Journal of Engineering Science and Technology Review 7 (3) (2014) 32– 38

JOURNAL OF
Engineering Science
and Technology Review

www.jestr.org

Research Article

Improved Algorithms OF CELF and CELF++ for Influence Maximization

Jiaguo Lv[1,2], Jingfeng Guo[2,*] , Zhen Yang[1], Wei Zhang[1] and Allen Jocshi[3]

[1] *School of Information Science and Engineering, Zaozhuang University, Zaozhuang, 277100- China*
[2] *School of Information Science and Engineering,Yanshan University, Qinhuangdao, 066004-China*
[3]*Network Information Center for Design and Analysis, MCCN Ltd, 11952 Gdansk- Poland*

Received 15 December 2013; Accepted 2 June 2014

Abstract

Motivated by the wide application in some fields, such as viral marketing, sales promotion etc, influence maximization has been the most important and extensively studied problem in social network. However, the most classical KK-Greedy algorithm for influence maximization is inefficient. Two major sources of the algorithm's inefficiency were analyzed in this paper. With the analysis of algorithms CELF and CELF++, all nodes in the influenced set of u would never bring any marginal gain when a new seed u was produced. Through this optimization strategy, a lot of redundant nodes will be removed from the candidate nodes. Basing on the strategy, two improved algorithms of Lv_CELF and Lv_CELF++ were proposed in this study. To evaluate the two algorithms, the two algorithms with their benchmark algorithms of CELF and CELF++ were conducted on some real world datasets. To estimate the algorithms, influence degree and running time were employed to measure the performance and efficiency respectively. Experimental results showed that, compared with benchmark algorithms of CELF and CELF++, matching effects and higher efficiency were achieved by the new algorithms Lv_CELF and Lv_CELF++. Solutions with the proposed optimization strategy can be useful for the decision-making problems under the scenarios related to the influence maximization problem.

Keywords: Social Network, Influence Propagation, Influence Maximization, CELF,CELF++, Lv_CELF, Lv_CELF++

1. Introduction

Social network is used to describe a social structure made of nodes that is tied by one or more types of relationships. As the wide promotion and application of some huge social network websites, such as Face book, Twitter, etc, social network has played a more and more important role in information diffusion.

Motivated by the wide applications in viral marketing, influence maximization was first proposed and studied as a fundamental algorithmic problem by Domingos and Richardson in [1,2]. Viral marketing is a new marketing technique that employs the pre-existing social network for the promotion of products. In viral marketing, when an advertisement reaches a susceptible user, the user will be infected, and shares the idea with others. So, with the word of mouth from user to user, the advertisement will spread quickly in the network, which is similar to the spreading of viruses or computer viruses.

In viral marketing, a few "influential" members of the network will be targeted and given free samples of the products. In this way, these "influential" members could trigger a cascading influence since their friends will recommend the product to other friends, and many individuals will try it as a result. The wide application of the social network websites, such as Facebook and Twitter, provide opportunities to conduct large-scale online viral marketing in these social networks. There are two most important technologies that would enable such large-scale online viral marketing, the modeling of influence diffusion and the influence maximization problem.

This work focuses on influence maximization, which is the problem of finding a small set of most influential nodes (seed nodes) in a social network that can maximize the spread of influence. As for the influence maximization, it was first studied as an algorithmic problem in [1,2]. However, in [3], Kempe et al. proposed two basic stochastic influence cascade models, the independent cascade model (ICM) and the linear threshold model (LTM). The two models are extracted from the earlier works on social network analysis, interactive particle systems, and marketing. In ICM and LTM, a social network is modeled as a directed graph G (V, E), where V is the node set, and E denotes the edge set. In the graph G, a node v in V represents an individual in the network, and an edge (u, v) in E denotes the relationship from u to v. In ICM, each edge has an activation probability. The influence is diffused by those active nodes' independently activating their inactive neighbors based on their edge's activation probability. In LTM, each edge has an influence weight, and each node has a threshold chosen uniformly at random. An inactive node will be activated if the sum of its incoming edges' weight exceeds its threshold. In [3], Kempe et al. showed that both models could be generalized and their generalized versions were equivalent.

Under some diffusion model, the influence spread of a given seed set is the expected number of the activated nodes when the diffusion process ends. Given the budget k, the influence maximization problem is to find a subset S of node set V with k nodes, so that the influence spread of S is maximal. In [3], Kempe et al. showed that the influence maximization

* E-mail address: lvjiaguo2004@163.com

problem under both propagation models was NP-hard, and they proposed a greedy algorithm (KK-Greedy) for both models. Moreover, they proved that KK-Greedy algorithm could achieve an approximation ratio of 1-1/e.

However, KK-Greedy relies on the computation of the influence spread for a given seed set. In this algorithm, the influence spread is estimated by the Monte-Carlo simulations on influence cascade, which makes the algorithm very inefficient. For a moderate size graph of 30K nodes, the algorithm will run days to get a seed set of size 50.

To tackle the inefficiency of KK-Greedy, researchers have done a lot of work to improve the efficiency of the algorithm or to propose some new heuristic algorithms ([4],[5],[6], [7],[8],[9],[10]).

The novelties of this study are summarized next. First, two reasons of the inefficiency of KK-Greedy are analyzed. Second, basing on the algorithm of CELF in [8] and the algorithm of CELF++ in [9], two new algorithms of Lv_CELF and Lv_CELF++ are proposed. Third, experiments of these algorithms are conducted on some real world datasets. And the experimental results show that both of the two proposed algorithms run faster than CELF and CELF++.

Since Lv_CELF and Lv_CELF++ are orthogonal to the algorithms which optimize the estimating of the influence spread, the proposed algorithms can be combined with them to achieve a highly scalable algorithm.

The remainder of this paper is organized as follows. Section II surveys the related work of the propagation model and influence maximization. Section III presents two improved algorithms of Lv_CELF and Lv_CELF++ for influence maximization. Section IV presents the related experiments of the algorithms. Section V concludes this paper.

2. Related Work

Influence maximization was firstly studied as an algorithmic problem by Domingos and Richardson in [1, 2]. In their works, social networks were modeled as Markov random fields, and the probability of a node's being activated was a function of both its intrinsic value and its active neighbors' influence weights. Moreover, some methods that could approximately determine the influential nodes have been proposed in their works. However, influence maximization was firstly formulated as a discrete optimization problem by Kempe et al. in [3]. In their work, the researchers showed that the optimization problem was NP-hard, and then they proposed a greedy algorithm called KK-Greedy with an approximation of 1-1/e. In [8], Leskovec et al. proposed a "lazy-forward" strategy in choosing new seeds, which significantly reduced the number of influence spread evaluations. In [5], Chen et al. proposed a new heuristic algorithm named "degree discount" for the uniform ICM model in which all edge live probabilities were the same. The algorithm was very efficient. However, in later experiments, compared with KK-Greedy, the influence spread of the algorithm was rather poor. Instead of using Monto-Carlo simulations, in [5], Chen et al. also proposed a new algorithm called NewGreedy with an alternate "live-edge" selecting process. Moreover, combining with the algorithm of CELF in [8], they proposed a more efficient algorithm of MixedGreedy. With the local directed acyclic graph of each node, Chen et al. proposed an efficient algorithm named LDAG under the LTM in [6]. With the local arborescence structure of every node, Wang et al. proposed an efficient algorithm named PMIA under the ICM in [7]. To further improve the efficiency of CELF, Goyal et al. proposed an efficient algorithm named CELF++ in [9].

In addition to the literatures above, there are many other works for the influence maximization problem. Under the LTM, with the concept of Shapley value in cooperative game theory, a novel efficient algorithm called SPIN was proposed in [10]. Different from other works, the complexity of the influence maximization problem under deterministic linear threshold model was studied in [11]. The researchers showed that, for a given seed set, the exact computation of the influence could be solved in polynomial time. Under the susceptible/infected /susceptible (SIS) model, two influence maximization problems were defined in [12].By constructing a layered graph from the original work and applying the bond percolation with two effective control strategies, the authors solved the influence maximization problems effectively. To study the influence diffusion of competitive influences in social network, a game-theoretic model was introduced in [13]. The relation between the diameter of the network and the existence of the pure Nash equilibria of the game was studied. The authors showed that if the diameter was at most two then equilibrium existed in the game. Different from other algorithms, an efficient algorithm for influence maximization based on the ant colony optimization was proposed in [14]. Motivated by the resource and time constraints on viral marketing campaigns, two influence maximization problems (MINTSS and MINTIME) were studied in [15]. For MINTSS, the authors developed a simple greedy algorithm and showed that the algorithms could provide a bicriteria approximation. For MINTIME, they showed that even bicriteria and tricriteria approximations were hard to provide under several conditions. Taking users' preferences into account, a two-stage mining algorithm (GAUP) for mining most influential nodes on a specific topic was proposed in [16]. To solve the influence maximization problem based on a realistic model, two efficient algorithms were proposed in [17]. Employing the community structure, the number of candidates of influential nodes was reduced greatly. So, the two proposed algorithms were very efficient. Under the LTM, by selecting the nodes with maximal potential influence, a threshold-based heuristic algorithm (TBH) for influence maximization was proposed in [18].

2.1 ICM

Now, let's consider a directed graph G=(V,E) with edge labels pp. For each edge (u,v) in E, pp(u,v) is the propagation probability of the edge, which is the probability that v is activated by u through the edge in the next time step after u is activated.

Given a seed set S ($S \subseteq V$), the ICM works as follows. Let S_t ($S_t \subseteq V$) be the node set whose nodes are activated at step t ($t \geq 0$, and $S_0=S$). At step t+1, every node u in S_t may activate its out-neighbor v ($v \in V- \cup_{0 \leq i \leq t} S_i$) with independent probability pp(u,v). When S_t is Φ, the process will end at step t.

Especially, each active node has only one chance to activate its out-neighbors at the step that is right after the node itself is activated. In the network G, given the seed set S, the influence spread of S is denoted as δ(G,S), which is the expected number of the activated nodes.

2.2 KK-Greedy Algorithm

Given an input integer k, the influence maximization problem is to find a subset S^* ($S^* \subseteq V$, $|S^*|=k$), such that $\delta(G, S^*)=\max\{\delta(G,S) \mid S\subseteq V, |S|=k\}$.

From literature [3], it is known that the problem is NP-hard, but a constant-ratio approximation algorithm is available.

A set function f on nodes of graph G(V,E) is a function f: $2^V \rightarrow R$. Set function f is sub-modular if $f(S\cup\{v\})-f(S) \geqslant f(T\cup\{v\})-f(T)$ for all $v\in V-T$ and $S\subseteq T$. Intuitively, this means that the function f has diminished marginal return. Moreover, for all $S\subseteq T$, if $f(S)\leqslant f(T)$ holds, the function f is monotone.

For any sub-modular and monotone function f with $f(\varphi)=0$, the problem of finding a set S of size k that maximizes f(S) can be approximated by a simple greedy algorithm.

Basing on this, Kempe et al. proposed the classic greedy algorithm KK-Greedy in [3]. The algorithm builds the initial seed set one node at a time, and always greedily chooses the node with the largest marginal gain in influence.

The algorithm is described in Algorithm 1.

Algorithm 1 KK-Greedy(G,k)

```
1) S=φ;
2) While |S|<k do
3) {
4)     u=$\operatorname{argmax}_{v\in V\setminus S}(|\mathrm{getInfluence}(G, S\cup\{v\}) - \mathrm{getInfluence}(G,S)|)$;
5)     S=S+{u};
6) }
7) Return S.
```

In KK-Greedy, getInfluence(G,S) is used to get the influenced node set for the seed set S in network G. The major limitation of KK-Greedy lies in its inefficiency. The inefficiency is two-folds:

(1)With the method of Monte Carlo simulation, the computation of getInfluence(G,S) is very time-consuming;

(2)There are too many candidate nodes whose marginal gains of influence need to be computed. As a matter of fact, the algorithm will call function getInfluence O(nk) times, where n is the number of nodes in network G.

In recent years, to address the inefficiency of influence maximization algorithm, researchers have proposed a lot of excellent algorithms, like NewGreedy in [5], LDAG in [6], and PMIA in [7], etc. To reduce the times of calling the function getInfluence, researchers have proposed CELF in [8] and CELF++ in [9]. To decrease the number of candidate nodes and further improve the efficiency of influence maximization, two improved algorithms for CELF and CELF++ have been proposed in this work. Experimental results show that, compared with the original benchmark algorithms, the proposed algorithms have matching performance and higher efficiency.

3. Improved Algorithms for CELF and CELF++

This section will detail the two improved algorithms for CELF and CELF++.

3.1 CELF Algorithm

From the algorithm 1, it can be seen that, in KK-Greedy, in order to choose a new seed node, each node in set V-S will be examined to see if it has the maximal gain in influence. That is, each v in V-S is chosen as the candidate seed node, which greatly reduces the efficiency of KK-Greedy.

Employing the sub-modularity of influence function, basing on a “lazy-forward” optimization, Jure et al. proposed the algorithm of CELF in [8]. The idea behind CELF is that the marginal gain provided by a node in the current iteration cannot be better than the marginal gain provided by the node in the previous iteration.

The algorithm works as follows. It maintains a table Q<u, u.mg,u.flag> that is sorted by u.mg in decreasing order. In table Q, a record corresponds to a node in the network. In a record <u, u.mg,u.flag>, u.mg denotes the marginal gain of node u w.r.t S, and u.mg is the iteration number when u.mg is last updated.

Algorithm of CELF is outlined in Algorithm 2.

Algorithm 2 CELF_Greedy(G,K)

```
1) S= φ;
2) Q=φ;
// First iteration
3) For each u in V do
4) {
5)     u.mg=|getInfluence(G,{v})|;
6)     u.flag=0;
7)     add u to Q by u.mg in descending order ;
8)     }
//CELF
9) While |S|<k do
10) {
11)   u=Q[top];
12)   if u.flag==|s| then
13)   {
14)     S=S+{u};
15)     Q=Q-{u};
16)   }
17)   Else
18)   {
19)     u.mg=|getInfluence(G,S+{u})-getInfluence(G,S)|;
20)     u.flag=|S|;
21)     Resort Q by u.mg in descending order;
22)       }
23) }
24) Return S.
```

From algorithm 2, it can be seen that, in the first iteration, the marginal gain u.mg of every node u in V-S will be computed and the record of u will be added to Q in decreasing order of u.mg (lines 3-9). Then, in each iteration, for the first node u in Q, the algorithm will examine if u.mg is last computed in the current iteration. If yes, due to the sub-modularity, u must be the node with the greatest marginal gain, and will be selected as the current seed (lines 14-18). Otherwise, the marginal gain u.mg will be recomputed, the flag of u (u.flag) will be updated, and then the table Q with the new u.mg will be resorted (lines 19-24). Obviously, through the optimization, the algorithm of CELF avoids the re-computation of u.mg for each node u in repeated iterations.

3.2 Lv_CELF Algorithm

The main idea of Lv_CELF is as follows. When a new seed node u is produced, u.mgset is used to denote the marginal influence set of u under the current seed set S. Apparently, u.mgset=getInfluence(G,S+{u})-getInfluence (G,S).

When node u becomes the new seed node, for each node v in u.mgset, if node v will be selected as a new seed, it will never get any marginal gain in influence. Because node v can be activated by seed set S+{u}, all nodes that can be influenced by node v can be activated by seed set S+{u} too. Basing on this strategy, for algorithms of CELF and CELF++, two improved algorithms of Lv_CELF and Lv_CELF++ have been proposed.

In Lv_CELF, the algorithm maintains a table Q<u,u.mg,u.mgset, u.flag> that is sorted by u.mg in decreasing order. Since the node in u.mgset can be activated by seed set S+ {u}, so, when u is selected as the current seed, all nodes in u.mgset will be removed from table Q.
In this way, many redundant candidate seed nodes are removed from the table Q, and the times of re-computing u.mg is greatly reduced.

Algorithm Lv_CELF is described in Algorithm 3.

Algorithm 3 Lv_CELF _Greedy(G,K)
```
1)  S= φ;
2)  Q=φ;
3)  For each u in V do
4)  {
5)      u.mgset=getInfluence(G,{u});
6)      u.mg=|u.mgset|;
7)      u.flag=0 ;
8)      add u to Q by u.mg in descending order;
9)  }
10) While |S|<k and |Q|>0 do
11) {
12)    u=Q[top];
13)    if u.flag==|s| then
14)     {
15)       S=S+{u};
16)       Q=Q-u.mgset;
17)     }
18)    Else
19)     {
20)       u.mgset=getInfluence(G,S+{v})-
          getInfluence(G,S);
21)       u.mg=|u.mgset|;
22)       u.flag=|S|;
23)       Resort Q by u.mg in descending order;
24)     }
25) }
26) Return S.
```
From algorithm 3, it can be seen that the optimization of lv_CELF comes from line 16. In line 16, all nodes in u.mgset are removed from Q, which is different from CELF. By this way, the number of candidate nodes is reduced greatly, and then it improves the efficiency greatly.

3.3 CELF++ Algorithm

From literature [9], it can be seen that the algorithm of CELF++ is an improved version of CELF. Different from CELF, CELF++ remains a table Q<u.mg1, u.prev_best, u.mg2, u.flag> for all nodes that should be examined. In table Q, u.mg1 is the size of marginal influence set of node u under the current seed set S, namely, u.mg1= |getInfluence(G,S+{u}) -getInfluence(G,S)|; u.prev_best is the node with the maximum marginal gain among those nodes examined in the current iteration before u; u.mg2 denotes the size of marginal influence set of u under the seed set S+{u.prev_best}, namely, u.mg2= |getInfluence (G,S +{u.prev_best}+{u})- getInfluence(G, S+{u.prev_best})|; u.flag is the iteration number when u.mg1 is last updated.

The main idea of CELF++ is that if the node u.prev_best is chosen as a seed in the current iteration, the marginal gain of u w.r.t S+{prev_best} in the next iteration doesn't need to be computed again. The algorithm of CELF++ is detailed in algorithm 4. From algorithm 4, it can be seen that the optimization comes from lines 26-27.

Algorithm 4 CELF++_Greedy(G,K)
```
1)  S= φ;
2)  Q=φ;
3)  last_seed=null;
4)  cur_best=null;
5)  For each u in V do
6)  {
7)     u.mg1=|getInfluence(G,{u})|
8)     u.prev_best=cur_best;
9)     u.mg2=|getInfluence(G,{u.prev_best}+{u})-
             getInfluence(G, {u.prev_best})|;
10)    u.flag=0;
11)    add u to Q by u.mg in descending order;
12)    update cur_best based on u.mg1;
13) }
14) While |S|<k do
15) {
16)    u=Q[top];
17)    If u.flag==|S| then
18)    {
19)      S=S+{u};
20)      Q=Q-{u};
21)      Last_seed=u;
22)      cur_best=null;
23)    }
24)    Else
25)    {
26)      if (u.prev_best==last_seed) and (u.flag=|S|-1)
         then
27)         u.mg1=u.mg2
28)      else
29)       {
30)         u.mg1=|getInfluence(G,S+{u})-
                 getInfluence(G,S)|;
31)         u.mg2=|getInfluence(G,S+{prev_best}+{u})-
                 getInfluence(G,S+{prev_best})|;
32)         u.prev_best=cur_best ;
33)       }
34)     u.flag=|S|;
35)    }
36)     Resort Q by u.mg1 in descending order ;
37) }
38) Return S.
```

3.4 Lv_CELF++ Algorithm

Basing on the same idea as in Lv_CELF, the improved algorithm of Lv_CELF++ for CELF++ has been proposed in this work. Like the algorithms of CELF++, Lv_CELF++ maintains a table Q<u, u.mg1, u.mgset1, u.mg2,u.mgset2,u.flag>. In table Q, u.mgset1 and u.mgset2 denote the influence sets of u.mg1 and u.mg2 respectively; and other items have the same meanings as in CELF++.

Lv_CELF++ is described in Algorithm 5.

Algorithm 5 Lv_CELF++_Greedy (G, K)

```
1) S= φ;
2) Q=φ;
3) last_seed=null;
4) cur_best=null;
5) For each u in V do
6)  {
7)    u.mgset1=getInfluence(G,{u});
8)    u.mg1=|u.mgset1|;
9)    u.mgset2=getInfluence(G,{cur_best}|{u})-
                getInfluence(G,{cur_best});
10)   u.mg2=|u.mgset2|;
11)   u.prev_best=cur_best;
12)   u.flag=0;
13)   add u to Q by u.mg1 in descending order;
14)   update cur_best based on u.mg1;
15) }
16) While |S|<k and |Q|>0 do
17) {
18)   u=Q[top];
19)   If u.flag==|S| then
20)   {
21)     S=S+{u};
22)     Q=Q-u.mgset1;
23)     Last_seed=u; cur_best=null;
24)   }
25)   Else
26)   {
27)    if u.prev_best==last_seed and u.flag=|S|-1
         then
28)     {
29)       u.mg1=u.mg2;
30)       u.mgset1=u.mgset2;
31)     }
32)     else
33)     {
34)       u.mgset1= getInfluence(G,S+{u})-
                  getInfluence(G,S);
35)       u.mg1=| u.mgset1|;
36)       u.mgset2=getInfluence(G,S+{prev_best}+
                {u})-getInfluence(G,S+{prev_best})
37)       u.mg2=| u.mgset2|;
38)       u.prev_best=cur_best;
39)     }//if
40)     u.flag=|S|;
41)   }//IF
42)  Resort Q by u.mg1 in descending order
43) }//While
44) Return S.
```

3.4 Complexity Comparison

As discussed before, Lv_CELF is the improved version of CELF. For a candidate node u, the times of computing marginal gain in Lv_CELF is equal to that in CELF. However, with the adding of a new node to the current seed set S, more nodes are removed from the candidate node set. So, in terms of computational efficiency, Lv_CELF is higher than CELF. As described above, Lv_CELF++ is the improved version of CELF++. Because of the same idea behind in Lv_CELF and in Lv_CELF++, the performance comparison of Lv_CELF++ and CELF++ is similar to that of Lv_CELF and CELF.

4. Experiments

To evaluate the improved algorithms of Lv_CELF and Lv_CELF++, in this section, the algorithms of CELF, CELF++, Lv_CELF and Lv_CELF++ are implemented. And then the algorithms are conducted on some real world datasets. In terms of efficiency, the running times of these algorithms with different budget k are reported in this section. To evaluate the performances of the algorithms, the influence spreads of these algorithms with different budget k are reported in this section.

For all the experiments in this paper, the ICM has been selected as the influence propagating model.

This section first describes the data sets and the experimental setup, and then demonstrates the performance and efficiency of the algorithms on various data sets.

4.1 Dataset

The real world dataset in this study comes from Epinions. Epinions is a well-known website for product ratings. The format of the Epinons dataset is (id,whoid,whomid). It means that the user whoid trusts the user whomid. Basing on the dataset, the directed graph can be obtained. At the beginning of running algorithms, for every edge (u,v), a random value in [0,1] has been chosen as its activation possibility.

All the experiments in this paper are running on a desktop computer with:

(1)I5 2400S and 4GB of RAM;
(2) 32-bit Windows 7+Python2.6+Networkx0.9.

To evaluate the expected influence spread of see set S in the social network G, the Monte-Carlo simulations method is adopted. For each edge (u,v) , when node u is activated, it will try to activate v R times. If the number of activation exceeds R*0.6, u can activate v successfully. To evaluate the performance of the algorithms, three different datasets with various numbers of nodes and edges at random has been chosen.

The statics of these three datasets are shown in Tab.1.

Tab. 1. Statics of datasets

id	dataset	nodes	edges	Avg. degree
1	data1	730	1094	2.9973
2	data2	3798	10601	5.2981
3	data3	18643	95724	10.2692

4.2 Experimental Results

To evaluate the performances of the algorithms, the algorithms are conducted on dataset data1, data2 and data3. The experimental results are shown in Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5 and Fig. 6.

The running times of these algorithms on dataset data1, data2 and data3 are shown in Fig. 1, Fig. 3 and Fig. 5. From the figures, it can be seen that Lv_CELF is more efficient than CELF, and Lv_CELF++ is faster than CELF++. Moreover, algorithm of CELF++ is much slower than CELF, which is different from the results reported in [9]. Obviously, Lv_CELF is the most efficient one among them.

The influence spreads of these algorithms on dataset data1, data2 and data3 are shown in Fig. 2, Fig. 4 and Fig. 6. From the figures, it can be seen that the influence spreads of the four algorithms are matching.

As for the scalability, when the network is given, with the increasing of seed set size k, the running times of Lv_CELF and Lv_CELF++ increase slowly. However, for a given seed set size k, when the scale of the network is increasing, the running times of the two algorithms increase quickly. So, algorithms of Lv_CELF and Lv_CELF++ are not scalable.

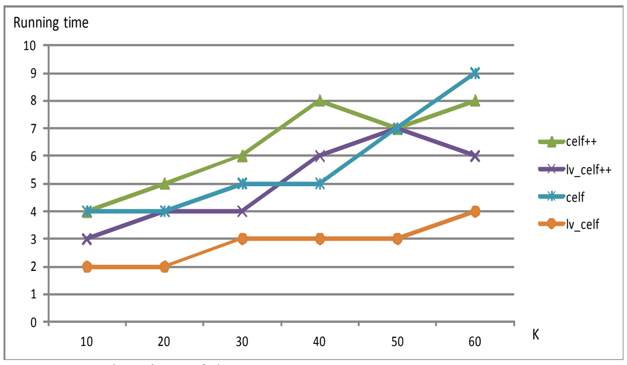

Fig. 1. Running time of data1

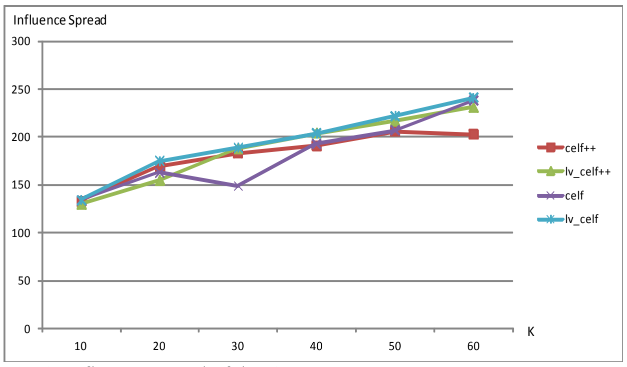

Fig. 2. Influence spread of data1

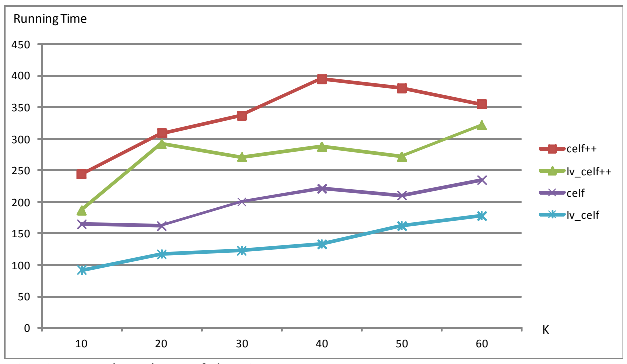

Fig. 3. Running time of data2

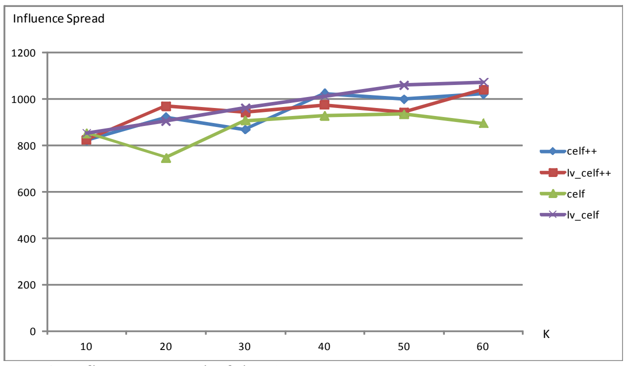

Fig. 4. Influence spread of data2

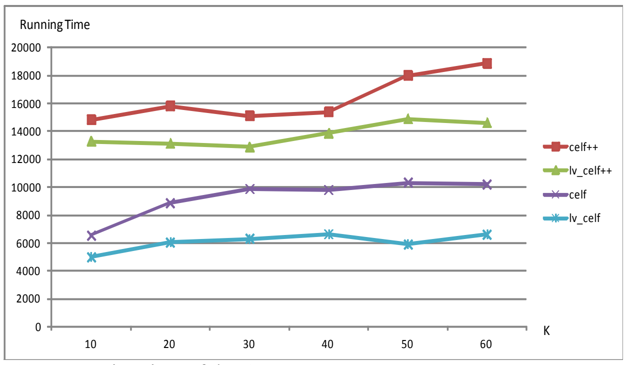

Fig. 5. Running time of data3

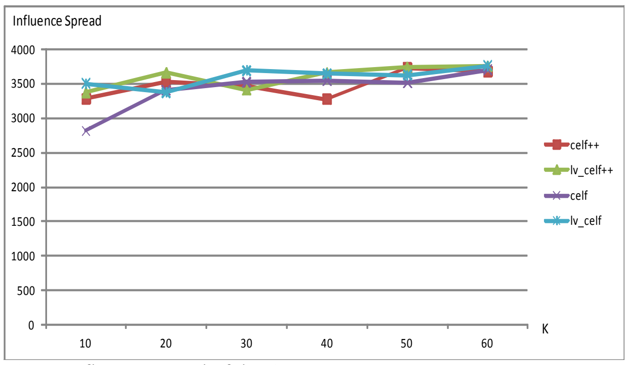

Fig. 6. Influence spread of data3

5. Conclusion

The reasons of the inefficiency of KK_Greedy were first analyzed in this paper. Then, in order to further improve the efficiency of the greedy algorithm for influence maximization in social network, two improved algorithm of Lv_CELF and Lv_CELF++ for CELF and CELF++ were proposed in this paper. Basing on the idea of eliminating the redundant candidate seed nodes, the improved algorithms further reduce the times of calling influence estimation function, and improve the performance of the greedy algorithm. The empirical studies on real world datasets show that the two algorithms can achieve matching influence spread with their benchmark algorithms, while being faster.

Acknowledgment

This work is partially supported by the scientific research project of universities and colleges in Hebei province (QN201483) and the Hebei province science and technology basic condition platform construction project (14960112D).
.

References

1. Domingos, P., Richardson, M., "Mining the network value of customers", Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2001, pp. 57-66.
2. Richardson, M., Domingos, P., "Mining knowledge-sharing sites for viral marketing", Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2002, pp. 61-70.
3. Kempe, D., Kleinberg, J., Tardos, É., "Maximizing the spread of influence through a social network", Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2003, pp. 137-146.
4. Chen, W., Wang, C., Wang, Y., "Scalable influence maximization for prevalent viral marketing in large-scale social networks", Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2010, pp. 1029-1038.
5. Chen, W., Wang, Y., Yang, S., "Efficient influence maximization in social networks", Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2009, pp. 199-208.
6. Chen, W., Yuan, Y., Zhang, L., "Scalable influence maximization in social networks under the linear threshold model", Proceedings of 2010 IEEE 10th International Conference in Data Mining, Washington, USA, 2010, pp. 88-97.
7. Wang C, Chen W, Wang Y., Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3), 2012, pp. 545-576.
8. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N. "Cost-effective outbreak detection in networks", Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2007, pp. 420-429.
9. Goyal, A., Lu, W., Lakshmanan, L. V., "Celf++: optimizing the greedy algorithm for influence maximization in social networks", Proceedings of the 20th international conference companion on world wide web, New York, USA, 2011,pp. 47-48.
10. Narayanam, R., Narahari, Y., "A shapley value-based approach to discover influential nodes in social networks", *IEEE Transactions on Automation Science and Engineering*, 8(1), 2011, pp.130-147.
11. Lu Z, Zhang W, Wu W, et al., "The complexity of influence maximization problem in the deterministic linear threshold model", *Journal of combinatorial optimization*, 24(3), 2012, pp. 374-378.
12. Saito K, Kimura M, Ohara K, et al., "Efficient discovery of influential nodes for SIS models in social networks", *Knowledge and information systems*, 30(3), 2012, pp. 613-635.
13. Alon N, Feldman M, Procaccia A D, et al., "A note on competitive diffusion through social networks", *Information Processing Letters*, 110(6), 2010, pp. 221-225.
14. Yang W S, Weng S X., "Application of the Ant Colony Optimization Algorithm to the Influence-Maximization Problem", *International Journal of Swarm Intelligence and Evolutionary Computation*, 1(1), 2012, pp.1- 8.
15. Goyal A, Bonchi F, Lakshmanan L V S, et al., "On minimizing budget and time in influence propagation over social networks", *Social Network Analysis and Mining*, 3(2), 2013,pp.179-192.
16. Zhou J, Zhang Y, Cheng J., "Preference-based mining of top-K influential nodes in social networks", *Future Generation Computer Systems*, 31, 2014, pp. 40-47.
17. Chen Y C, Peng W C, Lee S Y., "Efficient algorithms for influence maximization in social networks", *Knowledge and information systems*, 33(3), 2012, pp.577-601.
18. Tian J T, Wang Y T, Feng X J., "A new hybrid algorithm for influence maximization in social networks", *Jisuanji Xuebao* (*Chinese Journal of Computers*), 34(10), 2011, pp.1956-1965. (In Chinese)