Finding Influential Nodes by a Fast Marginal Ranking Method

Yipeng Zhang[1(✉)], Ping Zhang[2], Zhifeng Bao[1], Zizhe Xie[3], Qizhi Liu[3], and Bang Zhang[4]

[1] RMIT University, Melbourne, Australia
s3582779@student.rmit.edu.au
[2] Wuhan University, Wuhan, China
[3] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[4] CSIRO, Canberra, Australia

Abstract. The problem of Influence Maximization (IM) aims to find a small set of k nodes (seed nodes) in a social network G that could maximize the expected number of nodes. It has been proven to be #P-hard, and many approximation algorithms and heuristic algorithms have been proposed to solve this problem in polynomial time. Those algorithms, however, either trade effectiveness for practical efficiency or vice versa. In order to make a good balance between effectiveness and efficiency, this paper introduces a novel ranking method to identify the influential nodes without computing their exact influence. In particular, our method consists of two phases, the influence ranking and the node selection. At the first phase, we rank the node's influence based on the centrality of the network. At the second phase, we greedily pick the nodes of high ranks as seeds by considering their marginal influence to the current seed set. Experiments on real-world datasets show that the effectiveness of our method outperforms the state-of-the-art heuristic methods by 3% to 25%; and its speed is faster than the approximate method by at least three orders of magnitude (e.g., the approximate method could not complete in 12 h even for a social network of $|V| = 196{,}591$ and $|E| = 950{,}327$, while our method completes in 100 s).

1 Introduction

In this paper we study the Influence Maximization (IM) problem over social networks, which is first formulated as a discrete optimization problem by Kempe et al. [9]. Formally, its goal is to find a set of k nodes (seed nodes) in a social network G that could maximize the expected number of nodes under a given stochastic influence model. With the increasing popularity of social media, such as Facebook and Twitter, the influence maximization problem plays a critical role in effectively enabling large-scale viral marketing online. In the rest of this section, we will investigate the literature, identify the existing methods' drawbacks, and propose our solution.

J. Wang et al. (Eds.): ADC 2018, LNCS 10837, pp. 249–261, 2018.
https://doi.org/10.1007/978-3-319-92013-9_20

1.1 Related Work

Independent Cascade (IC) model and Linear Threshold (LT) model are two common influence models to capture the influence spread process. Under the two models, Kempe et al. [9] show that IM is NP-hard and present a MC greedy approximation algorithm (Greedy) guaranteeing that the influence spread is within $(1 - 1/e)$ of the optimal influence spread. In particular, the core idea is to calculate the influence of each individual node and take turns to choose the node maximizing the marginal influence until k nodes are selected. As computing the exact marginal gain (or the exact expected spread) under both the IC and LT models is #P-hard [3,4], Greedy tries to estimate the influence spread by running Monte Carlo (MC) simulations.

However, such a greedy method suffers from two major sources of inefficiency: (i) it has to repeatedly compute the influence for various seed sets, (ii) it needs to run a large number of Monte Carlo (MC) simulations to obtain a high-confidence result. Thereby, a plethora of algorithms are proposed to reduce the computation overhead [2,12,14], and they can be divided into two main categories.

Approximate Method. Approximate methods mainly focus on how to reduce the influence computation cost under certain models, while it can also provide $(1 - 1/e - \varepsilon)$-approximation. For IC and LT models, Leskovec et al. [12] present a 'lazy-forward' optimization for selecting new seeds, which greatly reduces the number of evaluations on the influence spread of nodes. Under IC model, Tang et al. [14] and Borgs et al. [2] use a reverse sampling approach to get the approximate influence and thus reduce the number of (MC) simulations greatly. However, these methods are time-consuming on large networks, e.g., the state-of-the-art method [14] needs to take four hours to find one seed from a graph $G(V, E)$ whose $|V| = 41.6 * 10^6$ and $|E| = 1.5 * 10^9$.

Heuristic Method. In general, heuristic methods are much faster than approximate methods since they only estimate the influence according to some simple heuristic rules; however, they cannot achieve an approximation ratio guarantee or a high influence value as approximate methods. In particular, Chen et al. [3] assume that the influence spread increases with the degree of nodes and present a degree discount heuristic method. Kimura and Saito [10] propose shortest-path based influence models and provide efficient algorithms for estimating influence under these models. Chen et al. [5] utilize the community structure to aggregate the features of nodes to reduce the number of nodes they need to check. PageRank [13] measures the centrality (or importance) of a node in a network, which is also used for selecting seeds in Influence Maximization. Chen et al. [3] show that it can achieve a competitive result compared to other heuristics. However, the heuristic rule cannot guarantee the effectiveness, thus the performance of a heuristic method would fluctuate dramatically on different networks.

1.2 Contributions

In this paper, we focus on the influence maximization problem under the IC model. In order to achieve a better trade-off between efficiency and effectiveness,

we propose a novel ranking method to select the influential nodes heuristically. Different from existing heuristic methods, our ranking metric is to measure the marginal influence of candidates rather than the influence itself. As a result, our proposed metric can help avoid the high influence overlap of the selected seeds to some extent, thus leading to a better effectiveness against existing heuristic measures. In particular, our contributions are summarized as follows.

- We measure the marginal influence of candidates iteratively based on PageRank and Personalized PageRank, and propose a ranking based greedy algorithm, called IRank, to identify the influential nodes. IRank achieves a better effectiveness on large-scale data than the heuristic methods (without losing much efficiency), which meets our primary goal of achieving an accurate seed selection. Meanwhile, its efficiency is at least three orders of magnitude faster than that of the approximate methods, which usually could not be completed in reasonable time. E.g., in our experiment, Greedy takes 2.4 h to run in the Email-Enron dataset at $k = 250$, while IRank only takes 8 s.
- We conduct a time complexity analysis and find that the computation cost of Personalized PageRank would increase as the growth of S. Thereby, in order to reduce the ranking cost, we propose InLocalPush to compute the Personalized PageRank incrementally and the cost of InLocalPush is independent of the size of S.
- We conduct extensive experiments on various real datasets to verify the efficiency, effectiveness and scalability of the proposed method.

2 Problem Formulation

2.1 Preliminaries

Independent Cascade (IC) Model. A social network can be represented by a (directed and undirected) graph $G(V, E)$, where V and E denote the users and their friendships respectively. To model the influence between users, we use a parameter p_{ij} $(0 \leq p_{ij} \leq 1)$ to represent the influence probability of node v_i to v_j. Note that, many studies [6,11] show that p_{ij} can actually be inferred from the historical influence cascade data. As this paper only focuses on the seed selection, we assume that p_{ij} is given in advance. The default p_{ij} is set as 0.01.

Given a seed set $S \subset V$, the IC model describes the influence spread by a discrete process as follows. Each node in the IC model has two states, active and inactive. Initially at time $t = 0$, we activate the nodes in S, while setting all other nodes inactive. At a time $t > 0$, any node v_i activated at time $(t - 1)$ has a single chance to influence each currently inactive neighbor v_j independently: it succeeds with the probability p_{ij} and fails with the probability $1 - p_{ij}$. If v_i succeeds, v_j will become active at time $t + 1$. This process runs until no more activations are possible. Here, the expected number of nodes activated by the process is the influence of S.

Let $\delta(S)$ denote the expected influence of S under IC model. Chen et al. [3] show that computing $\delta(S)$ is #P-hard by a reduction from the counting problem

of s-t connectedness [15]. As approximating the probability of s-t connectivity remains a long-standing open problem, it implies that we cannot expect to compute or even approximate $\delta(S)$ efficiently.

PageRank (PR). Let D and A be the degree matrix and the adjacency matrix of a graph $G(V, E)$ respectively. Given a subset $S \in V$, PageRank [13] can be described by the following equation:

$$pr(\alpha, \pi) = \alpha\pi + (1 - \alpha)\, pr(\alpha, \pi)W \tag{1}$$

where $pr(\alpha, \pi)$ is the pagerank vector of G, α is a constant in (0, 1] called the *teleportation constant*, π is a uniform distribution of V, and W is equal to $(I + D^{-1}A)$.

Personalized PageRank (PPR). A PageRank vector whose distribution π is concentrated on a smaller set of (targeted) nodes is called Personalized PageRank vector [1,7]. Therefore, Personalized PageRank is more generalized than PageRank and can be also described by Eq. 1. Given a target set S, we denote the Personalized PageRank vector of S as $pr(\alpha, S)$. Since our focus is to maximize the influence spread in social networks, please find more details of PageRank and Personalized PageRank in [1,7,13].

2.2 Problem Definition and Our Framework

Based on the IC model, we present the formal problem definition of IM as below.

Definition 1. *Given a social network $G(V, E)$ and a budget k, the problem of influence maximization is to find a k-size set S in V, such that S can maximize the expected influence spread $\delta(S)$.*

By Definition 1, our goal is to maximize the objective function $\delta(S)$ heuristically. As described in Sect. 1.1, PageRank is a good heuristic method for IM but far behind the MC greedy in term of effectiveness [3]. The main bottleneck is that the top-k ranking nodes can have large influence overlaps. It leads to the case that the influence of the union of seeds is much less than the sum of the influence of each seed.

In order to further improve the effectiveness of the current heuristic methods without sacrificing much efficiency degrade, we propose a ranking based greedy method to avoid selecting seeds with high overlaps. Given an initial seed set $S = \phi$, the core idea of our method is to select the influential nodes ranking by their marginal influence to S iteratively. Let $\mathcal{R}_s(v)$ $(v \in V)$ denote the ranking of the marginal influence of v to S. We note that the marginal influence can be decided by two parts: (1) the influence of v and (2) the overlap between v and S. According to this observation, we introduce a hybrid metric (as shown in Eq. 2) to measure $\mathcal{R}_s(v)$ by considering both parts aforementioned. In particular, the metric is represented by a linear combination of two items: the first item is PageRank (PR) of v which measures v's influence; the second item is the Personalized PageRank (PPR) of v (to S) which measures the overlap of v and S (in Sect. 3.1).

Table 1. Notations

Symbol	Description
$G(V, E)$	The social network
p_{ij}	The immediate influence from v_i to v_j
$\delta(S)$	The exact influence of S under the IC model
$\mathcal{R}_s(v)$	The marginal ranking of v (to S)

3 Our Solution

3.1 Influential Node Selection by Marginal Influence Ranking

In this section, we first introduce the ranking based algorithm called IRank and then present our ranking strategy. Important notations are presented in Table 1.

Algorithm 1 describes how our method works. It first initializes $S = \phi$ (line 1.3). In each iteration, IRank computes $\mathcal{R}_s(v)$ for each $v \in V \backslash S$ and picks the node with the highest ranking into S (lines 1.4 to 1.6). Here, CalMagRank(G, S) is used to compute $\mathcal{R}_s(v)$ for $\forall v \in V \backslash S$ (line 1.5). The algorithm repeats this process k times, and then returns S as the final solution. Our ranking metric of $\mathcal{R}_s(v)$ is defined as:

$$\mathcal{R}_s(v) = r(v) - \lambda r_s(v) \tag{2}$$

where $r(v)$ is to capture v's influence; $r_s(v)$ is to capture the overlap of v and S; and λ is a given parameter to balance the two items.

The Measurement of $r(v)$. Chen et al. [3] have shown that the nodes with the high influence usually have high PageRank values, which indicates that PageRank is a good indicator to measure the influence of nodes. Therefore, we compute $r(v)$ for each $v \in V$ by the PageRank of v.

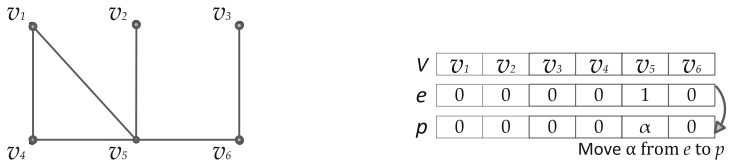
The Measurement of $r_s(v)$. In general, most individuals tend to trust the person who is important to their trusted ones. Given a seed set S and a candidate $v \in V \backslash S$, it indicates that the audiences of S tend to be influenced by v if v is very important to S, which implies that the influence overlap of v and S should be large. As a result, we compute $r_s(v)$ by the Personalized PageRank of v to S (Step 2), since PPR reflects the importance of v to S [1].

In this paper, we choose two most important algorithms, PageRank [13] and LocalPush [1], to compute the PR and PPR respectively. As PageRank is well-known, we only discuss how LocalPush works.

The core idea of LocalPush is to compute a pair of distributions p and e with the following property.

$$pr(\alpha, S) = p + pr(\alpha, e) \tag{3}$$

If p and e are two distributions with this property, Andersen et al. [1] indicate that p is close to the PPR vector $p(\alpha, S)$ when each element of $pr(\alpha, e)$ is sufficiently small. Note that, this claim is true because of the *Linearity* property of PPR which is discovered by Jeh and Widom [8].



(a) Graph (b) LocalPush Setting S={v_1}

Fig. 1. An example of LocalPush

Algorithm 1. IRank(G, k)

Input: a graph $G(V, E)$ and k
Output: a set S
$S \leftarrow \phi$
for $i \leftarrow 1$ *to* k **do**
  Invoke CalMagRank(G, S) to rank nodes in $V \backslash S$.
  Select the node with the highest ranking and add it into S.
return S

To obtain the PPR vector $pr(\alpha, S)$, LocalPush computes the p of Eq. 3 according to the following process. Each node $v \in V$ is assigned a pair of parameters: an *approximate ranking* $p(v)$ and a *residue* $e(v)$. Given a target set S as shown in Fig. 1a. Figure 1b shows how to initialize vectors p and e. Each $p(v)$ which $\forall v \in V$ is initialized by 0; and $e(v) = 1$ if $v \in S$, otherwise $e(v) = 0$. p (e) is the distribution of the approximate rankings (residues) of V. Note that, e are corresponding to $pr(\alpha, e)$ of Eq. 3. LocalPush maintains p and e by applying a series of push operations. Each push operation takes an arbitrary node v, moves an α fraction of the value from $e(v)$ into $p(v)$, and then spreads the remaining $(1-\alpha)$ fraction value to v's neighbors. Let d_v be the degree of v, each neighbor u of v receives $(1-\alpha)e(v)/d_v$ value from v and add it into $e(u)$. LocalPush returns p as the PPR vector of S when each element of e is less than a threshold ε.

Algorithm 2 is to compute $\mathcal{R}_s(v)$ for $\forall v \in V \backslash S$. In this algorithm, we use PageRank(G) [13] and LocalPush(G, S) [1] to compute $r(v)$ (PR) and $r_s(v)$ (PPR) respectively. It is worth noting that $r(v)$ is independent of S and does not change during the whole iterative process. Therefore, during implementation, we can store it in an n-dimension vector to avoid redundant computations.

The Time Complexity Analysis. The complexity of Algorithm 2 is determined by PageRank and LocalPush. Suppose PageRank should be terminated after a constant number of iterations. In each iteration, PageRank needs to traverse G and updates the ranking of each node, thus it takes $O(m + n)$ time. Moreover, the time complexity of LocalPush is the same as the PageRank at the worst case. Therefore Algorithm 2 takes $O(m + n)$ time. As Algorithm 1 invokes CalMagRank k times and k is a small constant, the complexity of IRank is $O(m + n)$.

Algorithm 2. CalMagRank

Input: a graph $G(V, E)$, a set S and λ
Output: a vector R
Initialize two vectors r and r_s.
`/* PageRank(`G`) algorithm is used to compute the pagerank in [13]. */`
Compute r by PageRank(G).
`/* LocalPush(`G, S`) algorithm is used to compute the PPR in [1]. */`
Compute r_s by LocalPush(G, S).
$R = r - \lambda r_s$
return R

Algorithm 3. InLocalPush(G, v, r_s)

Input: a graph $G(V, E)$, a node v and an n-dimension vector r
Output: a vector R
Initialize a vector r_v.
`/* LocalPush(`G, S`) algorithm is used to compute the PPR in [1]. */`
$r_v \leftarrow$ LocalPush(G, v).
$R \leftarrow r_v + r_s$
return R

4 Optimization

Incremental PPR Computation. According to Algorithm 1, when a new seed is added into S, IRank has to update $\mathcal{R}_s(v)$ for each node by calling CalMagRank. As mentioned above, as PR can be stored for reusing, the efficiency of IRank is mainly decided by LocalPush. To accelerate our method, we modify LocalPush to an incremental method (called InLocalPush) to further reduce the computation cost. In this section, we first show how InLocalPush works and then explain why it can reduce the cost.

InLocalPush is inspired by the following theorem introduced in [8].

Theorem 1. *Given two sets $S, S' \subseteq V$, and $S \cap S' = \phi$. Let $T = S \cup S'$, we have:*

$$pr(\alpha, T) = pr(\alpha, S) + pr(\alpha, S') \tag{4}$$

Let S_i be the seed set that is generated by IRank in the first ith iteration and $S_{i+1} = S_i \cup v$. According to Theorem 1, we have $pr(\alpha, S_{i+1}) = pr(\alpha, S_i) + pr(\alpha, v)$. As $pr(\alpha, S_i)$ is already obtained in the previous iteration, we only need to compute $pr(\alpha, v)$ to obtain $pr(\alpha, S_{i+1})$ rather than computing $pr(\alpha, S_{i+1})$ directly. It actually indicates that S can be computed incrementally.

Based on Theorem 1, we introduce InLocalPush to compute PPR incrementally, which is shown as Algorithm 3. Besides G, it has two extra input parameters, a node v and a vector r_s. When $r_s = pr(\alpha, S_i)$, InLocalPush returns the PPR vector of S_{i+1} as the result.

A Comparison of LocalPush and InLocalPush. Recall the description of LocalPush in Sect. 3.1, given a target set S, LocalPush initializes the two distributions p and e (according to S) and continues pushing the residues from e to p until each element of e is smaller than ε. Therefore, the cost of LocalPush is determined by the number of push operations. Intuitively, when ε is fixed, the larger the total residues are, the more push operations are needed to ensure the residue of each element in e small enough. As the sum of residues equals the cardinality of S, the cost of LocalPush should raise with the increase of the size of S. On the other hand, InLocalPush computes $pr(\alpha, S)$ incrementally and its cost is independent of the size of S. It implies that InLocalPush should be faster than LocalPush when $|S|$ is large, and thus Algorithm 2 can be benefited by replacing LocalPush by InLocalPush (line 3.4)[1].

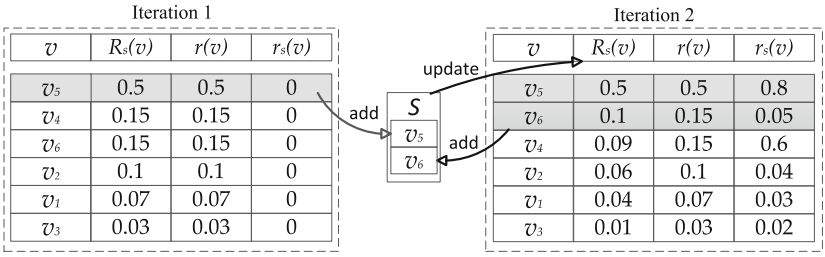


Fig. 2. A running example of IRank ($\lambda = 1$)

Example 1. *Figure 2 gives a running example of IRank for the instance of $G(V, E)$ in Fig. 1a. We initialize $S = \phi$, and assume $\lambda = 1$ and $k = 2$. At Step 1, $\mathcal{R}_s(v)$, $r(v)$ and $r_s(v)$ for each node $v \in V$ are listed in the left table. Note that, $\mathcal{R}_s(v) = r(v)$ and $r_s(v) = 0$ since the PPR of ϕ equals to 0. According to this table, v_5 is selected into S since it has the largest influence rank in this step. At Step 2, we update the rank of each $v \in V \backslash S$ and show the result in the right table. Therefore, v_6 is the optimal candidate and added into S. Since $k = 2$, IRank stops and returns $\{v_5, v_6\}$ as the final result.*

5 Experiments

In this section, we conduct four sets of experiments to evaluate: the impact of λ in effectiveness, the effectiveness, the efficiency, and the scalability of our method in real-life datasets.

[1] This modification also needs to vary the input parameters of Algorithm 2 from (G, S, λ) to (G, v, λ, r_s).

Table 2. Statistics of datasets

Datasets	Type	#Vertices	#Edges
Email-Enron	Undirected	36,692	183,831
Gowalla	Undirected	196,591	950,327
Twitter	Directed	11,316,811	85,331,846

Table 3. Parameter setting.

Parameter	Values
k	10, 20, 30, ..., 240, **250**
p	**0.01**, 0.02, ..., 0.09, 0.1
λ	1300, ..., **1350**, ..., 1400

5.1 Experimental Setup

Datasets. We use three real datasets, Email-Enron, Gowalla and Twitter. Email-Enron and Gowalla are two benchmark datasets that are obtained from http://snap.stanford.edu; Twitter is downloaded from http://socialcomputing.asu.edu/. All of them are social network datasets. The statistics of those datasets are shown in Table 2.

Algorithms. We compare our method against three algorithms: MC based greedy [9] (Greedy), Pagerank [13] (PR) and DegreeDiscount [3] (DD). To verify the effectiveness and efficiency of all algorithms, for each selected seed set, we run MC simulation 20000 times on the network

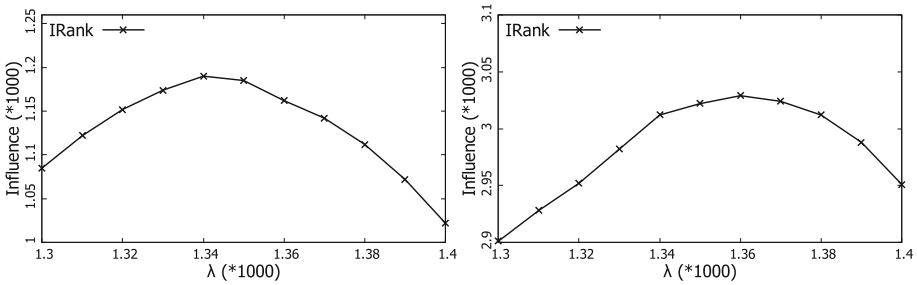
Setup. All algorithms are implemented using Java. and take the average of the influence spread as the benchmark. All experiments are conducted on a computer with two Intel(R) Xeon(R) E5630 3.0 GHZ processors and 48 GB RAM, running Ubuntu 10.04.

Parameter Setting. Table 3 shows the settings of all parameters, and the default one is highlighted in bold. In all experiments, we vary one parameter while the rest parameters are kept default unless specified otherwise. Note that, these networks do not capture the influence between the nodes explicitly. Therefore, we assume that the influence probability for each pair of neighboring nodes is 0.01, that is $p_{ij} = 0.01$ for $\forall v_i, \forall v_j \in V$, $e_{ij} \in E$.

Impact of λ. As we described in Sect. 3.1, λ is used to balance $r(v)$ and $r_s(v)$. In LocalPush, the sum of $e(v)$ and $p(v)$ equal to 1. Moreover, in these two vectors, the node that is the start point has the main part of possibility; it leads to a consequence that $r(v)$ is much bigger than $r_s(v)$. Therefore, we need a big λ for balance purpose. Figure 3 shows the influence of our method in Email-Enron and Gowalla by varying λ from 1,300 to 1,400. In both datasets, we find that the performance of IRank first increases as the growth of λ and then drops notably when λ is larger than 1360. It implies that our metric gets a good balance at $\lambda = 1350$ and thus we choose $\lambda = 1350$ as the default value.

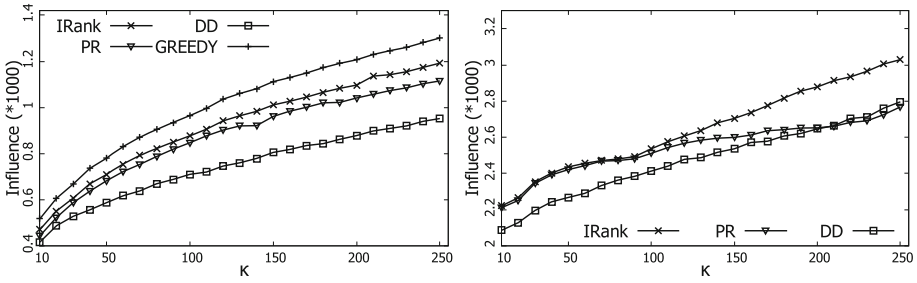
5.2 Effectiveness

We study how the influence spread is affected by varying k. Figure 4a shows that Greedy achieves the best performance, and it is followed by IRank, PR and DD. IRank is only at most 9% less than Greedy. Comparing to PR and DD respectively, IRank is 6.5% and 24.9% better at most. The reason is that, both



(a) Email-Enron (b) Gowalla

Fig. 3. Effectiveness of varying λ



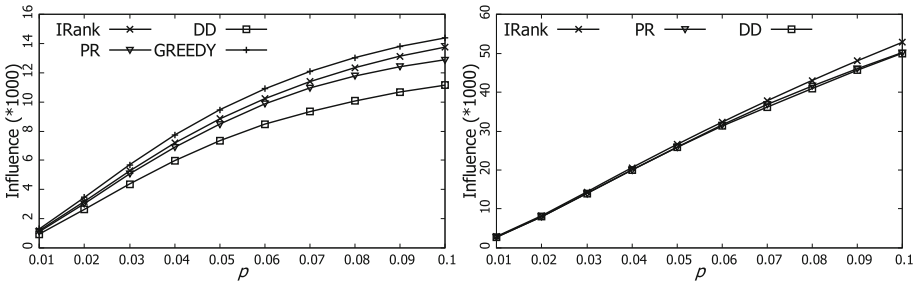
(a) Email-Enron (b) Gowalla

Fig. 4. Effectiveness of varying seed number k

PR and DD select the seeds according to the influence rank, and the influential nodes may have a high influence overlap between them. Clearly, the high overlap should diminish the total influence spread significantly. In contrast, IRank can avoid this issue because it is based on measuring the marginal influence.

Figure 4b reports the results for Gowalla dataset. After running 8 h, Greedy can only get 8 candidate nodes. As Greedy is too slow to complete in 12 h, we omit it. When k is small, IRank and PR have similar influence spread and both of them outperform DD by 10%. Note that when k increases, the influence of PR drops down and is close to that of DD, which is different to the observation in Fig. 4a; while IRank still beats PR and DD by around 8%. It indicates that PR is sensitive to k and its performance fluctuates in different datasets.

Figure 5 shows the results of increasing influence spread with increasing p. In Email-Enron dataset, although IRank is worse than Greedy by 6.6%, it is still much better than PR and DD. For example, at $p = 0.01$, the influence of PR is 93.5% of that of IRank, while the influence of DD is only 79.8% of that of IRank. In the Gowalla dataset, Fig. 5b shows that the influence spread of all seed sets as expected turns out to be a linear growth as the increasing of p. It is because that a higher influence possibility should increases influence spread of S. The performance of all algorithms is very close and IRank is slightly better



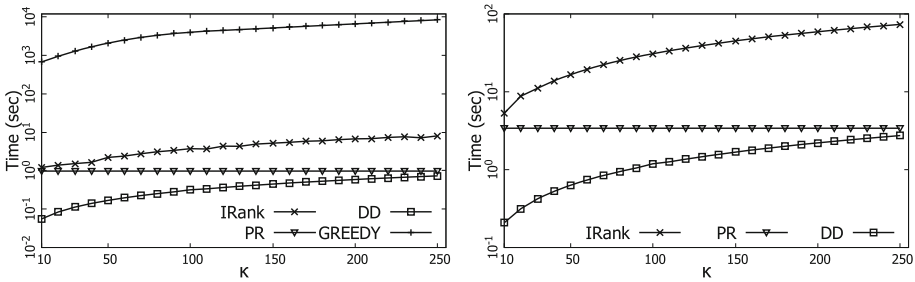
(a) Influence (Email-Enron) (b) Influence (Gowalla)

Fig. 5. Effectiveness of varying p

than the two heuristics by 5.4% and 6.8% respectively. It indicates that IRank is very robust to p and always better than PR and DD during the varying of p.

5.3 Efficiency

Figure 6 presents the efficiency results when k varies from 10 to 250. In Fig. 6a, we can see that Greedy is feasible to run in the Email-Enron dataset and takes 2.4 h at $k = 250$. In contrast, IRank only takes 8 s, which significantly outperforms Greedy by three orders of magnitude. We note that the running time of IRank and DD are proportional to the seed size k. For example, IRank and DD only take 1.2 and 0.05 s respectively at $k = 10$. However, when $k = 250$, their running time both increase one order of magnitude and are 8.0 and 0.73 s respectively. In contrast, the running time of PR does not change because its ranking process is independent of k. It indicates that when k is large, PR is more scalable than IRank and DD, although its influence spread is lower than the others.



(a) Email-Enron (log-scaled) (b) Gowalla (log-scaled)

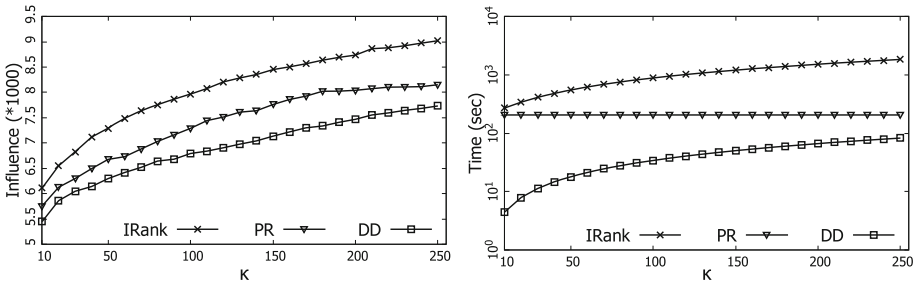
Fig. 6. Efficiency of varying seed number k

The observation is also verified in Gowalla dataset and its result is shown in Fig. 6b. Note that Greedy is out of this test since Gowalla is too large to

run Greedy. We can see that the result in Gowalla are similar to that in Email-Enron. In particular, the running time of DD ranges from 0.21 to 2.7 s when k varies from 10 to 250, and PR only takes 3.4 s since its cost is independent of k. Although IRank is slower than PR and DD around one order of magnitude, it is able to finish within 100 s.

5.4 Scalability

This experiment is to evaluate the scalability of the algorithms in large scale network Twitter. Figure 7a and b show the influence spread and the running time respectively. The result of Fig. 7a is consistent with the previous experiments that the effectiveness of IRank is better than that of PR and DD. Moreover, we can see the influence spread of IRank is relatively stable when k is varying. As shown in Fig. 7b, the runtime of both IRank and DD increases with the increase of k; although IRank is slower than DD, it still can handle million-sized graphs well.



(a) Influence (b) Time (log-scaled)

Fig. 7. Efficiency and effectiveness for Twitter

6 Conclusions

This paper introduces a ranking based method called IRank for the problem of Influence Maximization. In order to achieve a better trade-off between effectiveness and efficiency, we first rank the marginal influence based on PageRank and Personalized PageRank and use a ranking based greedy algorithm to select the influential nodes according to the rank iteratively. Moreover, the analysis show that the computation cost of Personalized PageRank would increase as the growth of S. To further reduce the cost, we accelerate IRank by computing Personalized PageRank incrementally and propose an incremental PPR algorithm InLocalPush. Empirical studies on a large real-world network show that IRank achieves a better effectiveness than the heuristic methods, which meets our primary goal of achieving an accurate seed selection. Meanwhile, its efficiency is at least three orders of magnitude faster than that of the approximate method.

Acknowledgement. This work is partially supported by the ARC (DP170102726, DP180102050), NSF of China (61728204, 91646204), and China National Key Research and Development Program (2016YFB1000700).

References

1. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: FOCS, pp. 475–486. IEEE (2006)
2. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing social influence in nearly optimal time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 946–957. SIAM (2014)
3. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: SIGKDD, pp. 1029–1038. ACM (2010)
4. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: ICDM, pp. 88–97. IEEE (2010)
5. Chen, Y.-C., Peng, W.-C., Lee, S.-Y.: Efficient algorithms for influence maximization in social networks. Knowl. Inf. Syst. **33**(3), 577–601 (2012)
6. Gomez Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: SIGKDD, pp. 1019–1028. ACM (2010)
7. Haveliwala, T.H.: Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)
8. Jeh, G., Widom, J.: Scaling personalized web search. In: WWW, pp. 271–279. ACM (2003)
9. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: SIGKDD, pp. 137–146. ACM (2003)
10. 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). https://doi.org/10.1007/11871637_27
11. Kurashima, T., Iwata, T., Takaya, N., Sawada, H.: Probabilistic latent network visualization: inferring and embedding diffusion networks. In: SIGKDD, pp. 1236–1245. ACM (2014)
12. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: SIGKDD, pp. 420–429. ACM (2007)
13. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab (1999)
14. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: near-optimal time complexity meets practical efficiency. In: SIGMOD, pp. 75–86. ACM (2014)
15. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**(3), 410–421 (1979)