SA-min: An Efficient Algorithm for Minimizing the Spread of Influence in a Social Network

Yong Liu[1, 2], Shengnan Xie[1], Wei Zhang[1], QianqianRen[1, 2]
[1]School of Computer Science and Technology
Heilongjiang University, Harbin, China
[2]Key Laboratory of Database and Parallel Computing of Heilongjiang Province
Harbin, China
acliuyong@sina.com

ABSTRACT: *Minimizing the spread of influence, a dual problem to influence maximization, is to find top-k links from a social network such that by blocking them the spread of influence is minimized. Kimura etal. [4] first proposed the problem and presented a greedy algorithm in order to solve this problem. But the greedy algorithm is too expensive and cannot scale to large scale social networks. In this paper, we design an efficient algorithm called SA-min based on Simulated Annealing (SA) for the problem. This is the first SA-based algorithm for the problem. Experimental results on real networks show that our algorithm can outperform the greedy algorithm by more than an order of magnitude while achieving comparable spread minimization.*

Keywords: Simulated Annealing, Social Networks, Influence Spread, Spread Minimization

Received: 10 January 2018, Revised 28 February 2018, Accepted 5 March 2018

DOI: 10.6025/jitr/2018/9/2/48-59

1. Introduction

In recent years, many large-scale social networks, such as Facebook, Twitter, Friendster, Microblog are becoming more and more popular. Social networks are playing a more and more important role in disseminating information. We usually regard a social network as a graph with nodes standing for persons and links standing for the connections between them [1]. Social networks can spread information and influence to a large number of people at a very fast speed. But social networks also allow viruses and other undesirable things to disseminate quickly. Thus, minimizing the spread of undesirable things through a social network is an important research problem in the data mining community.

In this paper, we focus on the problem of minimizing the propagation of undesirable things, such as computer viruses and malicious rumors, by removing a fixed number of edges in a social network. There are usually two ways to preventing the spread of influence: (a) deleting vertices in a social network; (b) deleting edges in a social network. Previous works usually focus on

deleting nodes from a network. They usually remove nodes by descending order of out-degree. Albert et al. [2], Callaway et al. [3] and Kimura et al. [4] show that when nodes are removed, links are also removed. But deleting edges is usually more effective than deleting vertices. Kimura et al. [4] propose a greedy algorithm by removing edges in a social network and show that the proposed methods is superior to the method which removes vertices. In their work, Kimura et al. [4] define the Contamination Minimization problem and greedily choose the edge that could minimize the average influence degrees of the whole nodes in the graph. The time complexity of the algorithm is $O(kn|E|\delta)$, where k is the number of edges that we need to choose, n stands for the number of nodes, $|E|$ is the number of edges, d represents the average out-degree in the network, and δ is the maximum time to calculate the average influence spread of single node. In order to accurately calculate the average influence spread of each node, the existing works usually use Monte Carlo (MC) simulation for many times. For example, 20000 times MC is applied in [5]. We call this greedy algorithm Greedy-min in the paper. When the size of the network is moderate, Greedy-min is still infeasible. In our experiments, Greedy-min cannot finish within 6 hours in a graph with 100 nodes when 10000 MC is applied. It is accordingly desirable to develop an efficient algorithm for Contamination Minimization problem that scales to large networks.

In this paper, we propose a new algorithm called SA-min for the Contamination Minimization problem. We apply the Simulated Annealing framework to the problem. The time complexity of SA-minis $O(nT\delta)$, where n stands for the number of nodes, T is the number of iterations, and δ is the maximum time to compute the influence spread of each node. The running time ratio between Greedy-min and SA-min is $k|E|/T$, thus, when $|E|$ and k become bigger, SA-min can run faster than Greedy-min by several orders of magnitude.

If we use MC simulation to compute the influence spread, the cost of SA-min is still expensive. To improve the efficiency of SA-min, we propose an efficient heuristic algorithm ML_CS for computing the influence spread. ML_CS only computes the influence of a node $v$on nodes that are at most m hops far away from v. The time complexity of ML_CS is $O(nd^m)$, where n stands for the number of nodes, d represents the average out-degree in the network, m is the number of hops. Therefore, the time complexity of SA-min is $O(nTd^m)$ when ML_CS is used in the framework of SA-min.

Our results on several real and synthetic networks show that SA-min can achieve almost matching results with Greedy-min and sometimes outperform Greedy-min. When ML_CS is applied in both SA-min and Greedy-min, SA-min is faster more than 30 times than Greedy-minon moderate social networks. Furthermore, SA-min scales beyond networks with tens of thousands of nodes where Greedy-min becomes infeasible.

The contributions of this paper are summarized as follows.

- We address the problem of minimizing the spread of influence in the social networks and propose a new algorithm called SA-min. It can obtain the matching result with Greedy-min. When the iteration number is large enough, SA-min can obtain a better result than Greedy-min.

- We propose an efficient heuristic algorithm ML_CS for computing the influence spread. It can accelerate the speed of SA-min.

- Extensive experiments demonstrate the effectiveness and efficiency of SA-min.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 introduces the preliminaries. Section 4 presents the problem definition and introduces the simulated annealing method. Section 5 provides the algorithm based on Simulated Annealing SA-min. Section 6 shows our experimental results. Conclusion and future work are presented in Section 7.

2. Related Work

Influence Maximizing is the dual problem to our problem. Kempe, Kltheein berg and Tardos [5]first formulated Influence Maximization problem as a discrete optimization problem. Given a directed graph $G = (V, E)$, a positive integer k, we aim to find k influential users, so that by activating them, the expected number of users that are activated is maximized under a given propagation model. Kempe et al. [5] investigate Influence Maximization problem on two classical diffusion models. One model is the independent cascade (IC) model and another model is the linear threshold (LT) model. Kempe et al. [5] show the discrete optimization problem is NP-hard and propose a greedy approximation algorithm that guarantees a $(1-1/e)$-approximation.

Since then, several studies present various optimization strategies to improve the efficiency of the greedy algorithm. Leskovec et al. [6] develop an efficient algorithm called CELF which is up to 700 times faster than the simple greedy algorithm. W. Chen et al. [7] propose a heuristic algorithm called degree discount to solve Influence Maximization problem. Qingye Jiang et al. [8] proposed an efficient approach based on Simulated Annealing (SA) to solve this problem and show that SA method can run faster than greedy algorithm and improve the accuracy of greedy algorithm. In addition to improving efficiency, the influence maximization problem is also extended to multiple variants [9,10].

Different from Influence Maximization problem, Contamination Minimization problem in social networks receives little attention. In 2000, Albert et al. [2] and Barabasi et al. [3] prevent the spread of influence in social networks by deleting nodes in accordance with the order of descending out-degrees. Kimura et al. [4] define the Contamination Minimization problem and present the greedy algorithm in order to choose top-k links in 2008. However, existing methods cannot scales to large networks because a large number of Monte Carlo simulations are applied.

3. Preliminaries

In the algorithm, we need to compute the influence spread of a node under the independent cascade (IC) model. Given a directed graph $G = (V, E)$, every edge is associated with a probability $P_{(u,v)}: \rightarrow E\,[0, 1]$, where $P_{(u,v)}$ denotes the chance that u actives v through the link after activating u. When a seed set $S \subseteq V$ is given, the IC model works as follows. We denote $S_t \subseteq V$ as the set of vertices which are activated at step $t \geq 0$ and initially we set $S_0 = S$. At $t + 1$ step, each node $u \in s_t$ only has one opportunity to activate his inactive out-neighbors $v \in V \setminus U_{0 \leq i \leq t} S_i$ with an independent probability $P_{(u,v)}$. If u successfully activates v, then v will become active at $t+1$ step. In the following rounds, u cannot attempt to active v any more. The procedure ends until $S_t = \varnothing$. The expected influence spread of S is denoted as $\delta(S; G)$ after the process ends.

We apply Simulated Annealing framework in our algorithm. We introduce the process of simulated annealing in this section. Metropolis et al. [15] proposed a new intelligent algorithm called Simulated Annealing in 1953. It is used to simulate the process of metal annealing. The purpose of the algorithm is to bring down the initial temperature of the system as possible as we can. We can use Simulated Annealing to optimize lots of NP-hard problems, such as Traveling Salesman Problem and other problems. Simulated Annealing works as follows. First, we construct an initial solution i and an initial system temperature $T = T_0$. Then the fitness function of the solution is computed, denoted by $f(i)$, which represents the initial energy of the system. Next, it seeks the neighbor solutions of the current solution and constructs a new solution j. If $\Delta f = f(j) - f(i)$ is negative, then the new solution is a better one and will replace the current one i; otherwise, the new solution j will replace the current one i with a probability, $P_{(i,j)} = exp(-\Delta f/T_t)$ where T_t is the current system temperature. During the whole process, we need to minimize the energy state of the system. The initial temperature T_0 must be set large enough. After lots of iterations for neighbor solutions, we cut down the current temperature $T_t = T_t - \Delta T$. The whole process ends until the temperature reaches the termination temperature T_f.

4. Problem Definition

In this paper, we use the same metric function as Kimura [4]. We use the contamination degree $c(G)$ of a graph G in order to represent the average influence degrees of all the nodes in G, i.e., $c(G) = \frac{1}{|V|}\sum_{v \in V} \delta(v; G)$. Here, $|V|$ stands for the number of nodes in graph G. For any link $e \in E$, let $G(e)$ represents the graph $G(V, E \setminus \{e\})$, thus $G(e)$ is created by deleting e from graph G. In the same way, for any $D \subseteq E$, let $G(D)$ represents the graph $(V, E \setminus \{D\})$, thus $G(D)$ is created by deleting D from graph G.

The **Contamination Minimization Problem** on graph G is defined as follows. Given a positive integer k, with $k < |E|$, find a subset D^* of E with $|D^*| = k$ such that $c(G(D^*)) < c(G(D))$ for any $D \subseteq E$ with $|D| = k$.

5. Algorithms

5.1 SA-min Algorithm

We define the fitness function of a solution $D \subseteq E$ as $c(G(D))$, which represents the average influence degrees of all the nodes in graph $G(D)$ under IC model. The fitness function $c(G(D))$ indicates the influence of every node in graph $G(D)$. In this case, we want to reduce the value of $c(G(D))$ as possible as we can. First, we create a neighbor solution A^* of the current solution A. If $\Delta c = c(G(A^*)) - c(G(A))$ is negative, i.e., the neighbor solution A^* is better, we replace the current solution A; otherwise,we generate a random number $\varepsilon \in (0, 1)$, if $exp(-\Delta c/T_t) > \varepsilon$, we replace the current one. When the number of inner iteration

researches q, we cut down the current temperature $T_t = T_t - \Delta T$, here T_t stands for the current temperature. When reaching the termination temperature T_f, the algorithm stops.

Algorithm 1 SA-min: SA based mining Top-K links

Input: graph $G = (V, E)$, a positive number k, initial temperature T_0, termination temperature T_f, the amount to cut down the current temperature ΔT, the number of inner loop q

Output: The set of Top-K links

1: $T_t \leftarrow T_0$, $count \leftarrow 0$;

2: Select an initial solution randomly $\subseteq E, |A| = k$;

3: **While** $T_t > T_f$ **do**

4: compute $c(G(A))$

5: $A^* = N(A)$ create a neighbor solution

6: $count$++;

7: Compute the change of the fitness function $\Delta c = c(G(A^*)) - c(G(A))$

8: **if** $\Delta c < 0$ **then**

9: $A \leftarrow A^*$;

10: **else**

11: Generate a random number $\varepsilon \in (0, 1)$

12: **if** $exp(-\Delta c / T_t) > \varepsilon$, **then**

13: A $\leftarrow$A* ;

14: **if** $count > q$ **then**

15: $T_t \leftarrow T_t - \Delta T$, $count \leftarrow 0$;

16: return A;

The pseudo-code is outlined in Algorithm 1

Complexity Analysis: The time complexity of the SA-min algorithm is $O(nT\delta)$, where n stands for the number of nodes, $T = \frac{T_0 - T_f}{\Delta T} q$ is the number of iterations, δ is the maximum time to compute the influence spread of each node. From the above analysis, we can see that SA-min can run $\frac{K|E|}{T}$ times faster than Greedy-min. when $|E|$ and k become bigger, SA-min runs several orders of magnitude than Greedy-min.

We will show that SA-min algorithm converges to the optimum when the iteration number t is large enough.

Theorem 1: The proposed algorithm based on Simulated Annealing for contamination minimization problem will converge to the optimum when the iteration number t becomes larger.

Proof. We notice that blocking the Top-k links corresponds to the Markov Chain. We refer to an edge set A_i as a state i, while A_i contains k edges. The neighbor solutions of A_i are denoted as $N(i)$. For a state $j \in N(i)$, the probability of generating j from i is $g_{i,j} = \frac{1}{k*(|E|-k)}$, here $|E|$ represents the number of links in the original graph. This is because that we choose one edge from the current solution randomly and choose one edge randomly from the rest links in order to replace the old link. The probability of accepting j is $a_{i,j} = min(1, exp(-\Delta c/T_t))$. Thus, the transition probability from state i to state j is:

$$\forall i,j, P_{i,j}(t) = \begin{cases} g_{i,j} a_{i,j}(t), & j \in N_i \text{ and } j \neq i \\ 0, & j \notin N_i \text{ and } j \neq i \\ 1 - \sum_{k \in N_i} P_{i,k}(t)), & j = i \end{cases}$$

The above finite state Markov Chain meets the following conditions:

(1). $\forall i,j \in \Omega(state\ union), g_{i,j}(t)$ is not related to t, and $g_{i,j} = g_{j,i}$;

(2). $\forall i,j,k \in \Omega(state\ union), if\ c(G(i)) \leq c(G(j)) \leq c(G(k))$; then, $a_{i,k}(t) = a_{i,j}(t) a_{j,k}(t)$

(3). $\forall i,j \in \Omega(state\ union), if\ c(G(i)) \geq c(G(j))$, then, $a_{i,j}(t) = 1$; $if\ c(G(i)) < c(G(j)), then, 0 < a_{i,j}(t) < 1$;

Thus, according to the work [11], for the stationary distribution of the Markov Chain $e = \{e_1, e_2, e_3 \ldots e_{|E|}\}$

$$\lim_{t \to \infty} e_i(t) = \begin{cases} \frac{1}{|\Omega_{opt}|}, & i \in \Omega_{opt} \\ 0, & i \notin \Omega_{opt} \end{cases}$$

Ω_{opt} is the union of optimal solutions. This demonstrates that our problem will tend to converge to optimum when the iteration number becomes large.

5.2 ML-CS Algorithm

To compute $c(G(A))$ for any $A \subseteq E$, the existing algorithms usually employ Monte-Carlo simulations. This is computationally expensive. We propose an efficient algorithm ML_CS to compute $c(G(A))$ efficiently. We estimate the influence spread of one node by only computing the m-hop area instead of $\delta(v; G)$, when the propagation probability p is small. For each $v_i \in V$, let $u_{i,j}$ be the number of j-hop paths ($j \in (1, 2....m)$) from the origin node to v_i. In order to further illustrate the algorithm, we give an

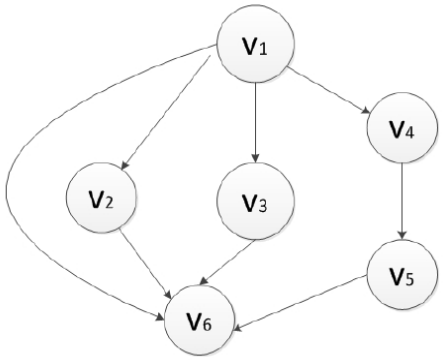

Figure 1. The directed graph G

example. In Figure 1, we compute the influence degree of node. From the graph, we can see the 1-hop path are $\{(v_1, v_6), (v_2, v_6), (v_3, v_6), (v_5, v_6)\}$, thus the number of $v_{6.1}$ is 4; the 2-hop path are $\{(v_1, v_2\, v_6), (v_1, v_3\, v_6), (v_4, v_5\, v_6)\}$, thus the number of $v_{6.2}$ is 3; the 3-hop path is $(v_1, v_4, v_5\, v_6)$, thus the number of $v_{6.3}$ is 1. The probability that will be activated is $1-(1-p)^4(1-p^2)^3(1-p^3)^1$.

Algorithm2 ML_CS: compute $c(G(A))$

Input: Graph $G(V, E)$, a positive number m, propagation probability p, the current solution $A \subseteq E$;

Output: The contamination degree $c(G(A))$ of the current solution $A \subseteq E$;

1. $Sum = 0$;

2. **for** each $e \in A$ **do**

3. $E \leftarrow E - \{e\}$; // delete e in graph G

4. Endfor

5. **for** $i = 1$ to $|V|$ **do**

6. $f = 1$;

7. **for** $j = 1$ to m **do**

8. **for** each j-hop neighbor u of v_i **do** // compute the number of j-hop to node v_i

9. $v_{i,j} = v_{i,j} + 1$;

10. **Endfor**

11. **Endfor**

12. **for** $j = 1$ to m **do**

13. $f = f * (1-p^j)^{v_{ij}}$ // compute the probability that v_i is not activated by its j-hop neighbour

14. Endfor

15. $Sum = Sum + (1 - f)$;// add up the inûuence spread of each node $v_i \in V$

16. **Endfor**

17. $c(G(A)) = Sum / |V|$;

18. Return $c(G(A))$;

The pseudo-code of ML_CS is outlined in Algorithm 2.

Complexity Analysis: The time complexity of the ML_CS algorithm is $O(nd^m)$, where d stands for the average out-degree of the network, m is the number of hops. The time complexity of previous work for computing $c(G(A))$ is $O(nR|E|)$. ML_CS can provide $\frac{R|E|}{d^m}$ speed-up ratio.

6. Experimental Evaluation

We conduct experiments for three algorithms on several real-life and synthetic networks. In the experiments, we want to compare our SA-min algorithm with other algorithms from two aspects: (a) its scalability (b) its contamination degree. The first algorithm is Random method, which removes links uniformly at random; the second one is Greedy-min, which blocks the edges that can minimize the contamination degree greedily.

6.1 Experimental Setup

The first network is obtained from Amazon website [12]. Products stand for nodes. If two products are purchased at the same time, the corresponding graph has a directed edge between them. The raw data contains n = 262111 nodes and m = 1234877 edges. We extract the first 500 nodes and their relevant 1911 edges. We denote the subgraph as Amazon-500. The second network is Who-trust-whom network of Epinions.com [13]. We obtain the data from the network. The consumers in the network represent nodes. Users in the network can show their trust to other users. If user i trust user j, there is a directed edge in the corresponding graph. The raw data includes n = 75879 nodes and m = 508837 edges. We extract the first 100 nodes and 2172 edges that connected to the nodes. We denote the subgraph as Epinions-100. The average out-degree of Epinions is higher than that of Amazon. Therefore, these two networks have different features. We compare our algorithm with Random and Greedy-min on different networks. In order to compare the scalability of different algorithms, we also extract n = 2000 nodes, m = 8145 edges; n = 5000 nodes, m = 20426 edges; n =10000 nodes, m = 41760 edges; n = 20000 nodes, m = 85535 edges on Amazon dataset.

We employ ML_CS algorithm to compute the influence spread in both SA-min and Greedy-min. For SA-min algorithm, we set T_0 = 100000, T_f = 10000. All experiments are performed on an Intel(R) Core(TM) 2 Duo CPU, 2GB memory. We conduct our experiments on Microsoft Visual Studio 2010 environment.

6.2 Experimental Results

6.2.1 Varying the Propagation Probability *p*

In Figure 2, we compare the contamination degree when p changes. From the tendency of curve, we can see that when the propagation probability increases, the contamination degree increases too. SA-min can provide the matching result with Greedy-min. When p = 0.25, the contamination degree of SA-min is about 8.25% smaller than that of Random.

6.2.2 Varying the Value of *k*

In the following experiments, we conduct experiments on Amazon-500 and Epinions-100 respectively. In Amazon-500, the number of nodes is 500 and the number of links is 1911. In Epinions-100, the number of nodes is 100 and the number of edges is 2172. They have different features. We compare the contamination degree when blocking different number of links with p = 0.3 on two graphs.

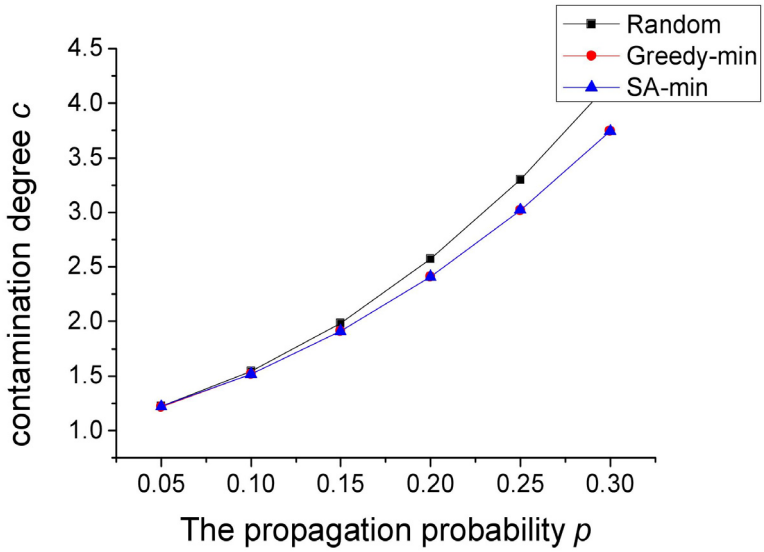

Figure 2. Contamination degree of SA-min compared with Random and Greedy-min when k = 50 on Amazon-500

From Figure 3(a), and Figure 3(c), we can see that SA-min can run close to the result of Greedy-min in terms of contamination degree. In Figure 3(a), when blocking 60 links, the contamination degree of SA-min is 4.0302, the result of Greedy-min is 4.0313. SA-min outperforms Greedy-min by 0.27%. In Figure 3(a), when k is 70, the contamination degree of SA-min and Greedy-min is 3.9855 and 3.9864 respectively. SA-min outperforms Greedy-min by 0.23%. In Figure 3(a), SA-min can exceed Random by about 9.3% when k is 70. Hence, we can conclude that SA-min is effective. From Figure 3(b) and Figure 3(d), we can see that when the number of blocking edges becomes large, the running time of Greedy-min is linear growth. At the same time, the running time of SA-min decreases. In Figure 3(b), when k is 70, SA-min runs 3.8 times faster than Greedy-min. Moreover, SA-min is more stable when removing more links compared with Greedy-min.

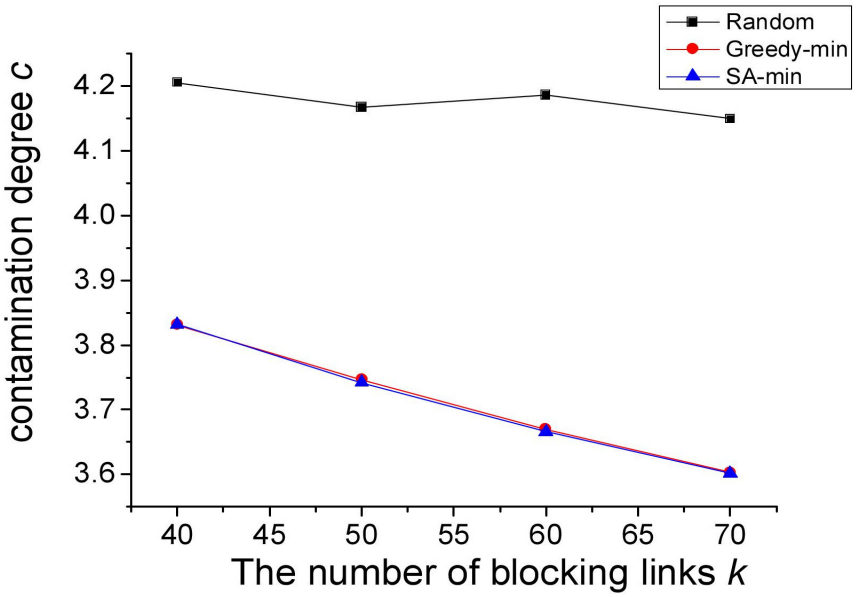

Figure 3(a). Contamination degree of SA-min compared with Random and Greedy-min when p=0.3 on Amazon-500

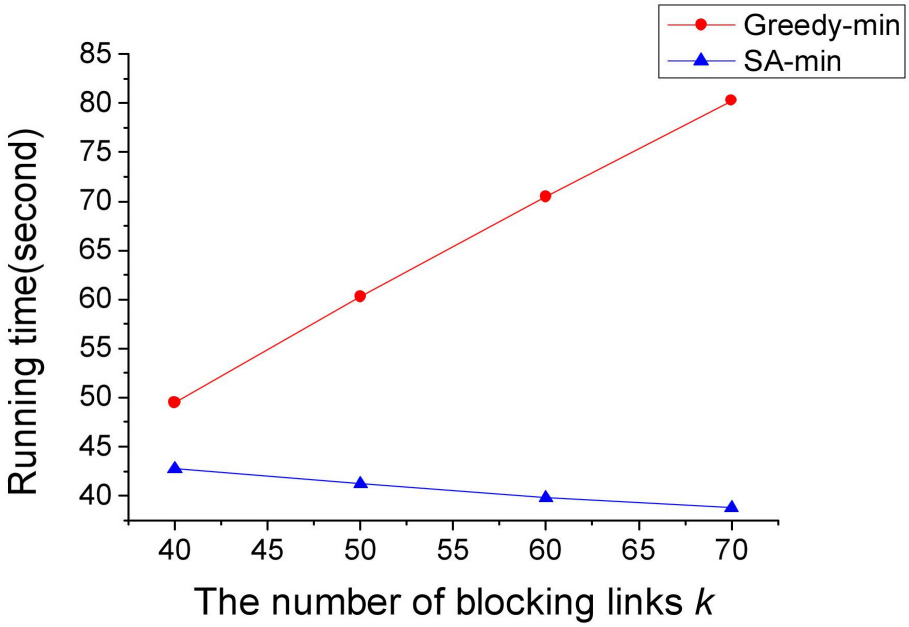

Figure 3(b). Running time of SA-min compared with Greedy-min when p=0.3 on Amazon-500

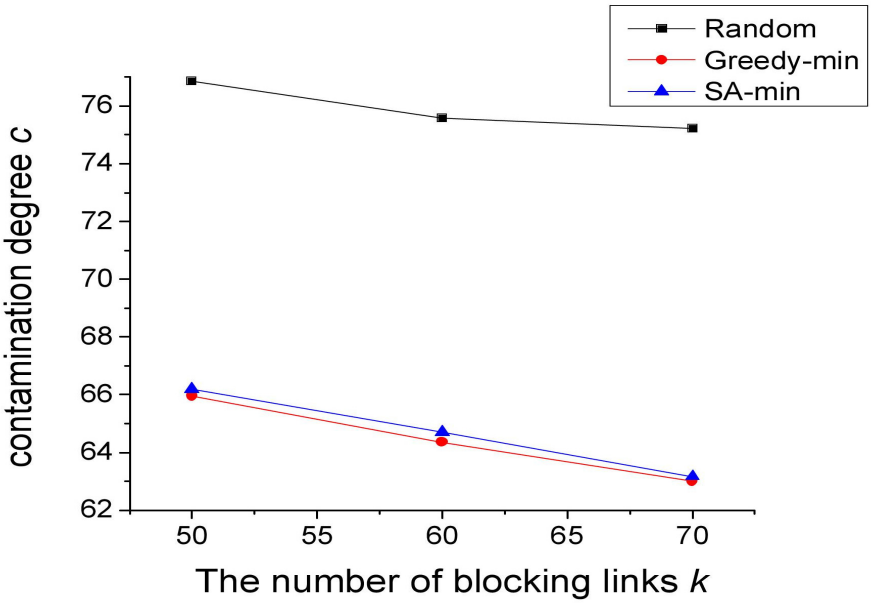

Figure3(c). Contamination degree of SA-min compared with Random and Greedy-min when $p = 0.3$ on Epinions-100

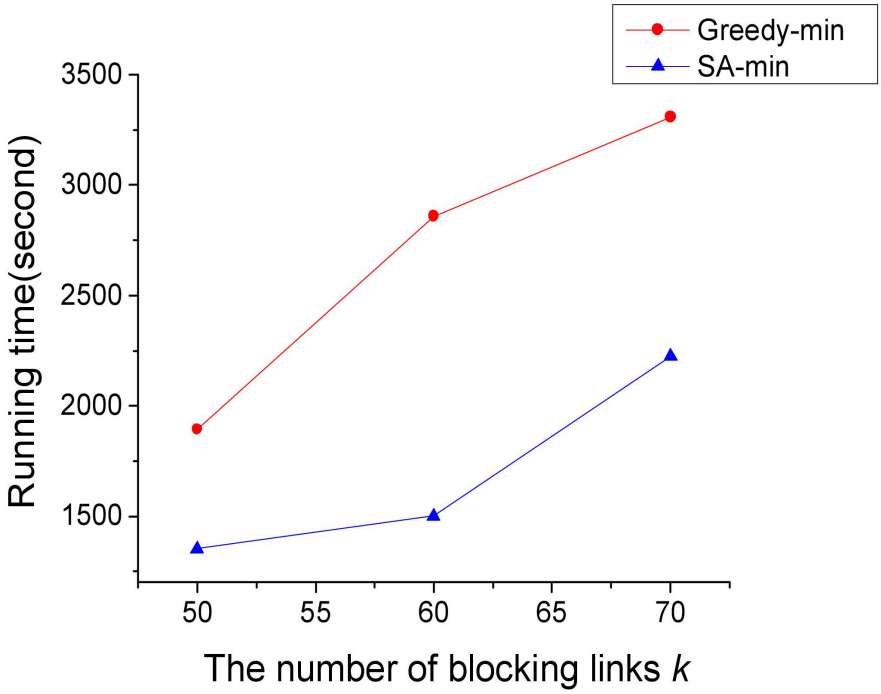

Figure 3(d). Running time of SA-min compared with Greedy-min when $p = 0.3$ on Epinions-100

6.3 Varying the Value of ΔT

In the following experiments, we compare the contamination degree when ΔT changes on Amazon-500 with $p = 0.3$, $k = 50$. From Figure 4, we can see that when the amount of cutting down the temperature becomes larger, the number of iteration decreases, so the contamination degree becomes bigger. When ΔT is 400, SA-min outperforms Greedy-min by 0.18% in term of contamination degree. From the tendency of the graph, when the iteration number is large enough, SA-min can converge to the optimum.

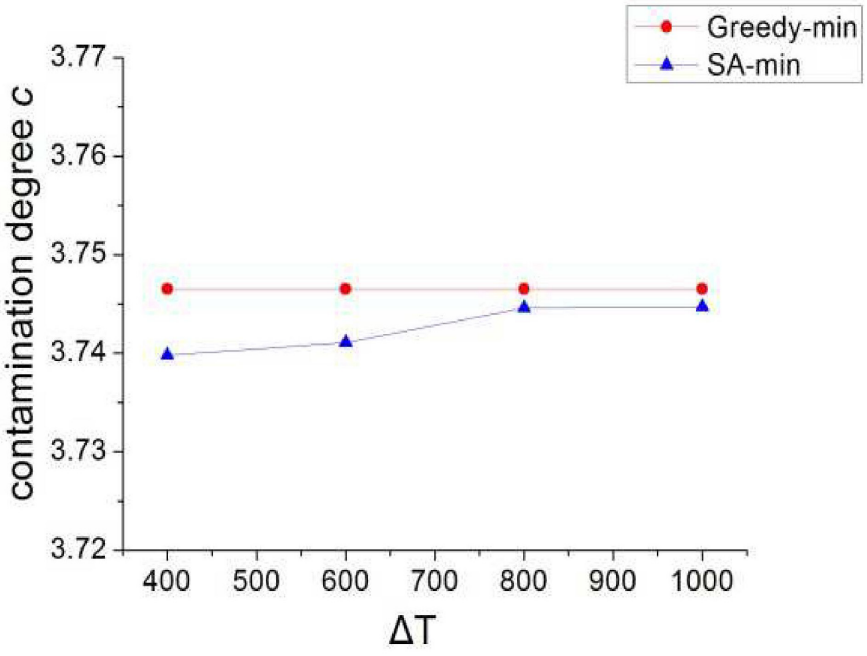

Figure 4. Contamination degree of SA-min and Greedy-min when $p = 0.3$ and $k = 50$ on Amazon-500

6.4 Varying the Number of Nodes

We compare the contamination degree when the number of nodes increases with $p = 0.3$, $k = 50$. From Figure 5(a), SA-min can achieve the matching result with Greedy-min in contamination degree. In Figure 5(b), when n is 10000 and m is 41760, the running time of SA-min and Greedy-min is about 30 minutes and 12 hours respectively. Thus SA-min runs 24 times faster than Greedy-min. We can see that SA-min is more scalable to large graph than Greedy-min.

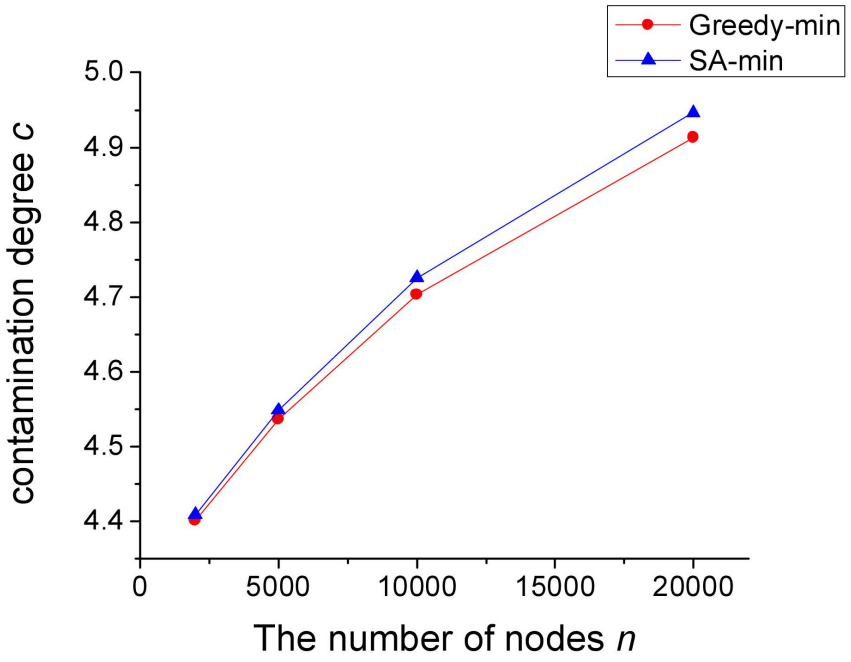

Figure 5 (a). Contamination degree of SA-min and Greedy-min when $p = 0.3$, $k = 50$ on different number of nodes of Amazon

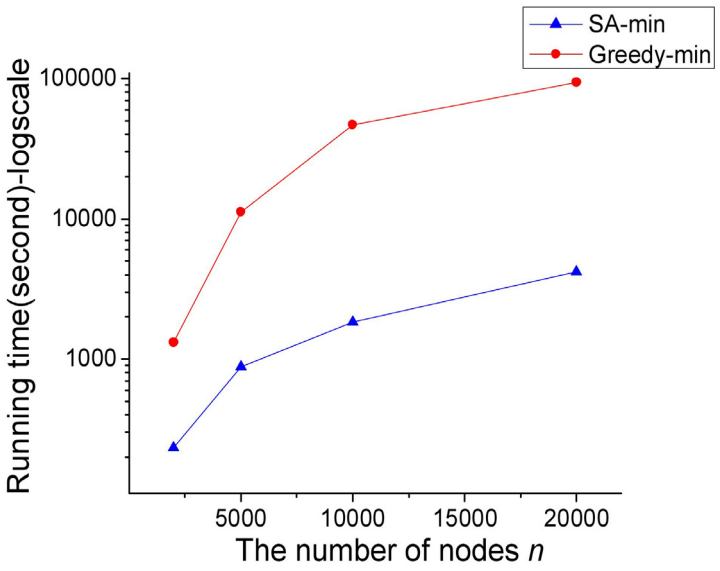

Figure 5 (b). Running time of SA-min and Greedy-min when $p = 0.3$, $k = 50$ on different number of nodes of Amazon

7. Conclusion and Future Work

In this paper, we propose a new efficient algorithm SA-min for contamination minimization problem. In order to improve the efficiency of SA-min algorithm, we design another algorithm ML_CS. Our algorithm both reduces the running time and the contamination degree compared with existing algorithms. Our experimental results show that SA-min can achieve a better performance both in efficiency and effectiveness. Especially when the number of edges becomes large, SA-min algorithm can run much faster than the Greedy-min by several orders of magnitude.

There are several promising directions for future work. Firstly, we can use the real data to obtain the propagation probability on the edge. Secondly, we want to obtain a good parameter for SA-min algorithm, such as T_0, q, ΔT, T_f in order to achieve a better performance.

Acknowledgments

This work was supported in part by the Scientific Research Fund of Heilongjiang Provincial Education Department (12531476), and the Research Foundation of Harbin for Youth Innovative Talents (2012RFQXG096).

References

[1] Domingos, P., Richardson, M. (2001). Mining the network value of customers. *In:* Proceedings of the 7th *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, p. 57–66, 2001.

[2] Albert, R., Jeong, H., Barabási, A. L. (2000). Error and attack tolerance of complex networks, *Nature*, 406 (6794) 378-382.

[3] Callaway, D. S, Newman, M. E. J., Strogatz, S. H. (2000). Network robustness and fragility: *Percolation on Random Graphs, Physical Review Letters*, 85 (25) 5468.

[4] Kimura, M., Saito, K., Motoda, H. (2008). Minimizing the spread of contamination by blocking links in a network, *In*: Proceedings of the 23rd *National conference on Artificial intelligence*. 2008: 1175-1180.

[5] Kempe, D., Kleinberg, J. M., Tardos, É. (2003). Maximizing the spread of influence through a social network. *In*: Proceedings of the 9th *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, p 137–146.

[6] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Van Briesen, J., Glance, N. S. (2007). Cost-effective outbreak detection in networks. *In*: Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, p. 420–429, 2007.

[7] Chen, W., Wang, Y., Yang, S. (2009). Efficient influence maximization in social networks. *In*: Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

[8] Jiang, Q., Song, G., Cong, G., Wang, Y., Si, W., Xie, K. (2011). Simulated annealing based influence maximization in social networks, *In*: AAAI.

[9] Tang, Y., Xiao, X., Shi, Y. (2014). Influence Maximization Near-Optimal Time Complexity Meets Practical Efficiency. *In*:Proceedings of the the 2015 ACM SIGMOD/PODS Conference. 796-805

[10] Lin, S., Chen. M. (2015). A Learning-based Framework to Handle Multi-round Multi-party Influence Maximization on Social Networks. *In*: Proceedingsg of the 21th ACM SIGKDD *Conference on Knowledge Discovery and DataMining,* 410–419.

[11] Mitra, D., Romeo, F., Vincentelli, A. S. (1985). Convergence and finite-time behavior of simulated annealing. *In*: Proceedings of 24th *Conference on Decision and Control*, 761 – 767.

[12] Leskovec, J., Adamic, L., Adamic, B. (2007). The Dynamics of Viral Marketing. *ACM Transactions on the Web* (ACM TWEB), 1(1).

[13] Richardson, M., Agrawal, R., Domingos. P. (2003). Trust Management for the Semantic Web. ISWC.