Knowledge-Based Systems 191 (2020) 105265

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Influence maximization based on the realistic independent cascade model☆

Jingyi Ding *, Wenjing Sun, Jianshe Wu, Yuwei Guo

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, China

ARTICLE INFO

Article history:
Received 12 July 2019
Received in revised form 14 September 2019
Accepted 23 November 2019
Available online 30 November 2019

Keywords:
Diffusion model
Influence maximization
Social networks
Seeding algorithm

ABSTRACT

In order to propagate information through the social network, how to find a seed set that can affect the maximum number of users is named as influence maximization problem. A lot of works have been done on this problem, mainly including two aspects: establishing a reasonable information diffusion model and putting forward the appropriate seeding strategy. However, there are few models in the existing ones that consider the acceptance probability of candidate seed nodes in social networks. So in this paper, we consider and solve this problem by introducing a more realistic model, which is the proposed Realistic Independent Cascade (RIC) model. Based on the RIC model, many state-of-the-art seeding algorithms perform not so well because there is no mechanism on dealing with the acceptance probability. So based on the RIC model, we propose a new seeding strategy which is called R-greedy. Furthermore, M-greedy algorithm is proposed to reduce the time complexity of R-greedy. Then, D-greedy algorithm which not only increased the performance but also reduced the time complexity of R-greedy is proposed by combining the advantages of R-greedy and M-greedy. Experiments on the real-world networks and synthetic networks demonstrate that the proposed R-greedy, M-greedy and D-greedy algorithms outperforms state-of-the-art algorithms.

1. Introduction

Social influence plays an important role in social life. When it comes to listening to a song, choosing a restaurant or buying a real estate, people's choices and decisions are often influenced by family members, colleagues, friends and a wider public trend. Understanding the generation and propagation mode of the influence is helpful for understanding the human behavior, by which we can predict the behavior of people and provide reliable suggestion for the government, institutions, enterprises and other departments. The process of influence diffusion in social networks has been studied in many fields, such as epidemiology [1], social median [2], economy and so on. It has been proved that the research on influence diffusion are of great use in many practical issues, such as marketing strategy [3,4], human behavior analysis [5] and rumor diffusion [6]. So, how to maximize the spread of influence through the social networks? As an example, in order to promote new products, a company would like to offer free samples to a set of initial users who will potentially introduce the new product to their friends (called seed set). Due to expense cost, only a limited number of samples are available and thus we have a budget of the seed set. A natural problem is that how to select a good set of seed users that is able to maximize the number of customers who finally adopt the target product. This problem is named as influence maximization problem, first proposed in Ref. [7]. A lot of works have been done on this problem to put forward an appropriate seeding algorithm on a certain information diffusion model.

In order to formulate the diffusion process, a number of models have been studied during the last decade. Linear threshold (LT) model and independent cascade (IC) model proposed by Kempe et al. in Ref. [8] are two basic operational models. Based on these two basic models, a large number of improved models have been proposed. By extending the classic IC model, dynamic independent cascade (DIC) model is proposed in [9], which is able to better capture the dynamic characteristics of real social networks.

Although a lot of diffusion models have been proposed, some characteristics in the real-world diffusion process are ignored. In the example of promoting new products, in order to advertise new products, a company need to choose a set of initial users and provide samples to them. For the classic IC model, a user is guaranteed to be a seed after selected, that is everyone is

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105265.

* Corresponding author.
E-mail address: jyding@xidian.edu.cn (J. Ding).

https://doi.org/10.1016/j.knosys.2019.105265
0950-7051/

willing to accept the samples and promote to their friends. But it is not realistic, because not everyone is willing to accept the samples and promote to their friends. Besides, in the process of information propagation, even though user u accepts the product and introduces it to user v, whether v will adopt it or not is unknown. Considering above two situations, we set a probability value for each user, called acceptance probability, reflecting the probability with which the user will accept the product. The acceptance probability has two meanings: (1) in the seeding process, when a node is selected according to certain criteria, the acceptance probability means whether it will become a seed; (2) in information propagating process, the acceptance probability means whether a node will accept the information propagated from its neighbors.

Due to individual differences, the acceptance probabilities of different users are different. Furthermore, the propagation probabilities of products are different among users. For example, considering that the promoting product is a computer game, which may have a greater effect on boys than on girls, so the propagation probabilities of products are different among users, which reflects the spread of the same information is different among different groups. Taking the above characteristics into account, we propose a more realistic model, named realistic independent cascade (RIC) model. In RIC model, different nodes have different acceptance probabilities which follows a certain distribution, and propagation probabilities between two nodes are different which also follows a certain distribution.

Based on the RIC model, we further considered how to design a seeding strategy that can help us find effective seeds. Due to the acceptance probability in RIC model, the user may not satisfy the sample and reject to be a seed, thus it cannot effect other nodes. So in seeding strategies, we must consider both its influence and its acceptance probability. Suppose that there are two users, an influential user with smaller acceptance probability and a less influential user with higher acceptance probability, it is difficult to determine which one to be chosen as a seed. So, we should not determine a seed just according to its influence, acceptance probability of nodes should be considered as well. In this paper, we firstly propose R-greedy algorithm, which considers both the influence and the acceptance probability of nodes. Then, M-greedy algorithm is proposed to reduce the time consumption of R-greedy. And D-greedy is proposed to increase the performance of R-greedy further. Three algorithms will be introduced in Sections 4–6.

Our main contributions are summarized as follows:

(1) We propose the realistic independent cascade (RIC) model that is better suited to the real-world network.
(2) Based on RIC model, we propose a new seeds selection algorithm named R-greedy.
(3) Inspired by the observation of simulations, we further propose a heuristic seeding strategy based on R-greedy, named M-greedy, which reduce the time consumption of R-greedy. Combining the advantages of these two seeding strategies, D-greedy is proposed which outperforms R-greedy and reduce time consumption of R-greedy.
(4) Experiments are conducted on real-world networks and synthetic networks to prove the superiority of the proposed algorithms to the existing seeding algorithms based on RIC model.

The rest of this paper is structured as follows. In Section 2, introduce the related works on the influence maximization problem. Section 3, detailed description of the RIC model is given. Section 4 introduces R-greedy algorithm based on RIC model, and M-greedy, D-greedy are introduced in Section 5 and Section 6 respectively. Section 7 is the experiments that show the superiority of our three algorithms to the existing seeding approaches on RIC model. We conclude this paper in Section 8.

2. Related works

The influence maximization problem requires, for a parameter B (budget of the seed set), to find a seed set which can maximize the final influence spread. Initially, the nodes in the seed set are active and the others are inactive. In each step, inactive nodes can become active through the influence of its active neighbors. The final influence spread caused by seed set means the finally total number of active nodes in the network.

Many related works have been done on seeding algorithms. According to a survey about influence maximization on social networks [10], the influence maximization algorithms are classified into three categories: the simulation-based approaches, the proxy-based approaches and the sketch-based approaches. This taxonomy is based on how an algorithm overcomes the NP hardness of evaluating the influence function.

For simulation-based algorithms, a classical algorithm is greedy algorithm which proposed by Kempe et al. [6]. Each time, the node with the greatest marginal profit is selected to seed set. This algorithm has an (1-1/e)-approximation, but the time consumption is very large, especially for large-scale networks. The CELF (Cost-effective Lazy-forward) framework, proposed by J. Leskovec et al. [11], reduces the time complexity of greedy algorithm by utilizing the submodular property of influence function. CELF++ [12] further improves the time complexity of CELF by avoiding unnecessary Monte Carlo simulations. Y. Wang et al. [13] proposed a community-based greedy algorithm (CGA), which divides the whole network into communities and selects seeds according to the influence of each node in its community. And many efficient heuristic influence maximization algorithms have been studied in many works, e.g. [14–16].

For proxy-based algorithms, degree, PageRank [17], distance centrality [18] and some other ranking proxies are used to select seeds. Degree Discount [19] is an algorithm based on degree algorithm, which the influence of the neighbors of node v are discounted when node v is selected into seed set. Liu et al. [20] proposed Group PR to extend PageRank of one node to a set of nodes with a discount for the influence overlap between different nodes. Jung et al. proposed IRIE [21] which the influence of nodes is estimated by solving n linear equations with n variables.

For sketch-based algorithms, influence function is estimated on sketches. S. Cheng et al. proposed the Static Greedy algorithm [22] in 2013 to construct sketches and use them to calculate all influence. Borges et al. [23] proposed the reverse reachable sketch (RR-SKETCH) method for the first time, by constructing random reverse reachable (RR) sets, a seed set is selected to cover the maximum number of RR set. Tang et al. proposed TIM\TIM+ [24] and IMM [25] to give a better analysis on how many RR sets are needed to ensure theoretical bound. Wang et al. Proposed lazy sampling technique (BKRIS) [26], which can greatly speed up IMM.

However, these algorithms cannot be directly used for RIC model, because when a node is selected to be a seed, whether it will be activated successfully is not determined. If it is not activated, what should we do next? So in this paper, we will focus on seeding strategy with acceptance probability.

3. Realistic Independent Cascade (RIC) model

The social network can be described as a graph where the nodes represent the individuals and the edges between the nodes represent the social relationship. In order to spread an information or promote a new product through a social network, a seed set is chosen to be activated by offering free samples or giving payments to trigger the spread of influence.

By extending the IC model, we proposed RIC model. Initially, all the nodes are inactive, and each node in the social graph only

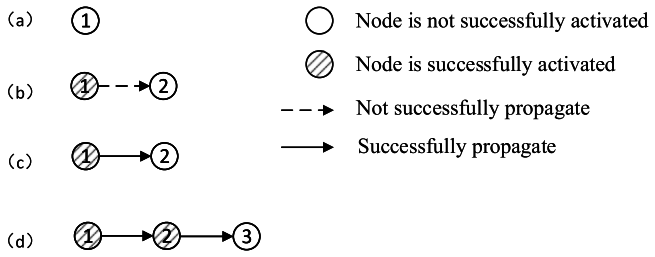

Fig. 1. An illustration of information diffusion process.

has two states: active or inactive. A node can become active in two ways: (1) chosen as a seed; (2) activated by its newly active neighbors.

Considering the acceptance of the initial user and the acceptance of the node during the propagation process, we set a probability value for each user, called acceptance probability. So in the RIC model, each node has an acceptance probability p_v, where $p_v \in [0, 1]$ is a random variable following a certain distribution denoted by f_v. Therefore node v is willing to accept the free samples, which means successfully activated, with acceptance probability p_v when it is selected as a seed and provided free samples; In addition, when the information is successfully propagated to node v from its neighbor, node v accepts it with acceptance probability p_v. Besides, an active node u is given a single chance to propagate the information to its inactive out-bound neighbor w via edge $\langle u, w\rangle$ with a propagation probability $p_{u,w}$, which means node u will propagate information to node w with probability $p_{u,w}$. We set $p_{u,w} \in [0, 1]$ a random variable following a certain distribution denoted by f_e, reflecting the spread of the same information is different among different groups. Due to the complexity of real-network, we cannot capture the accurate distribution of the acceptance probability and propagation probability, or maybe it can be a research content in the future. So we use many different distributions to simulate various situations that may appear in networks, such as Gaussian Distribution, Power-Law Distribution, Exponential Distribution and Uniform Distribution.

The influence propagation process unfolds in discrete time steps according to the following rule. When a node u first becomes active in time t, it has a single chance to propagate information to each currently inactive out-bound neighbor w with the propagation probability $p_{u,w}$. If w has multiple newly-active neighbors, their influence propagation can be sequenced in an arbitrary order. Once the information successfully propagates from node u to w, and node w is willing to accept it, we say that node w is successfully activated by node u and becomes active, then it will influence its inactive out-bound neighbors in time $t + 1$. Note that u cannot influence any nodes in subsequent time steps whether it successfully activates w or not. The above process terminates until no more activation is possible.

Fig. 1 is an example of information diffusion process. In (a), node 1 is not successfully activated as a seed. In (b), node 1 is successfully activated as a seed but the information is not successfully propagated. In (c), node 1 is successfully activated as a seed and the information is successfully propagated to node 2, but node 2 is not willing to accept it. In (d), node 1 is successfully activated as a seed, then the information is successfully propagated to node 2 and node 2 is willing to accept it, then the information is successfully propagated to node 3 but node 3 is not willing to accept it.

In conclusion, the RIC model can be formulated as $G = (V, E, f_v, f_e)$, where V is the set of all the nodes, E is the set of edges, f_v is the distribution of acceptance probability p_v and f_e is the distribution of propagation probability $p_{u,w}$. We denote the number of the nodes in V by n. Due to the limited number of free samples that are used to activate the initial seed set, there is a budget B for the seed set which satisfies $B \leq n$. Besides, due to the acceptance probability $p_v \in [0, 1]$, which means the samples may not satisfy the users, then we are failed to activate a seed and a sample is lost; If the sample satisfies the user, a seed is successfully activated and a sample is lost, so the size of final seed set is less than or equal to B. In Table 1, there are some frequently used notations in this paper.

Table 1
Notations.

Symbol	Definition
G	RIC network
n	Number of nodes in G
m	Number of edges in G
C	Candidate set
A	Seed set
B	Budget of seed set
k	Size of candidate set
$p_{u,w}$	The propagation probability from u to w
p_v	The acceptance probability of node v
f_v	The distribution of acceptance probability
f_e	The distribution of propagation probability
T	The criteria that used in choosing a seed set from candidate set
$\delta(\cdot)$	An influence function
R_s	Number of snapshots produced from network G
R_m	Times of Monte Carlo simulation

4. R-greedy algorithm introduction

In this section, we mainly introduce the realization of the seeding algorithm R-greedy based on the RIC model. The main idea of R-greedy is to select the seed set from the candidate set according to a certain criterion considering both the influence and the acceptance probability of the candidate nodes. The algorithm mainly includes three steps. Step 1, choosing a candidate seed set C according to the influence of nodes; Step 2, giving the criteria, denoted by T, for choosing a seed set from the candidate seed set; Step 3, selecting the seed set A from the candidate set C. The details of these three steps are described as follows.

Step 1: In this step, we choose a candidate seed set C according to the influence of nodes. The influence here means the number of active nodes which can active other nodes when it is activated. So in this step, a node is surely activated when calculating its influence, and the acceptance probability works only in propagating process.

The candidate set C of size k is selected before the diffusion process, where k is larger than the budget of seed set A ($k > B$) and $k \in K$,we have

$$K = \{k_0, k_1, k_2, k_3, \ldots, k_{l-1}, k_l\}, k_i = 10i, \\ i = 0, 1, 2, 3 \ldots I - 1, k_l = n, \tag{1}$$

where n is the number of the nodes in the social networks. If $k_{i-1} < B \leq k_i$ $(i = 1, 2, \ldots, I)$, $k = k_i$.

We define $\delta(C)$ as the influence spread of the candidate set C, initially only the nodes in C are active, after the information diffusion process, the expected number of all the active nodes at the end is defined as $\delta(C)$. The final candidate set C we selected satisfies $\delta(C) \geq \delta(C')$, where C' is any other candidate set of size k. Following the notations in Ref. [27], the concept of realization is introduced to describe the calculation of $\delta(C)$.

Definition 1. A realization of $G = (V, E, f_v, f_e)$ is a mapping from edges to states {live, not live} and a mapping from nodes to states {active, not active}. In G, each edge is mapped to two states: live or not live. If information is successfully propagated from u to w

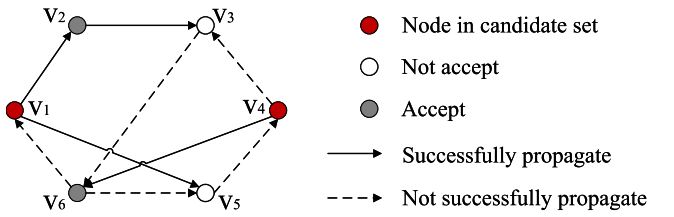

Fig. 2. An example of realization x. The solid lines are the live edge, and the dotted lines are the not-live edges.

via edge $\langle u, w\rangle$, then the edge $\langle u, w\rangle$ is live; if information is not successfully propagated from u to w, then the edge $\langle u, w\rangle$ is not live. Besides, if node v accepts the information, node v is active; if node v does not accept the information, node v is not active. A network can have many different combinations of states of edges and nodes.

Let x be a realization and $\delta_x(C)$ be the expected number of active nodes in the realization x under the set C, therefore,

$$\delta(C) = \sum_{\text{realization x}} \text{Prob(x)} \cdot \delta_x(C), \tag{2}$$

where Prob(x) is the probability of realization x. So the expected number of active nodes is just the weighted average over all realization [8].

For example, given a realization x in Fig. 2, suppose that the candidate set $C = \{v_1, v_4\}$. In this realization x, the final active nodes under the set C are v_1, v_2, v_4, v_6, so $\delta_x(C) = 4$.

It is a NP-hard problem to calculate the real value of Eq. (2) [8], so we can employ the Monte Carlo simulation to obtain an accurate estimation. Given a sufficiently large positive integer R_m, the propagation process of information on the network is simulated R_m times, by averaging the number of active nodes in the R_m simulations, estimation of $\delta(C)$ is obtained.

How to determine a candidate set C that satisfies $\delta(C) \geq \delta(C')$ where C' is any other candidate set? Following the greedy algorithm in Ref. [8] described in Section 2, we select candidate set step by step, in each step, a node which is able to maximize the marginal profit is selected to set C. Marginal profit of node v is formulated as Eq. (3).

$$H_v = \delta(C \cup v) - \delta(C), \tag{3}$$

where H_v denotes the marginal profit of node v. Then a node satisfies Eq. (4) is selected to set C.

$$w_{i+1} = \arg\max_{v \in V \setminus C_i} \{\delta(C_i \cup v) - \delta(C_i)\}, i = 0, 1, \ldots, k-1, \tag{4}$$

where w_i is the ith candidate node, set $C_0 = \emptyset$, $\delta(C_0) = 0$ and $C_i = \{w_1, w_2, \ldots, w_i\}(i = 1, 2, \ldots, k)$ is the candidate set determined after ith step.

The influence spread $\delta(\cdot)$ is estimated by Monte Carlo simulation, so CELF technique is used here, which leads to far fewer evaluations. The pseudo code of determining the candidate set with CELF technique is as following.

Function: candidate set

Input: $G = (V, E, f_v, f_e)$, k (size of the candidate set C)

1: Initialize $C=\emptyset$
2: for each v in V
3: $H_v = +\infty$
4: end for
5: while($|C|<k$)
6: for each v in $V\backslash C$
7: $h_v = false$
8: end for
9: while(true)
10: $v^* = \arg\max_{v \in V \setminus C} \{\delta(C \cup v) - \delta(C)\}$
11: if($h_{v^*} == true$)
12: $C \leftarrow C \cup v^*$
13: break
14: else
15: $H_{v^*} = \delta(C \cup v^*) - \delta(C)$
16: $h_{v^*} == true$
17: end if
18: output C

Line 2–4, initialize the marginal profit of each node v in V to $+\infty$. Let $h_v \in \{true, false\}$ be the flag whether the marginal profit of a node v is updated or not. We select a node which is able to maximize the marginal profit in line 10, if it has been updated, put it to C, otherwise, update its marginal profit, which is the CELF technique.

Step 2: In this step, we consider the acceptance probability of the candidate nodes. Given the budget B of the seed set, we can choose seed set from the candidate set C according to the criteria, named T, formulated as

$$T_{w_i} = H_{w_i} \cdot p_{w_i}, w_i \in C \tag{5}$$

where w_i is the ith candidate node, H_{w_i} is the marginal profit of w_i and p_{w_i} is the acceptance probability of w_i. Calculate the value of T_{w_i} for each candidate node and the larger value of T_{w_i} means it can be chosen to seed set A with a higher probability.

The pseudo code is as following.

Function: criteria

Input: candidate set C, acceptance probability p_{w_i} of w_i ($w_i \in C$), k (size of C)

1: for i=1 to k do
2: calculate $T_{w_i} = H_{w_i} \cdot p_{w_i}$
3: end for
4: output T

Step 3: We select seeds from set C step by step. Firstly, we select the node with the largest T value as a seed, if it accepts, this node is successfully selected into seed set; if it does not accept, then try to activate it again in next selection step, until it is successfully activated and turn to a seed, which means a node in candidate set can be chosen many times until it is successfully selected into seed set. Then we select the node with the second largest T value as the next seed and try to activate it. This seeding process continues until the budget B of the seed set is used up.

The pseudo code of step 3 is as following: function seed set.

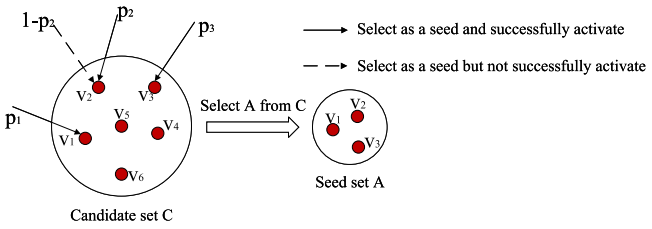

Fig. 3. Select seed set A from candidate set C. Denote T value of each node v_i by $T_i(i = 1, 2, \ldots, 6)$. Given that $T_1 > T_2 > T_3 > T_4 > T_5 > T_6$, and budget B of seed set is 4.

Function: seed set

Input: candidate set C, value of criteria T of candidate node, acceptance probability $p_v(v \in C)$, budget B

1: $A=\emptyset$, $i=0$
2: while($i<B$)
3: $v^* = \arg\max_{v \in C \setminus A} T_v$
4: while($i<B$)
5: activate v^* with acceptance probability p_{v^*}
6: $i=i+1$
7: if v^* is not successfully activated
8: continue
9: else
10: $A \leftarrow A \cup v^*$
11: break
12: return A

Line 4–11 means that if a node is successfully activated as a seed, it will be added to seed set A, or it will be activated more than one time. Thus the final size of A is not larger than B.

Here is an example for selecting seeds. In Fig. 3, we select seed set A from C, and the criteria of candidate nodes calculated by Eq. (5) is ordered: $T_1 > T_2 > T_3 > T_4 > T_5 > T_6$. Node v_1 is the first to be chosen from C and it is successfully activated, then v_2 is chosen, but it is not successfully activated at the first time, so it is chosen again and luckily it is successfully activated this time. The final seed set A is $\{v_1, v_2, v_3\}$ and its size is not larger than budget B.

So, through the three steps above, seed set A is determined.

5. M-greedy algorithm

In this section, we present M-greedy algorithm based on the R-greedy algorithm which is introduced in Section 3. In order to calculate influence spread of a node with Monte Carlo simulation, the whole network need to be traversed, which is very time consuming especially in the network with large number of nodes. To reduce the time consumption, a simple idea is to remove some nodes in the network, which means that Monte Carlo simulation can traverse fewer nodes so that the time consumption is reduced. So what kinds of nodes can be removed?

An important observation, shown latter in Section 6, is that there is a significant difference in the influence spread of nodes. That is to say, fewer nodes have greater influence and most nodes have less influence, which is also the characteristic of the real network. Based on this observation, we design the M-greedy algorithm, which reduced the number of nodes that Monte Carlo simulation need to traverse.

M- greedy: Let $\delta(v)$ be the influence of node v, that is the final active nodes which node v can activate through the network, and $\frac{1}{n}\sum_{v \in V} \delta(v)$ is the mean of influence of all the nodes, where n is the number of nodes in the network.

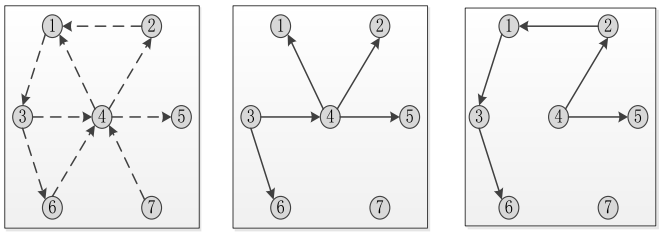

Fig. 4. Get snapshots from origin graph. Each edge is sampled with its associated propagation probability.

Step1: We use Monte Carlo simulation to obtain the influence of each node which expressed as $\delta(v)$, and calculate the value of $\frac{1}{n}\sum_{v \in V} \delta(v)$.

Step2: Nodes that satisfies the condition of $\delta(v) < \frac{1}{n}\sum_{v \in V} \delta(v)$ are removed.

Step3: R-greedy is run to get the seed set.

With the smaller set of nodes, the time consumption in the Monte Carlo simulation process can be greatly reduced. As shown later in Section 5, the performance of M-greedy is close to that of R-greedy, but time consumed is greatly reduced.

6. D-greedy algorithm

For the influence maximization problem, a critical problem which needed to solve is the estimation of value of the influence spread $\delta(\cdot)$. This is a NP hard problem. As discussed above, we use Monte Carlo simulation to obtain the accurate estimate of $\delta(\cdot)$ in R-greedy and M-greedy algorithms. Here, we use another way of thinking snapshots to estimate the value of $\delta(\cdot)$ which is used in Static Greedy [28]. Snapshots G^s are sampled from social networks G. Each edge $\langle u, w\rangle$ in G is sampled with its associated propagation probability $p_{u,w}$ to obtain a snapshot G_i^s.

As shown in Fig. 4, snapshot1 and snapshot2 are obtained by sampling each edge $\langle u, w\rangle$ in G. In snapshot1, edge $\langle 4, 1\rangle$, $\langle 4, 2\rangle$, $\langle 4, 5\rangle$, $\langle 3, 4\rangle$, $\langle 3, 6\rangle$ are remained. And in snapshot2, edge $\langle 2, 1\rangle$, $\langle 1, 3\rangle$, $\langle 3, 6\rangle$, $\langle 4, 2\rangle$, $\langle 4, 5\rangle$ are remained.

Then on each snapshot G_i^s, the influence spread of set A is calculated, which is the number of nodes reachable from A. Then $\delta(A)$ can be obtained by averaging over all the snapshots. Given the social network G, and the number of snapshots required is R_s. The method of producing snapshots G^s from G is formally described as following, in function snapshot.

Function: snapshot

Input: $G = (V, E, f_v, f_e)$, R_s

1: for i = 1 to R_s do
2: for each edge <u, w> in edge set E
3: remove edge <u, w> from G with probability $1 - p_{u,w}$
4: end for
5: get G_i^s
6: end for
7: output G^s

On network G, the number of active nodes influenced by seed set A denoted by $\delta(A)$, which can be calculated by Eq. (6).

$$\delta(A) = \frac{1}{R_s}\sum_{i \in R_s} Ran(G_i^s, A) \tag{6}$$

where R_s is the number of snapshots, and $Ran(G_i^s, A)$ is the number of active nodes influenced by set A on the snapshot $G_i^s (1 \leq$

$i \le R_s$), the value of $\delta(A)$ is obtained by averaging over all the snapshots.

Then, given RIC network $G = (V, E, f_v, f_e)$, snapshots $G_i^s (1 \le i \le R_s)$ and budget B, we need to select a seed set. A node satisfied Eq. (7) can be selected as a seed.

$$w_{i+1} = \underset{v \in V \setminus A_i}{\operatorname{argmax}} \{p_v * H_v\} \tag{7}$$

$$\begin{aligned} H_v &= \delta(A_i \cup v) - \delta(A) \\ &= \frac{1}{R_s} \sum_{j \in R_s} Ran(G_j^s, A_i \cup v) - \frac{1}{R_s} \sum_{j \in R_s} Ran(G_j^s, A_i) \end{aligned} \tag{8}$$

where w_i is the ith seed node, p_v is the acceptance probability of node v and H_v is the marginal profit of node v. Set $A_0 = \emptyset$, $\delta(A_0) = 0$ and $A_i = \{w_1, w_2, \ldots, w_i\}$ is the seed set selected after ith step. After w_i is selected, we try to activate it with its acceptance probability, if it accepts, this node is successfully selected into seed set; if it does not accept, this node is not successfully selected into seed set, then it will be selected again in next selection step. Repeat the process above until budget B is used up. Note that in each selection step, one budget is used. So the size of seed set is not larger than budget B. This algorithm is formally described in algorithm D-greedy.

Algorithm: D-greedy

Input: $G = (V, E, f_v, f_e)$, snapshots $G_i^s (1 \le i \le R_s)$, budget B

1: initialize $A = \emptyset$
2: for i = 1 to B do
3: set $H_v = 0$ for all $v \in V \setminus A$
4: for j=1 to R_s do
5: for each v in $V \setminus A$
6: $H_v += \frac{1}{R_s}(Ran(G_j^s, A \cup v) - Ran(G_j^s, A))$
7: end for
8: end for
9: $w = \arg\max_{v \in V \setminus A} \{p_v * H_v\}$
10: while (i <= B)
11: try to active w
12: if w is successfully activated
13: $A = A \cup \{w\}$
14: i=i+1
15: break
16: else
17: i=i+1
18: end if
19: end for
20: output A

Line 3–9 calculate the value of H_v for each node v in $V \setminus A$ and select the node which satisfies Eq. (7). Here, we can also use CELF technique to reduce time complexity, and in the experiments following, we do use it. In line 10–18, we activate the selected node until it is successfully activated as a seed before budget is used up.

7. Experiment

7.1. Setup

In order to compare our proposed algorithms to the existing approaches, we use two widely used real-world social networks and a LFR benchmark networks.

Wiki-Vote network. It contains the Wikipedia voting data from the inception of Wikipedia [29] and it has about 7115 nodes and 103 689 directed edges, where nodes represent Wikipedia users and a directed edge from node u to node v represents that user u votes on user v.

CA-HepTh network. It is an academic collaboration from co-authorship in physics and it has 9877 nodes and 51 971 edges, for each pair of authors who has a co-authorship, there is a directed edge.

Ca-Condmat network. It is from the e-print arXiv and covers scientific collaborations between authors papers submitted to Condense Matter category. It has about 23 133 nodes and 93 497 undirected edges.

Above three networks can be download from SNAP.[1]

LFR networks. The LFR network is proposed to generate networks which captures the feature of many real world networks [30]. The LFR network has six parameters: (1) the average degree $\langle k \rangle$; (2) the maximum degree $k_{\max}$; (3) the number of nodes n; (4) the exponent for the degree distribution γ; (5) the exponent for the distribution of community size β; and (6) the mixing parameter μ [30].

In the following experiments, we generate LFR networks with different parameters as shown in Table 2.

Table 2
Parameters for different LFR networks.

Network	Parameter					
	n	⟨k⟩	k_{max}	β	γ	μ
LFR100(1)	100	4	10	1	2	0.4
LFR100(2)	100	4	10	1	2	0.3
LFR100(3)	100	4	10	1	2	0.2
LFR100(4)	100	4	10	1	2	0.1
LFR1000(1)	1000	10	25	1	2	0.4
LFR1000(2)	1000	10	25	1	2	0.3
LFR1000(3)	1000	10	25	1	2	0.2
LFR1000(4)	1000	10	25	1	2	0.1
LFR5000(1)	5000	25	50	1	3	0.4
LFR5000(2)	5000	25	50	1	3	0.3
LFR5000(3)	5000	25	50	1	3	0.2
LFR5000(4)	5000	25	50	1	3	0.1

We compare our R-greedy, M-greedy and D-greedy with state-of-the-art algorithms, including CELF-greedy algorithm [11], Static Greedy proposed in Ref. [28] and BKRIS algorithm proposed in Ref. [26] on RIC model.

In RIC model $G = (V, E, f_v, f_e)$, f_v is the distribution of acceptance probability for each node and f_e is the distribution of propagation probability for each edge. In the following experiments on the several different networks, we set different distributions for acceptance probability and propagation probability to simulate various situations that may appear in networks, including Gaussian distribution, power-law distribution, exponential distribution and uniform distribution.

Power-law distribution, formulated as $y = c \cdot x^t$, has two parameters: c and t. Gaussian distribution has two parameters: expected value μ and deviation σ^2. Exponential distribution has one parameter λ.

7.2. Comparison algorithms

(1) R-greedy. This is the algorithm proposed in Section 4. Seed set is selected from candidate set according to the criteria T. For each value of spread function $\delta(\cdot)$, 10 000 Monte Carlo simulations are run to obtain accurate estimation.
(2) M-greedy. This algorithm is proposed in Section 5. Not only reduce the time consumption of R-greedy, but also keep the performance of R-greedy. Similarly, 10 000 Monte Carlo simulations are run to obtain accurate estimation of $\delta(\cdot)$.

[1] http://snap.stanford.edu/data/index.html.

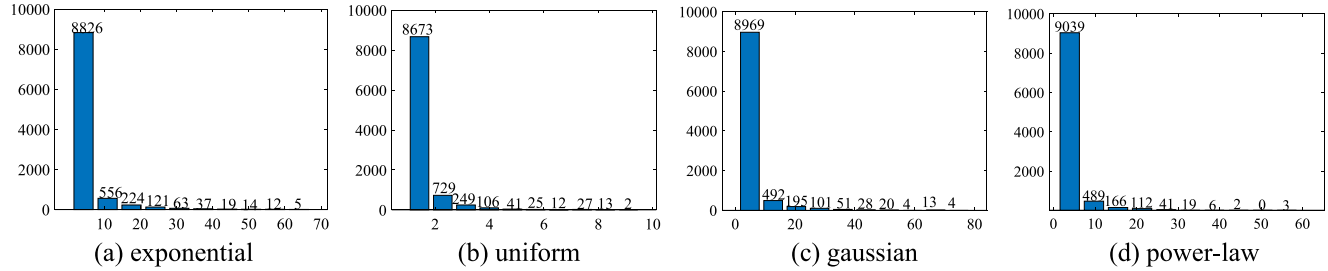

Fig. 5. Histogram of $\delta(v)$ for each node v on CA-HepTh network. (a) f_e is an exponential distribution with a mean of 0.25, and f_v is an exponential distribution with a mean of 0.4. (b) f_e is a uniform distribution on (0.1∼0.3), and f_v is a uniform distribution on (0.1∼0.3). (c) f_e is a gaussian distribution with $\mu = 0.2, \sigma^2 = 0.1$, and f_v is a gaussian distribution with $\mu = 0.3, \sigma^2 = 0.2$. (d) f_e is a power-law distribution with $c = 1$, $t = 4$, and f_v is a power-law distribution with $c = 1$, $t = 2$.

(3) D-greedy. We set $R_s = 100$, that is 100 snapshots are sampled from network G, then $\delta(\cdot)$ can be obtained by averaging over all the snapshots.
(4) CELF-greedy. Given the budget of seed set A, it is selected before information diffusion process, when the diffusion begins, due to the acceptance probability of the seeds, not all the seeds can be successfully activated, so the influence spread of A (*i.e.*, the number of final active nodes) is caused by the successfully activated seeds. For each value of spread function $\delta(\cdot)$, 10 000 Monte Carlo simulations are run to obtain accurate estimation.
(5) Static Greedy. As discussed in Section 6, 100 snapshots are sampled from the given social network G, then we select the seed set on the 100 snapshots before the diffusion process. But because of the nodes in the networks have acceptance probability, a seed is not guaranteed to be activated and has influence spread, that is only the seeds can be successfully activated will have influence spread. CELF technique is used in the following experiments.
(6) BKRIS. In the following experiment, θ reverse reachable sets are required,

$$\theta = \frac{2n((1-1/e)\cdot\alpha+\beta)^2}{OPT\cdot\varepsilon^2}, \tag{9}$$

where OPT is the influence spread of the optimal seed set, ε is the error of Monte Carlo simulation, $\alpha = \sqrt{l\log n + \log 2}$, $\beta = \sqrt{(1-1/e)\cdot(\log_k^n + l\log n + \log 2)}$, n is the number of nodes. We set $l = 1$ and $\varepsilon = 0.2$. Given the θ reverse reachable sets, seed set can be obtained, due to the acceptance probability of the seeds, not all the seeds can be successfully activated, so the influence spread of A is caused by the successfully activated seeds.

All the experiments are done on a computer with 3.20 GHz CPU and 32.0 G memory. And each experiment runs 30 times to obtain the average result in Figs. 6–12.

7.3. Results

As an example, we show the histogram of $\delta(v)$ for each node v on the CA-HepTh network with different setting of acceptance probability f_v and propagation probability f_e in Fig. 5(a)–(d). One can see that there is a significant difference in the influence spread of nodes, that fewer nodes have greater influence and most nodes have less influence which mostly less than the mean of influence of all the nodes.

In Fig. 5(a)–(d), the x-axis and y-axis denote the value of $\delta(v)$ and its statistic, respectively. In (a), the value of $\sum_{v\in V}\delta(v)/N$ is 3.662, and the influence of 77.99% nodes are less than it. In (b), the value of $\sum_{v\in V}\delta(v)/N$ is 1.4327, and the influence of 73.85% nodes are less than it. In (c), the value of $\sum_{v\in V}\delta(v)/N$ is 3.8168, and the influence of 78.58% nodes are less than it. In (d), the value of $\sum_{v\in V}\delta(v)/N$ is 2.7926, and the influence of 78.86% nodes are less than it.

Results on Wiki-Vote, ca-Condmat and LFR networks are similar and we do not show here. In Figs. 6–11, one can see that the performance of M-greedy is close to that of R-greedy, with the reduction of nodes. And Table 4 and Fig. 12 show the time consumption of different comparison algorithms, and compared to R-greedy, M-greedy and D-greedy greatly reduce the time consumption.

We do experiments on CA-HepTh, Wiki-Vote, ca-Condmat and LFR networks, for each network, we set different distributions for f_e and f_v. As shown in Fig. 6, we set four different distributions for LFR100(1), LFR100(2), LFR100(3) and LFR100(4), respectively. D-greedy outperforms the others in all the situations. For LFR100(1), the performance of M-greedy is almost the same as that of R-greedy in Fig. 6(a), (b) and (d), in Fig. 6(c), it is slightly worse than that of R-greedy. For LFR100(2), the performance of M-greedy is close to that of R-greedy in Fig. 6(f)–(h), in Fig. 6(e), it is slightly worse than that of R-greedy. For LFR100(3), the curve of M-greedy is close to that of R-greedy in the four different distributions. For LFR100(4), the curves of M-greedy and R-greedy almost coincide in Fig. 6(m)–(o), and the gap between the two curves is bigger and bigger with the increasing of budget.

Similarly, we set four different distributions for LFR1000(1), LFR1000(2), LFR1000(3) and LFR1000(4), respectively. D-greedy outperforms the others in all the situations. For LFR1000(1), the performance of M-greedy is close to that of R-greedy in Fig. 7(a)–(c), in Fig. 7(d), the gap of the two algorithm appears when budget is larger than 10. For LFR1000(2), in Fig. 7(g), M-greedy return a result with close influence to R-greedy, while in Fig. 7(e), (f) and (h), M-greedy is slightly worse than R-greedy. For LFR1000(3) and LFR1000(4), the curve of M-greedy almost coincides with that of R-greedy in four different distributions, respectively.

As shown in Fig. 8, for LFR5000(1), LFR5000(2), LFR5000(3) and LFR5000(4), we set four different distributions, respectively. D-greedy outperforms the others in all the situations. For M-greedy and R-greedy, in LFR5000(1), as shown in Fig. 8(a), (b) and (c), the gap between the M-greedy and R-greedy is bigger and bigger with the increasing of budget. But the performance of M-greedy is still better than the performance of CELF-greedy, BKRIS and static greedy; In Fig. 8(d), the performance of M-greedy is almost the same as that of R-greedy. In LFR5000(2), as shown in Fig. 8(e), (g) and (h), the performance of M-greedy is slightly worse than that of R-greedy, but it is still much better than that of ELF-greedy, BKRIS and static greedy in Fig. 8(e) and (h), while it is close to static greedy in Fig. 8(g); In Fig. 8(f), the performance of M-greedy is close to that of R-greedy, and it has little advantage over static greedy. In LFR5000(3), as shown in Fig. 8(i)–(l), the performance of M-greedy is slightly worse than that of R-greedy for the different distributions, and it is close to static greedy when we set uniform distribution in Fig. 8(g). In LFR5000(4), the performance of M-greedy is close to R-greedy for the different

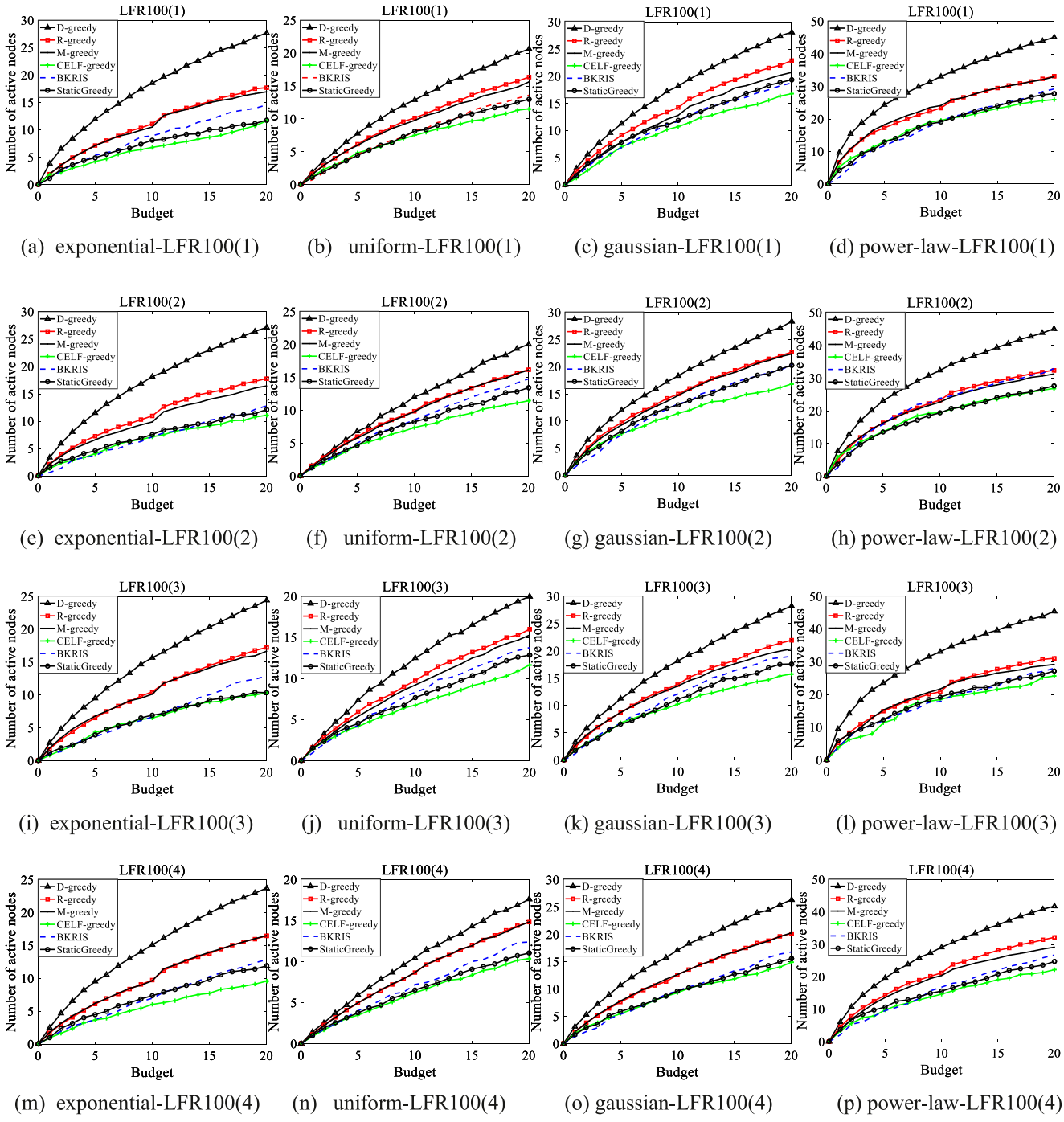

Fig. 6. Comparison of algorithms on the LFR100 networks. *y*-axis and *x*-axis denote the number of active nodes and the budget, respectively. Each title under figures means a certain distributions of f_e and f_v on the given network, for example, in (a), "exponential-LFR100(1)" means experiment on LFR100(1) with exponential distribution of f_e and f_v. Exponential distribution: f_e with a mean of 0.8 and f_v with a mean of 0.5. Uniform distribution: f_e is a uniform distribution on (0.4∼0.6) and f_v is a uniform distribution on (0.1∼0.5). Gaussian distribution: f_e with $\mu = 0.4, \sigma^2 = 0.2$, f_v with $\mu = 0.4, \sigma^2 = 0.2$. Power-law distribution: f_e with $c = 1$, $t = 1$, f_v with $c = 1$, $t = 1$.

distributions, and has little advantage over static greedy for exponential, uniform and Gaussian distribution, while has a bigger advantage over static greedy for power-law distribution.

In a word, for all the 12 LFR networks, we set different distributions for them, the performance of D-greedy outperforms the others in all the situations, and the performance of M-greedy, with a reduction of nodes in networks, is close to or slightly worse than R-greedy.

For the CA-HepTh network, as shown in Fig. 9, the performance of D-greedy is the best among all of the six algorithms for different distributions. M-greedy can return a result with close influence to R-greedy in Fig. 9(a)–(c), with 77.99%, 73.85% and 78.58% nodes reduction in M-greedy, respectively, as shown in Fig. 5(a)–(c). In Fig. 9(d), with 78.86% nodes reduction, M-greedy performs slightly worse than R-greedy, but still better than CELF-greedy, BKRIS and static greedy.

For the Wiki-Vote network, as shown in Fig. 10(a), the performance of D-greedy is the best among all of the six algorithms, and the performance of M-greedy is almost the same as that of R-greedy. In Fig. 10(b), R-greedy and M-greedy are still close

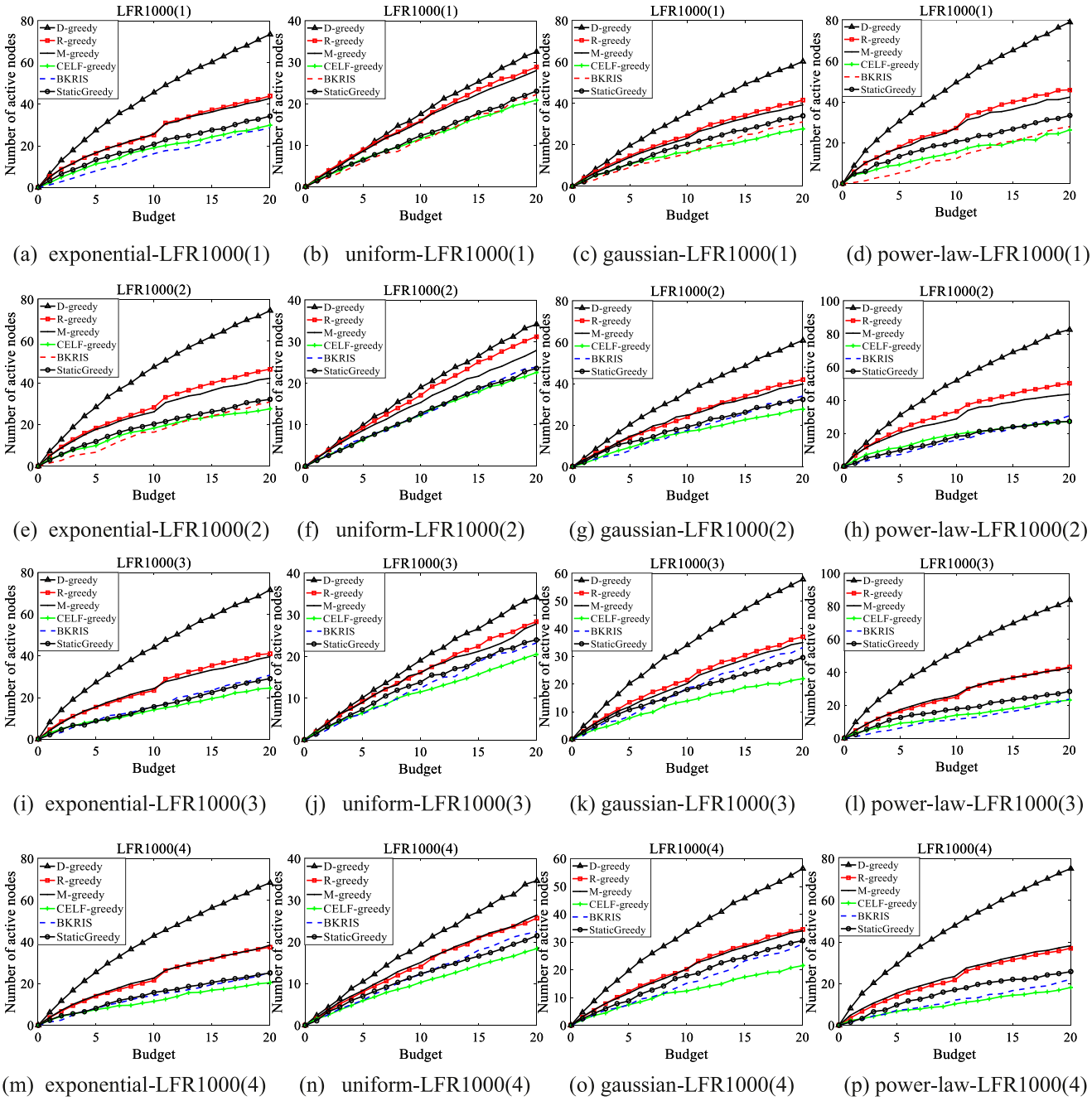

Fig. 7. Comparison of algorithms on the LFR1000 networks. *y*-axis and *x*-axis denote the number of active nodes and the budget, respectively. Each title under figures means a certain distribution of f_e and f_v on the given network, which is the same as that in Fig. 6. Exponential distribution: f_e with a mean of 0.2 and f_v with a mean of 0.5. Uniform distribution: f_e is a uniform distribution on (0.1∼0.4) and f_v is a uniform distribution on (0.1∼0.4). Gaussian distribution: f_e with $\mu = 0.2, \sigma^2 = 0.1$, f_v with $\mu = 0.3, \sigma^2 = 0.2$. Power-law distribution: f_e with $c = 1$, $t = 4$, f_v with $c = 1$, $t = 2$.

in performance, and D-greedy performs almost close to them. In Fig. 10(c), the performance of D-greedy is close to that of R-greedy, and M-greedy are slightly worse than them. In Fig. 10(d), the performance of R-greedy and M-greedy are almost the same, and D-greedy performs much better than them.

For the ca-Condmat network, as shown in Fig. 11, the performance of D-greedy is the best among all of the six algorithms, and the performance of M-greedy is almost the same as that of R-greedy in Fig. 11(b) and (c). In Fig. 11(a), the performance of M-greedy is slightly worse than R-greedy. In Fig. 11(d), the performance of D-greedy performs much better than other algorithms.

In a word, in the three real networks, for different distributions of propagation probability and acceptance probability, D-greedy outperforms R-greedy and M-greedy, the performance of M-greedy, with a reduction of nodes in networks, is close to or slightly worse than that of R-greedy.

7.4. Time complexity of algorithms

Now, we analyze the time complexity of all the comparison algorithms in our experiments.

For clarity, we use R_m to denote the number of Monte Carlo simulations used in greedy, R-greedy and M-greedy algorithms, B is the budget of seed set, k is the size of candidate set, n is the

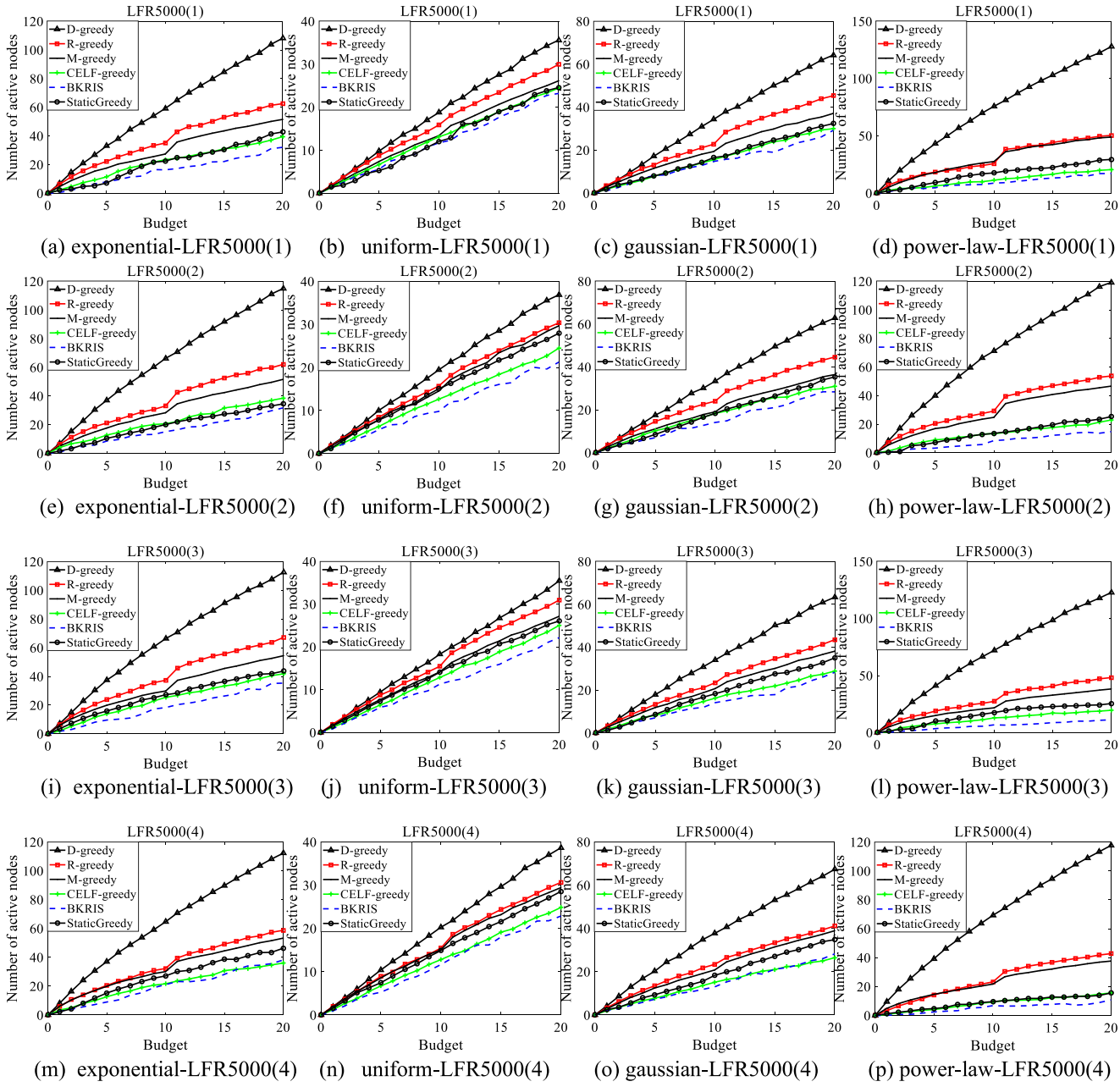

Fig. 8. Comparison of algorithms on the LFR5000 networks. *y*-axis and *x*-axis denote the number of active nodes and the budget, respectively. Each title under figures means a certain distributions of f_e and f_v on the given network, which is the same as that in Fig. 6. Exponential distribution: f_e with a mean of 0.08 and f_v with a mean of 0.5. Uniform distribution: f_e is a uniform distribution on (0.02∼0.08) and f_v is a uniform distribution on (0.2∼0.6). Gaussian distribution: f_e with $\mu = 0.08, \sigma^2 = 0.03$, f_v with $\mu = 0.3, \sigma^2 = 0.2$. Power-law distribution: f_e with $c = 1$, $t = 6$, f_v with $c = 1$, $t = 4$.

number of nodes in network G, and m is the number of edges in network G. For our R-greedy, the time complexity includes two parts: firstly, the time complexity of generating candidate set is $O(knR_m m)$; secondly, it takes $O(B^2)$ time to select seed nodes from candidate set. Thus the total time complexity is $O(knR_m m + B^2)$.

M-greedy eliminates many nodes and edges, so we use m' and n' to denote the number of the remaining edges and the remaining nodes in network G, respectively. The time complexity of M-greedy is $O(kn'R_m m' + B^2)$.

For D-greedy, we use $R_s$to denote the number of snapshots required, m_s is the average number of active edges in the snapshots sampled from the network G, and the definition of m, n, B is the same as above. The time complexity of D-greedy includes two parts: firstly, R_s snapshots are generated from the origin network G, and for each snapshot, m edges $\langle u, w\rangle$ are sampled one by one with probability $p_{u,w}$, so it takes $O(R_s m)$ time; secondly, greedy manner is used to select seeds step by step, which takes $O(BnR_s m_s)$ time. Thus the total time complexity of D-greedy is $O(R_s m + BnR_s m_s)$.

The other state of art comparison algorithms in our experiments are listed in Table 3.

In Table 3, the time complexity of CELF-greedy is $O(BnR_m m)$. The time complexity of Static Greedy is $O(R_s m + BnR_s m_s)$, and the time complexity of BKRIS is $O((m+n)|R|(\log n + \log_k^n)/\varepsilon^2)$, where ε is the error of Monte Carlo simulation.

As an example, given budget B = 10, Table 4 lists the running time of selecting seed set A by using different algorithms on Wiki-Vote network with different distributions setting in Fig. 10.

And Fig. 12 shows this result intuitively. Here different color bars represent the time consumption of different methods. For

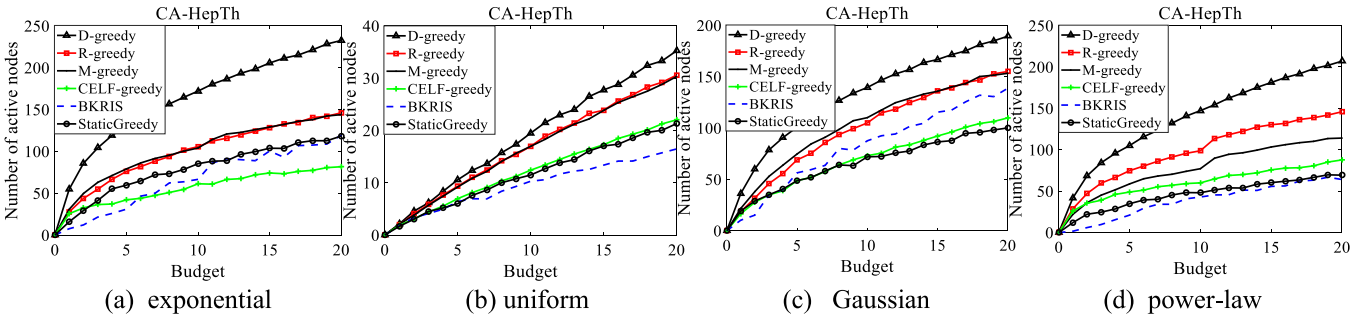

Fig. 9. Comparison of algorithms on the CA-HepTh network. And y-axis and x-axis denote the number of active nodes and the budget, respectively. (a) f_e is an exponential distribution with a mean of 0.25, and f_v is an exponential distribution with a mean of 0.4. (b) f_e is a uniform distribution on (0.1∼0.3), and f_v is a uniform distribution on (0.1∼0.3). (c) f_e is a gaussian distribution with $\mu = 0.2, \sigma^2 = 0.1$, and f_v is a gaussian distribution with $\mu = 0.3, \sigma^2 = 0.2$. (d) f_e is a power-law distribution with $c = 1$, $t = 4$, and f_v is a power-law distribution with $c = 1$, $t = 2$.

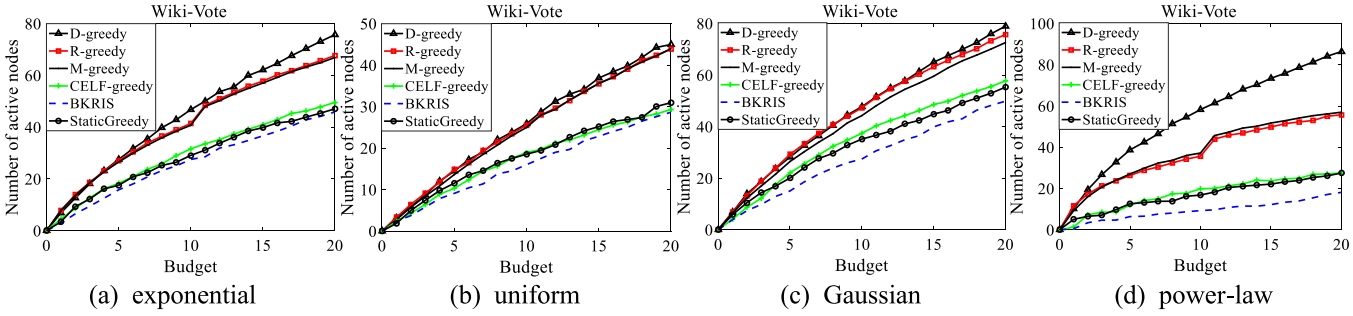

Fig. 10. Comparison of algorithms on the Wiki-Vote network. y-axis and x-axis denote the number of active nodes and the budget, respectively. (a) f_e is an exponential distribution with a mean of 0.02, and f_v is an exponential distribution with a mean of 0.8. (b) f_e is a uniform distribution on (0.02∼0.04), and f_v is a uniform distribution on (0.1∼0.4). (c) f_e is a gaussian distribution with $\mu = 0.02, \sigma^2 = 0.01$, and f_v is a gaussian distribution with $\mu = 0.4, \sigma^2 = 0.2$. (d) f_e is a power-law distribution with $c = 1$, $t = 10$, and f_v is a power-law distribution with $c = 1$, $t = 6$.

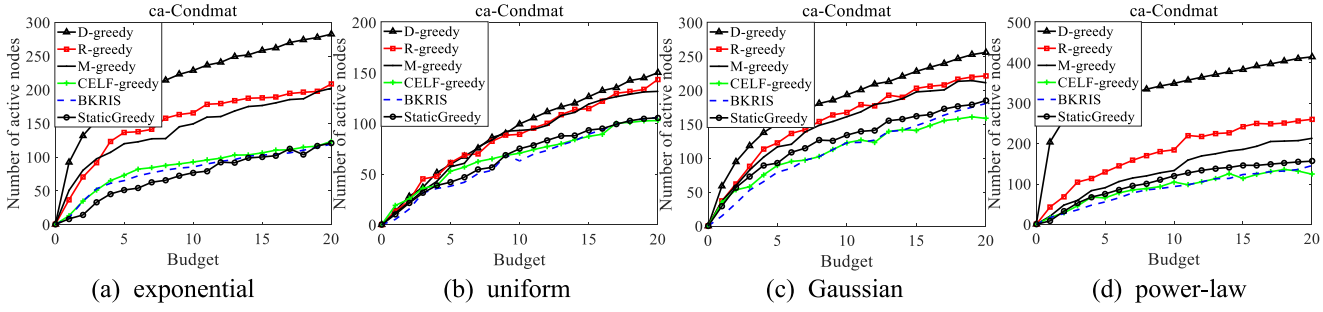

Fig. 11. Comparison of algorithms on the ca-Condmat network. y-axis and x-axis denote the number of active nodes and the budget, respectively. (a) f_e is an exponential distribution with a mean of 0.01, and f_v is an exponential distribution with a mean of 0.5. (b) f_e is a uniform distribution on (0.1∼0.2), and f_v is a uniform distribution on (0.1∼0.3). (c) f_e is a gaussian distribution with $\mu = 0.08, \sigma^2 = 0.05$, and f_v is a gaussian distribution with $\mu = 0.3, \sigma^2 = 0.2$. (d) f_e is a power-law distribution with $c = 1$, $t = 5$, and f_v is a power-law distribution with $c = 1$, $t = 3$.

Table 3
Time complexity of different algorithms.

Algorithm	Time complexity
CELF-greedy	$O(BnR_m m)$
Static greedy	$O(R_s m + BnR_s m_s)$
BKRIS	$O((m+n)\cdot\lvert R\rvert\cdot(\log n + \log_k^n)/\varepsilon^2)$
D-greedy	$O(R_s m + BnR_s m_s)$
R-greedy	$O(knR_m m + B^2)$
M-greedy	$O(kn'R_m m' + B^2)$

Table 4
Running time of selecting seed set, with budget b = 10, for different algorithms on Wiki-Vote network with distributions setting in Fig. 10.

Algorithm	Distribution			
	Exponential	Uniform	Gaussian	Power-law
R-greedy(s)	2553.189	1732.928	2707.759	4375.696
M-greedy(s)	1600.680	710.623	1727.600	2550.457
D-greedy(s)	241.357	300.508	298.761	960.335
CELF-greedy(s)	2954.774	2487.626	3066.014	5921.246
Static greedy(s)	274.118	276.923	265.416	1690.871
BKRIS(s)	1365.223	1367.381	1357.473	1378.168

example, blue bar represents the algorithm of R-greedy, the red one represents the method of M-greedy, the green bar represents the D-greedy algorithm, the purple one represents the CELF-greedy method, the cyan bar represents the StaticGreedy algorithm, and the orange one represents the BKRIS method. The Y-axis represents the running time of the algorithms and the X-axis represents different probability distributions. Here, the details of the distributions are as follows: for an exponential distribution, f_e is an exponential distribution of propagation probability for each edge with a mean of 0.02, and f_v is an

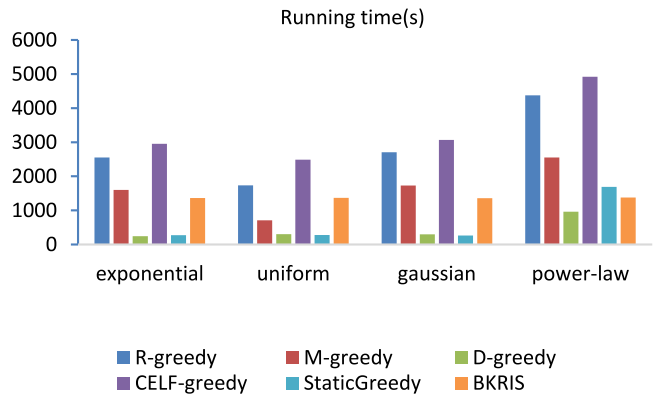

Fig. 12. Running time of selecting seed set A, with budget $B = 10$, by using different algorithms on Wiki-Vote network with distributions used in Fig. 10. here, different color bars represent the running time of different algorithms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exponential distribution of acceptance probability for each node with a mean of 0.8. For a uniform distribution, f_e is a uniform distribution of propagation probability on (0.02∼0.04), and f_v is a uniform distribution of acceptance probability on (0.1∼0.4). For a gaussian distribution, f_e is a gaussian distribution of propagation probability with $\mu = 0.02, \sigma^2 = 0.01$, and f_v is a gaussian distribution of acceptance probability with $\mu = 0.4, \sigma^2 = 0.2$. For a power-law distribution, f_e is a power-law distribution of propagation probability with $c = 1$, $t = 10$, and f_v is a power-law distribution of acceptance probability with $c = 1$, $t = 6$.

From Fig. 12, we can see that among all the different probability distributions, the time consumption of D-greedy algorithm is the least, followed by M-greedy algorithm, and R-greedy algorithm is the most. Moreover, M-greedy reduce the time consumption of R-greedy and it can return a result close to that of R-greedy as shown in Fig. 10. D-greedy performs the best among the three algorithms proposed in this paper, as shown in Figs. 6–11, with less time consumption. In summary, D-greedy algorithm achieves better results with less running time.

8. Conclusion

The work of this paper is how to maximize the spread of influence in social network. We propose RIC model which is able to capture the uncertainness of the diffusion process. In RIC model, when a node is selected as a seed or when the information is propagated to a node, the node is not guaranteed to accept it, thus we set a random value following a certain distribution; besides the propagation probabilities between two nodes are not the same value, which follows a certain distribution reflecting the spread of the same information among different individuals is different. Based on the RIC model, we propose the R-greedy algorithm which select the seed set from the candidate set according to a certain criterion and M-greedy is also proposed which greatly reduce the time consumption in R-greedy. Finally, D-greedy is proposed by combining the advantages of R-greedy and M-greedy, it performs better than the others with less time consumption compared to R-greedy and M-greedy.

Acknowledgments

The work was supported by the China Postdoctoral Science Foundation Funded Project (No.: 2018M643586), the Fundamental Research Funds for the Central Universities of China (No.: XJS18030), the Fundamental Research Funds for the Central Universities of China (No.: JB191904), the National Natural Science Foundation of China (No.: 61902294).

References

[1] L.A. Viana, M.P. Cristelli, D.W. Santos, et al., Influence of epidemiology, immunosuppressive regimens, clinical presentation, and treatment on kidney transplant outcomes of patients diagnosed with tuberculosis: A retrospective cohort analysis, Am. J. Transplant. 19 (5) (2019).
[2] K. Bok, Y. Noh, J. Lim, et al., Hot topic prediction considering influence and expertise in social media, Electron. Commer. Res. (1) (2019) 1–17.
[3] V. Mahajan, E. Muller, F.M. Bass, New product diffusion models in marketing: A review and directions for research, Marketing 54 (1) (1990) 1–26.
[4] A. Sela, D. Goldenberg, I. Ben-Gal, et al., Active viral marketing: Incorporating continuous active seeding efforts into the diffusion model, Expert Syst. Appl. 107 (2018).
[5] R.M. Bond, et al., A 61-million-person experiment in social influence and political mobilization, Nature 489 (7415) (2012) 295–298.
[6] L. Fan, et al., Least cost rumor blocking in social networks, in Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. (ICDCS), 2013, pp. 540–549.
[7] P. Domingos, M. Richardson, Mining the network value of customers, in: Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2001, pp. 57–66.
[8] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2003, pp. 137–146.
[9] G.M. Tong, W.L. Wu, S.J. Tang, D.J. Du, Adaptive influence maximization in dynamic social networks, IEEE Trans. Netw. 25 (1) (2017) 112–125.
[10] Y. Li, J. Fan, Y. Wang, et al., Influence maximization on social graphs: A survey, IEEE Trans. Knowl. Data Eng. PP (99) (2018) 1.
[11] J. Leskovec, et al., Cost-effective outbreak detection in networks, in: Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2007, pp. 420–429.
[12] A. Goyal, W. Lu, L.V. Lakshmanan, Celf++: Optimizing the greedy algorithm for influence maximization in social networks, in: WWW, 2011, pp. 47–48.
[13] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy Algorithm for mining top-k influential nodes in mobile social networks, in: KDD, 2010, pp. 1039–1048.
[14] X. Wang, K. Deng, J. Li, et al., Targeted influence minimization in social networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, Cham, 2018.
[15] M. Gomez-Rodriguez, L. Song, N. Du, et al., Influence estimation and maximization in continuous-time diffusion networks, ACM Trans. Inf. Syst. 34 (2) (2016) 1–33.
[16] X.B. Rui, F.R. Meng, Z.X. Wang, et al., A reversed node ranking approach for influence maximization in social networks, Appl. Intell. (6) (2019) 1–15.
[17] L. Page, S. Brin, R. Motwani, T. Winograd, The Pagerank Citation Ranking: Bringing Order to the Web, Tech. Rep., The Stanford InfoLab, 1999, 1999-66.
[18] L.C. Freeman, Centrality in social networks conceptual clarification, Social Networks 1 (3) (1978–1979) 215–239.
[19] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in KDD, 2009 pp. 199–208.
[20] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, J.X. Yu, Influence maximization over large-scale social networks: A bounded linear approach, in: CIKM, 2014, pp. 171–180.
[21] K. Jung, W. Heo, W. Chen, IRIE: scalable and robust influence maximization in social networks, in: ICDM, 2012, pp. 918–923.
[22] S. Cheng, H. Shen, J. Huang, G. Zhang, X. Cheng, Static greedy: Solving the scalability-accuracy dilemma in influence maximization, in: CIKM, 2013, pp. 509–518.
[23] C. Borgs, M. Brautbar, J. Chayes, B. Lucier, Maximizing social influence in nearly optimal time, in: SODA, 2014, pp. 946–957, revision available at https://arxiv.org/abs/1212.0884.
[24] Y. Tang, X. Xiao, Y. Shi, Influence maximization: Near-optimal time complexity meets practical efficiency, in: SIGMOD, 2014, pp. 75–86.
[25] Y. Tang, Y. Shi, X. Xiao, Influence maximization in near-linear time: A martingale approach, in: SIGMOD, 2015, pp. 1539–1554.
[26] X. Wang, Y. Zhang, W. Zhang, X. Lin, C. Chen, Bring order into the samples: A novel scalable method for influence maximization, IEEE Trans. Knowl. Data Eng. 29 (2) (2017) 243–256.
[27] A. Asadpour, H. Nazerzadeh, A. Saberi, Stochastic submodular maximization, in: Proc. 4th Int. Workshop WINE, 2008, Shanghai, China, pp. 477–489.
[28] S. Cheng, H. Shen, J. Huang, G. Zhang, X. Cheng, Static greedy: Solving the scalability-accuracy dilemma in influence maximization, in: CIKM, 2013, pp. 509–518.
[29] J. Leskovec, Wikipedia vote network. [Online]. Available: http://snap.stanford.edu/data/wiki-Vote.html.
[30] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.