$$p\{v(t)\} = p\{\tilde{v}(t-1)\}[1 - \prod_{u \in N(v)} (1 - p_{uv} * p\{u(t-1)\})] \tag{5}$$

Note that under the small propagation probabilities condition, p_{uv} is very small, the probability that v will be influenced is even smaller. So $p\{\tilde{v}(t-1)\}$ is very

[1] If v is influenced at time t, it should stay inactive from time 0 to $t-1$.

close to 1 [2]. Thus, $p\{v(t)\}$ can be approximated by

$$p\{v(t)\} = 1 - \prod_{u \in N(v)} (1 - p_{uv} * p\{u(t-1)\}) \tag{6}$$

Since $p_{uv} * p\{u(t-1)\} << 1$ and $(1-a)(1-b) \approx 1 - a - b$ when $a, b << 1$, we can further approximate $p\{v(t)\}$ with a linear equation:

$$p\{v(t)\} = \sum_{u \in N(v)} p_{uv} * p\{u(t-1)\} \tag{7}$$

Now we add up $p\{v(t)\}$ over t to get p_v. Note that $p_v = 1$ if $v \in S$. For $v \notin S$, we have

$$\begin{aligned} p_v &= \sum_{t=0} p\{v(t)\} \\ &= p\{v(0)\} + \sum_{u \in N(v)} p_{uv} \sum_{t=1} p\{u(t-1)\} \\ &= \sum_{u \in N(v)} p_{uv} p_u \end{aligned} \tag{8}$$

Because we are discussing approximation under the condition that $\sum_{u \in N(v)} p_{uv} < 1$, it is guaranteed that $0 \leq p_v < 1$. Thus, the influence probability p_v computed by Eq.(8) is well defined.

Approximation From SteadyStateSpread. As have said, under the small propagation probabilities condition $p_{uv} * p_u << 1$, and recall Eq.(1), it is easy to conclude that $\sum_{u \in N(v)} p_{uv} p_u \approx 1 - \prod_{u \in N(v)} (1 - p_{uv} p_u)$. In this situation, we can see that *SteadyStateSpread* is actually also an approximation for the true influence spread. In the following experiment section, our experimental results verify that the smaller p_{uv} is, the more accurate approximation will be obtained by *SteadyStateSpread.*

3.3 Linear System Formulation

Rewriting Eq.(8) in a matrix form we can get a linear system. To facilitate the following discussion, first we list some math notations in Table 1.

Table 1. Math Notations

Notations	DESCRIPTION
$[\mathbf{M}]_{\lvert V\rvert * \lvert V\rvert}$	probability adjacent matrix, $\mathbf{M}_{uv} = p_{uv}$
$\mathbf{P} = [p_1, p_2, ..., p_{\lvert V\rvert}]^T$	p_v is the probability of v being influenced by seed set S
$\mathbf{B} = [b_1, b_2, ..., b_{\lvert V\rvert}]^T$	$b_v = 1$ if $v \in S$, otherwise $b_v = 0$
$\mathbf{M}_{\overline{S}}$	matrix which is cut down from matrix $\mathbf{M}$ by removing rows whose index $v \in S$
$\mathbf{M}_{\overline{SS}}$	matrix which is cut down from matrix $\mathbf{M}$ by removing rows and columns whose index $v \in S$

[2] The assumption on this value will be verified in the experimental section 5.3.

For node $v \notin S$, we rewrite Eq.(8) by

$$p_v = \sum_{u \in N(v) \bigwedge u \notin S} p_{uv} * p_u + \sum_{u \in N(v) \bigwedge u \in S} p_{uv} \tag{9}$$

Further, using matrix form to represent the right side of Eq.(9), we have

$$p_v = (\mathbf{M}_{\overline{S}})_v^T \mathbf{P}_{\overline{S}} + (\mathbf{M}^T \mathbf{B})_v \tag{10}$$

Put all nodes v that $v \notin S$ together,

$$\mathbf{P}_{\overline{S}} = (\mathbf{M}_{\overline{SS}})^T \mathbf{P}_{\overline{S}} + (\mathbf{M}^T \mathbf{B})_{\overline{S}} \tag{11}$$

thus we have

$$[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T] \mathbf{P}_{\overline{S}} = (\mathbf{M}^T \mathbf{B})_{\overline{S}} \tag{12}$$

Given that we are dealing with approximations of influence spread under the condition that $\sum_{u \in N(v)} p_{uv} < 1$, the matrix $[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]$ is strictly diagonally dominant. Thus, $[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]$ is invertible and the linear system of Eq.(12) has only one solution that $\mathbf{P}_{\overline{S}}$=$[\mathbf{I} - (\mathbf{M}_{\overline{SS}})^T]^{-1}(\mathbf{M}^T \mathbf{B})_{\overline{S}}$. In this way, by summing up p_v over each v, we get influence spread for seed set S(i.e., $\sigma(S)$).

3.4 A Simple Iterative Algorithm

As we stated above, to approximately compute the influence spread of a seed set S, we only have to solve the linear equation Eq.(12) and sum up all p_v.

Algorithm 1. GS(G, S)

```
1:  for $v = 1$ to $|V|$ do
2:      if $v \in S$ then
3:          $p_v \leftarrow 1$
4:      else
5:          $p_v \leftarrow 0$
6:      end if
7:  end for
8:  while not converge do
9:      for $v = 1$ to $|V|$ do
10:         if $v \notin S$ then
11:             $p_v \leftarrow \sum_{u \in N(v)} p_{uv} * p_u$
12:         end if
13:     end for
14: end while
15: $sum \leftarrow 0$
16: for $v = 1$ to $|V|$ do
17:     $sum \leftarrow sum + p_v$
18: end for
19: return $sum$
```

Since $[\mathbf{I}-(\mathbf{M}_{\overline{SS}})^T]$ is strictly diagonally dominant, we can use *Gauss-Seidel(GS)* algorithm to efficiently solve Eq.(12) in $O(E)$ time. The entire computation process can be illustrated by Algorithm 1, where the iteration formula used is Eq.(8). Based on the output of Algorithm 1, we can rank each seed set S, so as to find the most influential individuals for marketing.

4 Algorithms Incorporating MIP Heuristic

In this section, we first illustrate the structural defect problem existing in many methods for social networks. Then, to address this problem and to get a better approximation, we propose enhanced algorithms by incorporating the Maximum Influence Path(MIP) heuristic into both *GS* algorithm and *SteadyStateSpread*.

4.1 Structural Defect

According to Eq.(1) and Eq.(8), for $v \notin S$, the value of p_v depends on all of its in-neighbors. However, in fact sometimes a node u's influence probability p_u is independent from some of its in-neighbors. Consider the following example of an undirected network, as shown in Fig.1.

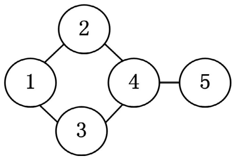

Fig. 1. An undirected network

In this undirected network, assume that node 1 is the only seed node. Based on Eq.(1) and Eq.(8), the probability of node 4 being influenced depends on the probability of node 5 being influenced. However, actually, to influence node 5, node 4 should be influenced first. Thus, p_4 is irrelevant with p_5, and this contradicts with Eq.(1) and Eq.(8). We have reasons to believe that each network, especially an undirected network, may have many sub-structures like Fig.1, and in this situation, if we use Eq.(1) or Eq.(8) to calculate p_v would get misleading results. In summary, we call structures like (node 4, node 5) in Fig.1 *structural defect* of Eq.(1) and Eq.(8), and we define it in a mathematical way.

Definition 2 (Structural Defect). *Given a network $G=(V,E)$, a seed set S and an influence spread algorithm A, if $\exists u, v \notin S$, $\forall P=(p_{n_1}, p_{n_2}, ..., p_{n_m}=v) \in Path(S,v)$, $u \in P$(that is, every path from S to v has to pass u), and according to A, the value of p_u depends on p_v, we say that (u,v) is a **structural defect** of algorithm A on G.*

4.2 Incorporating Maximum Influence Path Heuristic

To handle the problem of structural defect, we propose to incorporate the Maximum Influence Path(MIP) heuristic [7] into our algorithm. The main idea is to set an iteration threshold $\beta(v)$ for node v, i.e., for each v, we only update p_v in the first $\beta(v)$ iterations.

Let's consider the algorithms of *Gauss-Seidel(GS)* and *SteadyStateSpread* [1], which are both iterative processes of accumulating each path's influence probability for a node (e.g., v) by Eq.(8) and Eq.(1) respectively. Specifically, for a path P from S to v, if we want to include the propagation probability through P, we have to update p_v for no less than $length(P)$(hops of P) iterations. Since the maximum influence path is the path with biggest propagation probability, we do not want to abandon this path's influence. Thus, let $step[v]$ denotes hops of this maximum influence path from S to v, and to include the influence probability of maximum influence path, we should set the iteration threshold for v to be at least $step[v]$.

However, since $step[v]$ is usually also the shortest distance from S to v, and if we just update p_v in the first $step[v]$ rounds of iterations, we may miss some important influences from other paths. For example, the value of p_2 for node 2 in Fig.1 does depend on the influence from all its neighbors. In contrast, if we update p_v for too many iterations, the structural defect will have serious impact on the estimation of influence spread. Take node 4 in Fig.1 for example, as the

Algorithm 2. GSbyStep(G, S)

```
1:  for $v = 1$ to $|V|$ do
2:      if $v \in S$ then
3:          $p_v \leftarrow 1$
4:      else
5:          $p_v \leftarrow 0$
6:          calculate hops of the maximum influence path from $S$ to $v$, denoted by $step[v]$
7:      end if
8:  end for
9:  $times \leftarrow 0$, $updated \leftarrow true$
10: while $updated$ is $true$ do
11:     $updated \leftarrow false$, $times \leftarrow times + 1$
12:     for $v = 1$ to $|V|$ do
13:         if $v \notin S \bigwedge times \leqslant step[v] + 1$ then
14:             $p_v \leftarrow \sum_{u \in N(v)} p_{uv} * p_u$
15:             $updated \leftarrow true$
16:         end if
17:     end for
18: end while
19: $sum \leftarrow 0$
20: for $v = 1$ to $|V|$ do
21:     $sum \leftarrow sum + p_v$
22: end for
23: return $sum$
```

round of iterations increases, p_5 will become larger and the structural defect caused by node 5 for p_4 will be more and more serious.

To that end, as a tradeoff, we choose $step[v] + 1$ as v's iteration threshold $\beta(v)$. Choosing such value is because in the $(step[v] + 1)$-th iteration there are few miss counted influence probabilities, and meanwhile, the bad impact from structural defect is limited. Along this line, by incorporating MIP heuristic into Algorithm 1 and *SteadyStateSpread*, we propose *GSbyStep* algorithm(as illustrated by Algorithm 2) and *SSSbyStep* algorithm[3] for better approximations. Worth noting that our solution also works for some other iterative algorithms, where the *structural defect* problem exists.

5 Experimental Evaluation

5.1 Experimental Setup

We evaluate our algorithms by experiments on four real-world datasets (two directed networks and two undirected networks) that we downloaded from SNAP [4]. Detailed information of these four data sets can be seen in Table 2.

Table 2. Description of Data Sets

Name	Description	Type	Nodes	Edges
wiki-Vote	Wikipedia who-votes-on-whom network	Directed	7,115	103,689
p2p-Gnutella04	Gnutella peer to peer network from August 4 2002	Directed	10,876	39,994
email-Enron	Email communication network from Enron	Undirected	36,692	367,662
ca-AstroPh	Collaboration network of Arxiv Astro Physics	Undirected	18,772	396,160

Unlike Kimura and Saito using uniform IC model [13] where propagation probabilities are all the same, for simulating the real-world influence propagation process more accutately, we evaluate our algorithms following more general cases of IC model. Specifically, we set propagation probabilities in a WC model [12] like way. As stated in the related work section, the propagation probability of an edge (u, v) under WC model is $weight(u, v)/indegree(v)$. Thus, this model assumes that the total influence probabilities of a node's in-neighbors are 1. As we have discussed in Section 3.1, this assumption is too idealistic for real scenarios. In order to make the influence propagation process more realistic, and following Definition 1, we slightly change WC model by setting $\sum_{u \in N(v)} p_{uv} = \alpha$, thus $p_{uv} = \alpha * weight(u, v)/indegree(v)$. In this way, we can not only cover more general cases of IC model (by slightly changing weights of the edges) but also capture the characteristics (i.e., small and different values) of the real-world influence propagation probabilities. Thus, in the following, we call such modified model as *General IC Model*, or IC model for short.

[3] Modifying the 14-th line of *GSbyStep* algorithm by replacing $\sum_{u \in N(v)} p_{uv} * p_u$ with $1 - \prod_{u \in N(v)} (1 - p_{uv} * p_u)$, we get the *SSSbyStep* algorithm.

[4] `http://snap.stanford.edu`

For each network, we conduct three groups of experiment by setting α as 0.5 for all nodes, setting α as 0.75 for all nodes, and randomly generating α with a uniform distribution on [0.1,0.9] for each node, respectively. For evaluation, the average result of 20,000 times Monte Carlo simulation of IC model (*GICM*) is regarded as the real influence spread. In the following, we compare our algorithms (*SSSbyStep*,*GSbyStep*,*GS*) with four baseline algorithms, *MIP* [7], *SteadyStateSpread(SSS)* [1], *SP1M* [13] and an insufficient times Monte Carlo simulation of IC model, i.e., 200 times of Monte Carlo simulation(*GICM200*).

5.2 Ranking Problem

First we consider the problem of extracting influential seeds by ranking candidate nodes. Specifically, we try to find the rank list of $100(|S| = 100)$ influential seeds, i.e., the top-100 nodes with higher influence spread than others.

We use the result of 20,000 times Monte Carlo simulation as the ground truth, and the ranking according to such simulation is regarded as the true nodes rank. For evaluating rankings obtained by different algorithms, we compare similarities between those rankings with the true ranking. We employ the same measurement which is used in [13], the *ranking similarity* $F(k)$. $F(k)$ quantifies the degree of similarity between two ranking methods at rank k, and it is defined as follows: Let $L(k)$ and $L'(k)$ be the respective sets of top-k nodes for the two ranking methods that we compare. Then, $F(k) = |L(k) \bigcap L'(k)| \ / \ k$.

Fig.2-Fig.5 show the experimental results . We can see that rankings obtained by our proposed algorithms and *SteadyStateSpread(SSS)* are very similar

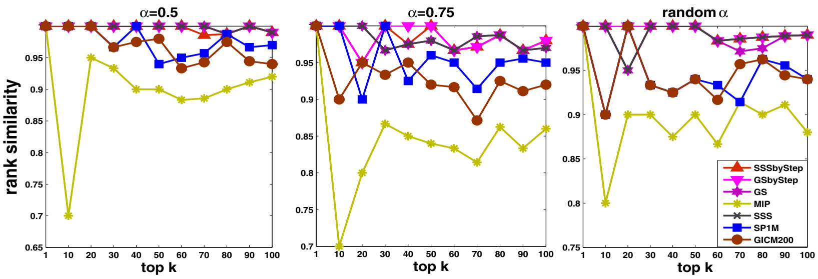

Fig. 2. Rank Similarity Comparison on wiki-Vote Network Data

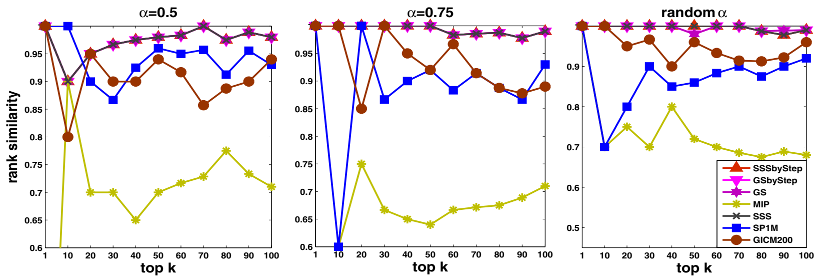

Fig. 3. Rank Similarity Comparison on p2p-Gnutella04 Network Data

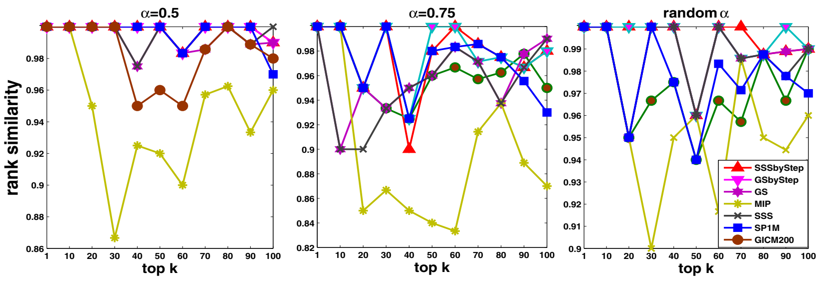

Fig. 4. Rank Similarity Comparison on email-Enron Network Data

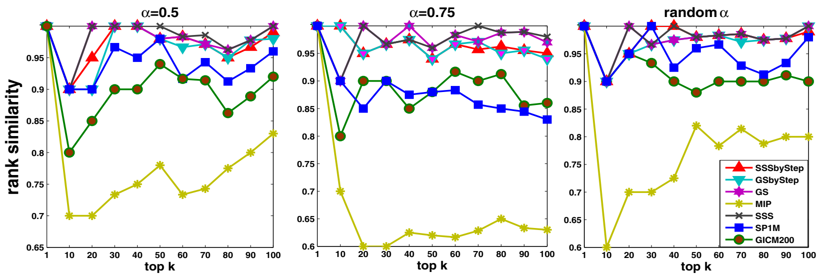

Fig. 5. Rank Similarity Comparison on ca-AstroPh Network Data

to the true ranking while others are not. This is because both our algorithms and *SteadyStateSpread(SSS)* are approximations for the real influence spread as stated in Section 3.

Another observation is that in most cases all algorithms can choose the most influential seed(top-1 node) accurately, even algorithms like *MIP* which is not very effective on ranking similarity metric. This may explain why these algorithms work well for the influence maximization problem [7][12][13]. Since under the greedy framework [12] for solving this problem, an algorithm only cares about finding the node with the maximum increasing influence spread in each round, and this node will be added into seed set.

5.3 Estimation of Influence Spread

We then evaluate each algorithm by their performances on estimating the influence spread for a given seed set. For each dataset, we first calculate influence spreads of the top-1000 nodes with highest out-degrees[5], i.e., experiment with setting the size of seed set equals to 1. Then we randomly generate 100 different seed sets with the size to be 10, 20, 30, 40, 50, respectively. At last, the effectiveness of each algorithm is measured by *error rate* of the influence spread returned by this algorithm. Here, we define $\sigma(S)$ as the true influence spread

[5] Since people often care about top influential nodes, and nodes with higher out-degree are usually more influential.

Table 3. Average Influence Spread ($\sigma(S)$) under $\alpha = 0.5$ Setting

dataset \ size of seed set	1	10	20	30	40	50
wiki-Vote	4.58	45.22	87.83	133.25	178.60	222.61
p2p-Gnutella04	3.76	37.34	74.15	112.60	148.97	185.13
email-Enron	14.21	151.34	269.06	428.79	559.47	733.68
ca-AstroPh	4.41	44.06	86.04	129.91	171.13	215.14

Table 4. Average Influence Spread ($\sigma(S)$) under $\alpha = 0.75$ Setting

dataset \ size of seed set	1	10	20	30	40	50
wiki-Vote	8.29	80.76	156.41	234.35	310.25	384.17
p2p-Gnutella04	8.91	89.58	169.30	260.26	343.52	419.03
email-Enron	31.04	318.28	579.39	895.69	1,160.61	1,495.89
ca-AstroPh	10.65	104.97	201.66	301.34	392.89	485.68

Table 5. Average Influence Spread ($\sigma(S)$) under random α Setting

dataset \ size of seed set	1	10	20	30	40	50
wiki-Vote	4.71	46.34	90.73	136.85	183.16	227.99
p2p-Gnutella04	3.81	37.74	75.29	113.77	151.07	188.41
email-Enron	14.02	150.40	264.93	424.46	552.89	724.63
ca-AstroPh	4.39	44.09	85.69	129.41	170.09	212.86

computed by 20,000 times Monte Carlo simulation, and $\sigma^{'}(S)$ as the influence spread calculated by an approximate algorithm. Then the error rate of $\sigma^{'}(S)$ can be measured by $|\sigma(S) - \sigma^{'}(S)|/\sigma(S)$.

Ground Truth. The average values of $\sigma(S)$ under different α settings are illustrated in Table 3 - Table 5, where a much larger group of nodes are finally influenced compared to the size of each seed set. Based on these tables, we then present a verification for the assumption that $p\{\tilde{v}(t-1)\}$ is very close to 1 for each v and t. For simplicity, we take $\alpha = 0.75$ and the email-Enron network (with the largest number of influenced nodes) as an example. From Table 4 we can see that when $|S| = 50$, the average p_v for each node in email-Enron network is $1,495.89/36,692 = 0.04$, thus $1-p_v = 0.96$. Since $1-p_v < p\{\tilde{v}(t-1)\}$, it means $p\{\tilde{v}(t-1)\} > 0.96$. Similarly, when $|S| = 1$, we can get $p\{\tilde{v}(t-1)\} > 0.999$. From these lower bounds we can summarize that $p\{\tilde{v}(t-1)\}$ is very close to 1.

Effectiveness. Correspondingly, Fig.6-Fig.9 show the average error rates of different algorithms when the size of each seed set varies from 1 to 50. From Fig.6 and Fig.7 we can see that our two linear algorithms *GSbyStep* and *GS* are the most accurate algorithms for the two directed networks. Error rates of these two algorithms are lower than 1% in most cases. While from Fig.8 and Fig.9, we find that for the two undirected networks, *SSSbyStep* has the lowest error rate in most cases(with error rate lower than 3%). Worth noting that, for the two undirected networks, when $\alpha = 0.75$, accuracy has been significantly

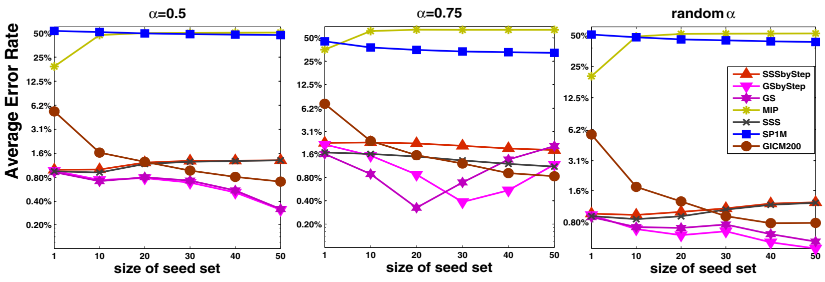

Fig. 6. Average Error Rate Comparison on wiki-Vote Network Data

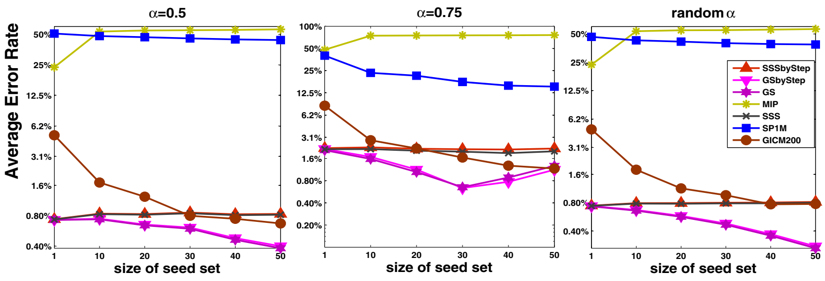

Fig. 7. Average Error Rate Comparison on p2p-Gnutella04 Network Data

improved by incorporating MIP heuristic into algorithms, especially when the size of the seed set is 1. Specifically, average error rate of *SSSbyStep* algorithm is about 7% lower than *SteadyStateSpread(SSS)* algorithm, and average error rate of *GSbyStep* algorithm is about 10% lower than *GS* algorithm.

Another observation is that error rates of our algorithms and *SteadyState-Spread(SSS)* are lower when setting $\alpha = 0.5$ than setting $\alpha = 0.75$. This substantiates that both our algorithms and *SteadyStateSpread* can better approximate the real influence spread when propagation probabilities are smaller.

Efficiency. Our algorithms are also very efficient compared to baseline algorithms. Fig.10 shows the average processing time when $\alpha = 0.75$ and the size of each seed set varies from 1 to 50, where *GICM* denotes the algorithm of 20,000 times Monte Carlo simulation of IC model. Similar results under other α settings are omitted due to the space limit. We can see that tens thousand times Monte Carlo simulation (*GICM*) is very time-consuming. And insufficient times Monte Carlo simulation (*GICM200*) is fast when the size of seed set is small, but the error rate is correspondingly very high. Moreover, another drawback of Monte Carlo simulation is that the processing time increases as seed set gets larger. In contrast, all of our algorithms are iteration-based algorithms, so the running time does not increase when seed sets become larger.

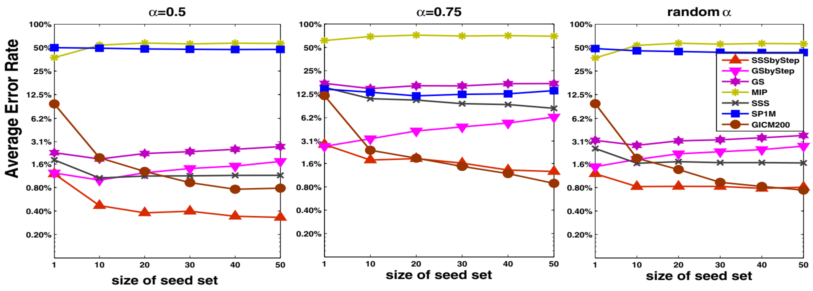

Fig. 8. Average Error Rate Comparison on email-Enron Network Data

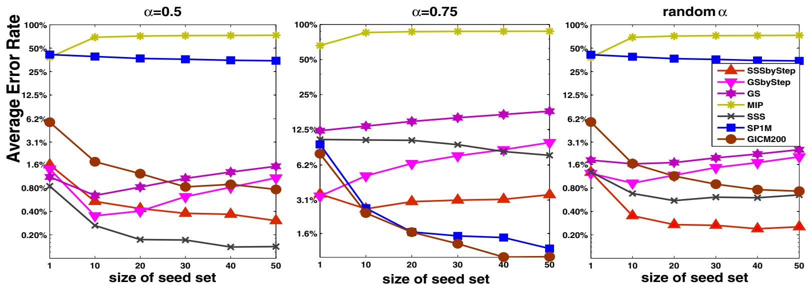

Fig. 9. Average Error Rate Comparison on ca-AstroPh Network Data

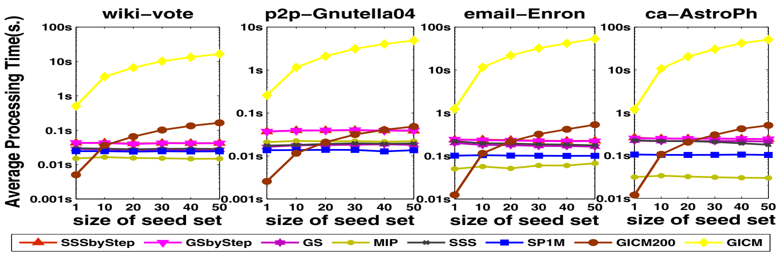

Fig. 10. Average Processing Time($\alpha = 0.75$)

5.4 Effectiveness of Iteration Threshold

In previous experiments, we set $step[v] + 1$ as the iteration threshold for each node v. In this subsection, we provide an experimental proof of this value. We set $\alpha = 0.75$ and use *GSbyStep* algorithm as an example for evaluation, and similar results can be observed for other parameter settings and algorithm *SSSbyStep*. Let $step[v] + r$ be the iteration threshold that we set in *GSbyStep* algorithm for node v. We did 11 groups of experiments of computing influence spreads of the 1000 nodes we picked in Section 5.3 by setting $r = 0, 1, 2, 3..., 10$, respectively. The corresponding average error rates for each r are shown in Fig.11.

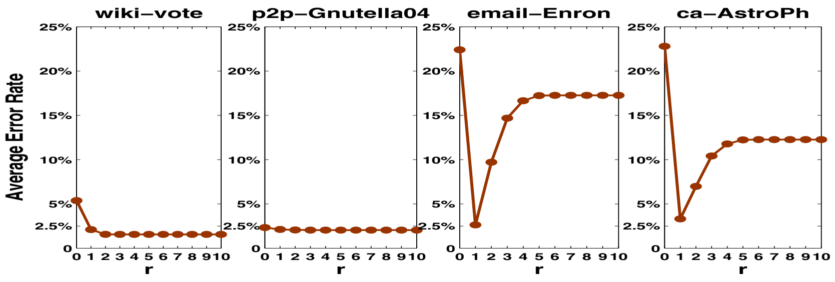

Fig. 11. Average Error Rate under Different Iteration Threshold Settings

From Fig.11 we can see that error rates under different iteration thresholds do not vary very much for the two directed networks, wiki-Vote and p2p-Gnutella04. These results are similar to that in Fig.6 and Fig.7, where the error rates of algorithms incorporating MIP heuristic(*GSbyStep* and *SSSbyStep*) are close to the algorithms not considering such heuristic(*GS* and *SSS*). In contrast, for the two undirected networks, email-Enron and ca-AstroPh, when setting iteration threshold to be $step[v] + 1(r = 1)$ we can get the smallest error rate.

In all, the above results verify that setting iteration threshold to be $step[v]+1$ is reasonable for both directed and undirected networks. In this way, we can not only achieve a better approximation for the real influence spread but also has low computational cost(as shown in Fig.10).

6 Conclusion

In this paper, we exploited the fact that propagation probabilities in real-world social networks are quite small for developing an iterative algorithm *GS* to quickly compute the real influence spread, under IC model. Specifically, we first explain that the influence spread can be well approximated by solving a linear system. Then, we point out the structure defect problem existing in social networks which bothers many iterative computation algorithms. Based on this discovery, we further improve both our *GS* algorithm and the *SteadyState-Spread* algorithm by incorporating the MIP heuristic. Experimental results on four real-world data sets demonstrate that *GS* algorithm can approximate the real influence spread very well. Meanwhile, the improved algorithms(*GSbyStep* and *SSSbyStep*) can achieve better approximations and save computational cost.

Acknowledgments. The work described in this paper was supported by grants from Natural Science Foundation of China (Grant No. 61073110), The National Major Special Science & Technology Projects (Grant No. 2011ZX04016-071), and Research Fund for the Doctoral Program of Higher Education of China (20113402110024), the Key Program of National Natural Science Foundation of China (Grant No. 60933013).

References

1. Aggarwal, C.C., Khan, A., Yan, X.: On Flow Authority Discovery in Social Networks. In: SDM, pp. 522–533 (2011)
2. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: KDD, pp. 7–15 (2008)
3. Aral, S., Muchnik, L., Sundararajan, A.: Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proceedings of the National Academy of Sciences 106(51), 21544–21549 (2009)
4. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: Membership, growth, and evolution. In: KDD, pp. 44–54 (2006)
5. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.A.: The Role of Social Networks in Information Diffusion. In: WWW, pp. 519–528 (2012)
6. Bus, J.C.P.: Convergence of Newton-Like Methods for Solving Systems of Nonlinear Equations. Numer. Math. 27(3), 271–281 (1977)
7. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: KDD, pp. 1029–1038 (2010)
8. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD, pp. 57–66 (2001)
9. Goldenberg, K.J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters 12(3), 211–223 (2001)
10. Grosan, C., Abraham, A.: Multiple Solutions for a System of Nonlinear Equations. International Journal of Innovative Computing, Information and Control 4(9), 2161–2170 (2008)
11. Keller, E., Berry, J.: The influentials: One American in ten tells the other nine how to vote, where to eat, and what to buy. The Free Press (2003)
12. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
13. 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)
14. Langville, A.N., Meyer, C.D.: Deeper Inside PageRank. Internet Mathematics 1(3), 335–380 (2004)
15. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. ACM Transactions on the Web 1(1), 5 (2007)
16. Lewis, K., Gonzalez, M., Kaufman, J.: Social selection and peer influence in an online social network. Proceedings of the National Academy of Sciences 109(1), 68–72 (2012)