J Inf Process Syst, Vol.10, No.3, pp.471~482, September 2014
http://dx.doi.org/10.3745/JIPS.04.0003

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

Efficient Greedy Algorithms for Influence Maximization in Social Networks

Jiaguo Lv*,**, Jingfeng Guo*, and Huixiao Ren*

Abstract—Influence maximization is an important problem of finding a small subset of nodes in a social network, such that by targeting this set, one will maximize the expected spread of influence in the network. To improve the efficiency of algorithm KK_Greedy proposed by Kempe et al., we propose two improved algorithms, Lv_NewGreedy and Lv_CELF. By combining all of advantages of these two algorithms, we propose a mixed algorithm Lv_MixedGreedy. We conducted experiments on two synthetically datasets and show that our improved algorithms have a matching influence with their benchmark algorithms, while being faster than them.

Keywords—Greedy Algorithm, Influence Maximization, Social Network

1. INTRODUCTION

Influence maximization is the problem of finding a small subset of nodes (seed nodes) in a social network so that their aggregated influence in the network is maximized. Due to the wide application wide application of influence maximization in viral marketing, it has been one of the most important research topics in social networks. Influence maximization was first studied as an algorithmic problem in [1,2]. Kempe et al. [3] formulate the problem as a discrete optimization problem. They show that the problem is NP-hard, and present a greedy algorithm (KK_Greedy), which achieves an approximation ratio of $1-1/e$. However, the KK_Greedy algorithm relies on the computation of the influence spread given a seed set, and on the use of Monte Carlo simulations on an influence cascade to estimate the influence spread, which makes the algorithm rather slow. To address this problem, a lot of work has been done to improve the efficiency of the algorithm. For the time constrained influence maximization problem, Liu et al. [4] proposed some influence spreading path based methods.

In this paper, we first analyze the inefficient sources of KK_Greedy, and then we propose three improved greedy algorithms, Lv_NewGreedy, Lv_CELF, and Lv_MixedGreedy. Our experiments on two synthetic datasets show that our algorithms outperform their original algorithms.

The remainder of the paper is organized as follows: Section 2 provides an overview of other

※ This work was supported financially by the Natural Science Foundation (60673136), Natural Science Foundation of Hebei Province (F2012209019), the Scientific Research Project of Universities of Hebei Province (QN201483) and the Science and Technology Infrastructure Platform Construction Project of Hebei Province (14960112D).

Manuscript received May 23, 2013; first revision August 7, 2013; accepted September 26, 2013; onlinefirst August 25, 2014.

Corresponding Author: Jingfeng Guo (jfguo@ysu.edu.cn)

* School of Information Science and Engineering, Yanshan University, Qinhuangdao, China (lvjiaguo2004@163.com, jfguo@ysu.edu.cn, 76542929@qq.com)

** School of Information Science and Engineering, Zaozhuang University, Zaozhuang, China (lvjiaguo2004@163.com)

Copyright ⓒ 2014 KIPS

related work. Section 3 details KK_Greedy and three improved algorithms for influence maximization. We provide a performance evaluation in Section 4. Finally, we present our conclusions in Section 5.

2. RELATED WORK

In this section, we first provide details about the KK_Greedy algorithm, and then present the recent works to improve the efficiency of KK_Greedy.

Table 1 lists the notations used in this paper. In this paper, a social network is a directed graph G=(V,E). Vertices in V are the nodes in the network and edges in E denote the relationship between a pair of nodes. The KK_Greedy algorithm is described in Algorithm 1. This algorithm builds the initial seed set one node at a time. It always greedily chooses the node with the largest marginal gain in influence.

Table 1. Notations

Terms	Description
S	Seed node set (in Algorithm 1, Algorithm 2, Algorithm 3, Algorithm 4)
K	The size of seed node set (in Algorithm 1, Algorithm 2, Algorithm 3, Algorithm 4)
getInfluenceSet(G,S)	A function which returns the influenced node set of S in G (in Algorithm 1)
$\sigma_G(S)$	The size of the influenced set of the seed set S in G (in Section 3.1)
$F_{Gi}(S)$	The reachable set from S in G_i (in Section 3.1)
MG(G,S,v)	|getInfluenceSet(G,S+{v})- getInfluenceSet (G,S)| (in Algorithm 2, Algorithm 4)
$G_i=(V_i, E_i)$	The graph that we get by running a random live edge selection process on G in iteration i (in Algorithm 2, Algorithm 4)
$G_i^S\ (V_i^S,\ E_i^S)$	The induced graph from G_i,where $V_i^S=V_i- F_{Gi}(S)$, $E_i^S=\{(u,v)\|u,v\in V_i^S,(u,v)\in E_i\}$ (in Algorithm 2, Algorithm 4)
SCC_i	The macro node that denotes the *i*th strong connected component in the induced graph G_i^S (in Algorithm 2, Algorithm 4)
sccCount	The number of strong connected components (in Algorithm 2, Algorithm 4)
u.mg1	The property of node u to denote the marginal gain of u for the current iteration (in Algorithm 2, Algorithm 4)
u.mg	The property of node u to denote the expected marginal gain of u for all iterations (in Algorithm 2, Algorithm 4)
Q<u,u.mg,mgset,u.flag>	A table for all candidate nodes. In Q, u.mgset is the marginal influenced set of node u for the current s, that is, u.mgset = getInfluenceSet (g,S+{u})-getInfluenceSet(g,S), u.mg=|u.mgset|, and u.flag is the number of iteration when u.mg was last updated. (in Algorithm 3, Algorithm 4)

Algorithm 1 KK_Greedy (G,K)

1. $S=\phi$;
2. While |S|<K do
3. $u=\underset{v\in V\setminus S}{\operatorname{argmax}}(|\ getInfluenceSet(G,S+\{v\})-getInfluenceSet(G,S)\ |)$
4. $S=S\cup\{u\}$
5. End While
6. Return S

In KK_Greedy, getInfluenceSet(G,S) is used to get the influenced node set by S in G. The major limitation of KK_Greedy is its inefficiency. The inefficiency is two-fold:

1) The computation of function getInfluenceSet(G,S) is computationally expensive with the Monte Carlo simulation.

2) There are too many candidate nodes that need to be examined by computing their marginal gain of influence.

In recent years, considerable work has been done to improve the efficiency of KK_Greedy. To address the first problem, a lot of excellent algorithms have been proposed, such as pattern-induced multi-sequence alignment (PMIA) in [5], NewGreedy in [6], LDAG in [7], and SPIN in [8]. To tackle the second problem, in [9] and CELF++ in [10] have been proposed. In addition, based on the community structure of social networks, many algorithms have been proposed, such as CGA in [11] and CGINA in [12].

3. IMPROVED ALGORITHMS FOR THE KK_GREEDY ALGORITHM

In this section, we try to improve the efficiency of KK_Greedy, as it has inefficient sources.

3.1 Improving the Efficiency of the Influence Function

The efficiency of evaluating the influence spread by the Monte Carlo simulation is very low. On the basis of the equivalent live edge selection process with the independent cascade model defined in [3], a new efficient method to evaluate the influence spread is proposed, which is similar to the method of NewGreedy in [6]. First, we selected a large enough integer R as the simulation times. Then, in iteration i, we ran the random live edge selection process on the original directed graph G, and get the graph G_i. Suppose, $F_{Gi}(S)$ is the reachable set from S in G_i, $\sigma_G(S)$ is the size of the influenced set of the seed set S, then, we have:

$$\sigma_G(S) = (1/R) * \sum_{i=1}^{R} | F_{G_i}(S) | \tag{1}$$

For the current seed set S in each iteration, from all the nodes in V–S, we greedily choose the node v with the maximal value of |getInfluenceSet(G,S+{v})- getInfluenceSet(G,S)|, and added it to S. For simplicity, MG(G,S,v) denotes the expression |getInfluenceSet(G,S+{v})-getInfluenceSet (G,S)|, and F_G (S) is the reachable set from node set S in graph G.

For every graph $G_i=(V_i,E_i)$ obtained from the random live edge process, with the concept of strongly connected component in graph theory, we could further reduce the computational complexity of $MG(G_i,S,v)$. From the graph theory, we know that for a strong connected component SCC_i in graph G and any two nodes u and v ($u,v \in SCC_i$ and $u \neq v$), $F_G(\{u\}) = F_G(\{v\})$. So, the steps for computing $MG(G_i,S,v)$ are as follows:

1) First, we computed the reachable set $F_{Gi}(S)$ for the current S. Then, for v in $S \cup F_{Gi}(S)$, $MG(G_i, S,v)=0$.

2) Let $V_i^S = V_i - F_{Gi}(S)$, $E_i^S = \{(u,v) | u,v \in V_i^S, (u,v) \in E_i\}$, then we could get the induced graph G_i^S (V_i^S, E_i^S) from G_i.

3) Get all of the strongly connected components from G_i^S. Since all nodes in a strong

connected component have the same reachable set, we used a macro node to denote a strongly connected component. Now, the macro node SCC_i and SCC_j denote the strongly connected components of SCC_i and SCC_j, respectively. If there is an edge in G_i^S from the nodes in the strongly connected components of SCC_i to SCC_j, we then added an edge from macro node SCC_i to SCC_j. Thus, we were able to get a macrograph SCC_i^S from the induced graph G_i^S.

4) Compute the reachable set for every node in G_i^S. Suppose, $F_{SCC_i^S}(SCC_i)$ is the reachable set of macro node SCC_i ($v \in SCC_i$) in macrograph SCC_i^S. Then for v in V_i^S, we have,

$$MG(G_i,S,v)=\sum_{scc\in F_{SCC_i^S}(SCC_i)} |scc| \tag{2}$$

Discussion. From the empirical results of social networks, we know that there are a lot of certain-scale strongly connected components in the graph. But, if this is not the case, the above method will be inefficient, and we will directly compute all reachable sets for every node as NewGreedy in [6]. So, in our algorithm, we introduced threshold θ, when the number of strongly connected components sccCount was less than $\theta*|V_i^S|$, we computed $MG(G_i,S,v)$ with the method detailed in Eq. (2), otherwise, we computed $MG(G_i,S,v)$ directly with the method in NewGreedy. So, we have:

$$MG(G_i,S,v)=\begin{cases} 0 & \text{if } v \text{ in } S\cup F_{G_i}(S) \text{ and sccCount} < \theta*|V_i^S| \\ \sum_{scc\in F_{SCC_i^S}(SCC_i)} |scc| & \text{if } v \text{ in } G_i^S \text{ and sccCount} < \theta*|V_i^S| \\ |F_{G_i^S}(\{v\})| & \text{if sccCount} \geq \theta*|V_i^S| \end{cases} \tag{3}$$

To obtain all of the strongly connected components from the graph, we adopted the Tarjan algorithm in [13], which has the complexity O(m+n). Our algorithm Lv_NewGreedy is described in Algorithm 2.

Algorithm2 Lv_NewGreedy(G,K)
1. S= ϕ ;Q= ϕ ;
2. For v in V do
3. v.mg1=0
4. v.mg=0
5. Add v to Q
6. End For
7. While |S|<K do:
8. For i=1 to R do:
9. $G_i(V_i,E_i)$=GetLiveGraph(G)
10. Compute $F_{Gi}(S)$
11. Compute $G_i^S(V_i^S,E_i^S)$

12. Get all SCC from G_i with Tarjan algorithm to SCCList
13. If | SCCList |<| V_i^S |* Θ then
14. SCC_i^S =GetSccGraph(G_i,SCCList)
15. For scc in SCCList do
16. Compute $F_{SCC_i^S}(scc)$
17. End For
18. End If
19. For v in $F_{Gi}(S)$ do:
20. v.mg1=0
21. v.mg=v.mg+v.mg1
22. End For
23. If | SCCList |<| V_i^S |* Θ then
24. For scc in SCCList do
25. mg=|scc.nodes|
26. For ascc in $F_{SCC_i^S}(v.scc)$ do
27. mg=mg+|ascc.nodes|
28. End For
29. For v in scc.nodes do
30. v.mg1=mg
31. v.mg=v.mg+v.mg1
32. End For
33. End For
34. Else
35. For v in V_i^S do
38. v.mg1=| $F_{G_i^S}(\{v\})$ |
37. v.mg=v.mg+v.mg1
38. End For
39. End If
40. End For //*R
41. For v in Q do
42. v.mg=v.mg/R
43. End For
44. u= $\underset{v \in V \setminus S}{\operatorname{argmax}}(v.mc)$
45. S=S+{u}
46. End While
47. Return S

Example. We consider the graph G_i shown in Fig. 1(a), where V_i={1,2,3,4,5,6,7,8,9,10,11}. Suppose, the current set is S={1}. In line 10, we can get $F_{Gi}(S)$={1,2,3,4}; in line 11, we can get the induced graph G_i^S shown in Fig. 1(b). In line 12, we can get strongly connected components, SCC_1={5,6,7}, SCC_2={8,9,10}, SCC_3={11}, which are shown in Fig. 2. Suppose threshold θ is 0.5, so, in lines 13-18, we can get the macrograph SCC_i^S, which is

shown in Fig. 3. And, we will get the reachable set of SCC_1, SCC_2, and SCC_3, that is {SCC1,SCC2,SCC3}, {SCC2,SCC3}, and {SCC3}. In lines 19-22, we can get $MG(G_i,S,v)$ for all nodes in $F_{Gi}(S)$, that is, 1.mg1=2.mg1=3.mg1=4.mg1=0. In lines 23-39, we can get $MG(G_i,S,v)$ for all nodes in V_i, that is, 5.mg1=6.mg1=7.mg1=|SCC1|+|SCC2|+|SCC3|=3+1+3=7, 8.mg1=9.mg1=10.mg1=|SCC2|+|SCC3|=4, 11.mg1=|SCC3|=1. In Algorithm 2, u.mg1 is used to store the marginal gain of u for the current iteration, and u.mg is used to get the expected marginal gain of u for all iterations.

3.2 Reducing the Number of Candidate Nodes that Need to be Examined

From KK_Greedy, we know that when we choose a new seed, any node in V-S would be examined as a candidate node. Clearly, this will greatly reduce the efficiency of the algorithm. With the submodularity of influence function, cost effective lazy forward (CELF) is proposed in [9]. In my opinion, when a new seed u is selected, any node v in the marginal influenced set of u should not be as a candidate node. Since v can be influenced by seed set S+{u}, all nodes that can be influenced by v can be influenced by S+{u} too. Based on this, we propose our improved algorithm Lv_CELF for CELF. In Lv_CELF, we maintain a table Q<u,u.mg,mgset,u.flag> for all candidate nodes. In Q, u.mgset= getInfluenceSet (g,S+{u})-getInfluenceSet(g,S), u.mg=|u.mgset|, and u.flag is the number of iterations when u.mg was last updated. In our algorithm, when a new seed u is selected, any node in u.mgset will be removed from Q. Algorithm Lv_CELF is described in Algorithm 3.

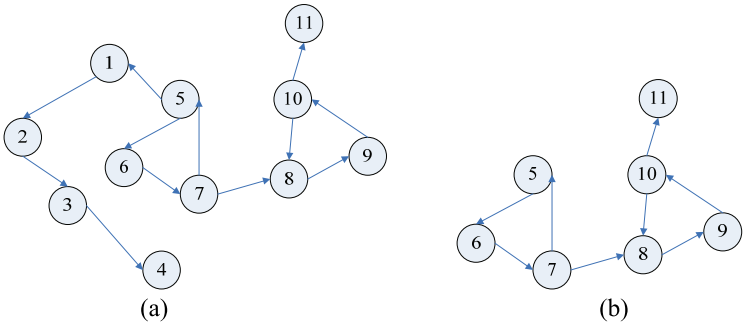


Fig. 1. An example of (a) graph G_i and (b) its induced graph G_i^S (S={1}).

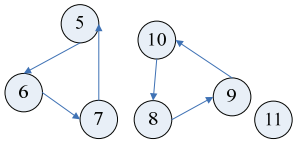


Fig. 2. All of the strongly connected components in G_i^S.

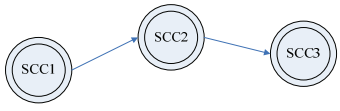


Fig. 3. Macrograph SCC_i^S.

Algorithm3 Lv_CELF (G,K)
1. S= ϕ ; Q= ϕ ;
2. For each v in V do
3. u.mgset=getInfluenceSet(G,{v})
4. u.mg=|u.mgset|

5. u.flag=0
6. add u to Q by u.mg in descending order
7. End For
8. While |S|<k and |Q|>0 do
9. u=Q[top]
10. if u.flag==|S| then
11. S=S+{u}
12. Q=Q-u.mgset
13. Else
14. u.mgset=getInfluenceSet(G,S+{v})- getInfluenceSet (G,S)
15. u.mg=|u.mgset|
16. u.flag=|S|
17. Resort Q by u.mg in descending order
18. End If
19. End While
20. Return S

3.3 A Mixed Algorithm for Influence Maximization

To improve the efficiency of the influence function, we are proposing the efficient algorithm Lv_NewGreedy. To reduce the number of calling influence functions, we are proposing the improved algorithm Lv_CELF. Based on the idea of MixedGreedy in [5], we are proposing an efficient algorithm Lv_MixedGreedy. In the first iteration, we used the method in Lv_NewGreedy to get u.mg for all u in V, then, we used the method in Lv_CELF to reduce the times of the calling influence function. Algorithm Lv_MixedGreedy is described in Algorithm 4.

Algorithm4 Lv_MixedGreedy(G,K)
/* initialize */
1. S= Φ ;Q= Φ ;
2. For u in V do
3. u.mg1=0; u.mg=0; u.mgset={};u.flag=0;
4. Add u to Q
5. End For
/* get u.mg for all u in V */
6. For i=1 to R do:
7. $G_i(V_i,E_i)$=GetLiveGraph(G)
8. Compute $G_i^S(V_i^S,E_i^S)$
9. Get all SCC from G_i with Tarjan algorithm to SCCList
10. If | SCCList |<|V_i|* θ then
11. $SccGraph_i$=GetSccGraph(G_i,SCCList)
12. For scc in SCCList do
13. Compute $F_{SCC_i^S}(scc)$
14. End For
15. End If
16. If | SCCList |<|V_i|* θ then

17. For scc in SCCList do
18. mg=|scc.nodes|
19. For ascc in $F_{SCC_i}(scc)$ do
20. mg=mg+|ascc.nodes|
21. End For
22. For v in scc.nodes do
23. v.mg1=mg
24. v.mg=v.mg+v.mg1
25. End For
26. End For
27. Else
28. For v in V_i do
29. v.mg1=$|F_{G_i^S}(\{v\})|$
30. v.mg=v.mg+v.mg1
31. End For
32. End If
33. End For //*R
34. For v in Q do
35. v.mg=v.mg/R
36. End For
37. Resort Q by v.mg in descending order
38. While |S|<k and |Q|>0 do
39. u=Q[top]
40. if u.flag==|S| then
41. S=S+{u}
42. If |u.mgset|=0 then
43. u.mgset= getInfluenceSet (G,S+{u})- getInfluenceSet (G,S)
44. end if
45. Q=Q-u.mgset
46. Else
47. u.mgset= getInfluenceSet (G,S+{v})- getInfluenceSet (G,S)
48. u.mg=|u.mgset|
49. u.flag=|S|
50. Resort Q by u.mg in descending order
51. End if
52. End While
53. Return S

4. EXPERIMENTS

To evaluate the efficiency and performance of Lv_NewGreedy, Lv_CELF, and Lv_MixedGreedy, we implemented them and other benchmark algorithms, such as NewGreedy, CELF, MixedGreedy, and PMIA, and conducted them on two synthetic networks. We were interested in comparing both the influence spread and the running time of these algorithms.

4.1 Dataset

By the benchmarks developed by Lancichinetti et al. [14], we obtained our experimental datasets. They are directed graphs. Basic statistics about these networks are given in Table 2. For the weight of every edge, we first assigned it a random value in [0,1]. Then, we normalized it by $W_{ij}=W_{ij}/\sum_i W_{ij}$. For the algorithms Lv_NewGreedy and Lv_MixedGreedy, the parameter θ is 0.5. For the threshold parameter of the baseline algorithm PMIA, we ran the algorithm numerous times with many threshold values. In our experiments, we found that PMIA achieves its matching influence spread with other algorithms when the threshold values are set to 1/320 and 1/80 for Data 1 and Data 2. In our experiments, the number of simulations R is 1,000. The experiments were run on a desktop computer with I5 2400S and 4 G memory.

Table 2. Statics of the datasets

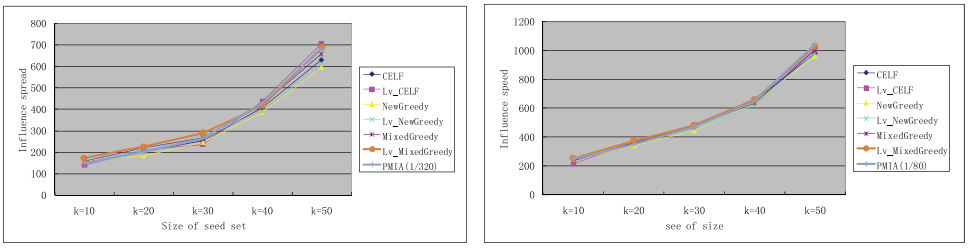
ID	Dataset	Node	Edge	Avg. degree	Number of SCCs
1	Data 1	14,780	40,159	4.8745	498
2	Data 2	7,200	86,968	39.7940	8

4.2 Experimental Results

4.2.1 Influence spread of our algorithms on two datasets

To evaluate the performance of our algorithms, we ran the algorithms NewGreedy, Lv_NewGreedy, CELF, Lv_CELF_Greedy, MixedGreedy, Lv_MixedGreedy, and PMIA on Data 1 and Data 2. In the experiment, we varied the size of the seed set k from 10 to 50. The influence spread of these algorithms on Data 1 and Data 2 is shown in Fig. 4.

As for the influence spread, we can see from Fig. 4 that the influence spreads of these algorithms are very close.



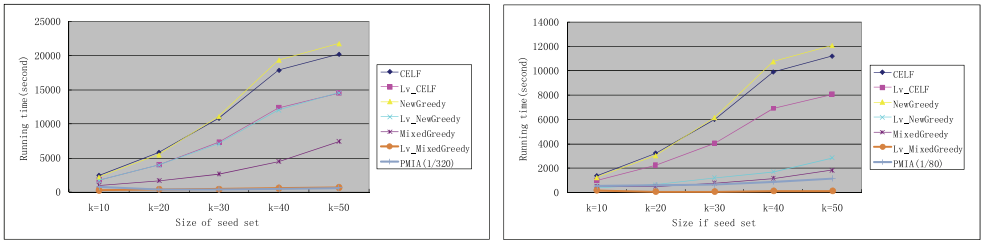
(a) (b)

Fig. 4. Influence spread vs. k on Data 1 (a) and Data 2 (b).

4.2.2 Running time of our algorithms on two datasets

The running times of these algorithms on Data 1 and Data 2 are shown in Fig. 5. From Fig. 5, we can see that algorithm Lv_CELF is faster than CELF; algorithm Lv_NewGreedy is faster than NewGreedy; and that algorithm Lv_MixedGreedy is faster than MixedGreedy. These results are consistent with what we expected. For the baseline algorithm PMIA in Data 1, we can see from Fig. 5 that it has the matching running time with algorithm Lv_MixedGreedy in Data 1.

However, it runs more slowly than algorithm Lv_MixedGreedy in Data 2. From Section 3.1, we can see that when the number of the strongly connected components is small and the scale of the strongly connected component is large, algorithms Lv_NewGreedy and Lv_MixedGreedy have greater superiority than other algorithms in terms of efficiency. From Table 2, we can see that the number of the strongly connected components is only 8. So, in Data 2, algorithm Lv_MixedGreedy clearly outperforms PMIA in terms of efficiency. Moreover, we can see that the number of strongly connected components in the network has a strong influence on the comparison of Lv_NewGreedy and NewGreedy. In summation, in our experiments Lv_MixedGreedy combines all if the advantages of Lv_CELF and Lv_NewGreedy, and has the highest efficiency out of all of the algorithms for influence maximization.



(a) (b)

Fig. 5. Running time vs. k on Data 1 (a) and Data 2 (b).

5. CONCLUSION

To improve the efficiency of KK_Greedy for influence maximization, we gave two reasons for the inefficient problem. To tackle the inefficient problem in the influence function, we have proposed the improved algorithm Lv_NewGreedy. In this algorithm, with the strongly connected component in graph theory, we greatly improved the efficiency of the influence function. To reduce the number of calling influence functions with the elimination of redundant candidate nodes, we have proposed the improved algorithm of Lv_CELF. Based on the idea of MixedGreedy, and by combining the advantages of Lv_NewGreedy and Lv_CELF, we have proposed a new algorithm called Lv_MixedGreedy. By having run experiments on two synthetically datasets, we showed that our improved algorithms have a matching influence with their benchmark algorithms, while being faster than them.

REFERENCES

[1] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 26-29, 2001, pp. 57-66.

[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 23-25, 2002, pp. 61-70.

[3] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, August 24-27, 2003, pp. 137-146.

[4] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *Proceedings of the IEEE 12th International Conference on Data Mining*, Brussels, Belgium, December 10-13, 2012, pp. 439-448.
[5] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, July 25-28, 2010, pp. 1029-1038.
[6] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, June 28-July 1, 2009, pp. 199-208.
[7] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *Proceedings of the IEEE 10th International Conference on Data Mining*, Sydney, Australia, December 13-17, 2010, pp. 88-97.
[8] R. Narayanam and Y. Narahari, "A shapley value-based approach to discover influential nodes in social networks," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 1, pp. 130-147, Jan. 2011.
[9] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, August 12-15, 2007, pp. 420-429.
[10] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "CELF++: optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the 20th international conference companion on World wide web*, Hyderabad, India, March 28-April 1, 2011, pp. 47-48.
[11] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-K influential nodes in mobile social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, July 25-28, 2010, pp. 1039-1048.
[12] J. Lv, J. Guo, and H. Ren, "A new community-based algorithm for influence maximization in social network," *Journal of Computational Information Systems*, vol. 9, no. 14, pp. 5659-5666, Jan. 2013.
[13] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146-160, 1972.
[14] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, 046110, Oct. 2008.

Jiaguo Lv

He received Master degree in computer science, in 2005, from the Yanshan University, China. Now, he is a student in Doctor course. His current research interests include social networks, data mining.

Jingfeng Guo

He is currently Professor at the Yanshan University, China. His current research interests include database theory and its application, data mining, social network, image processing, etc.

Huixiao Ren

She is currently a student in Master degree course in Yanshan University. Her current research interest is social network.