Knowledge-Based Systems 134 (2017) 149–158

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Community-based influence maximization in social networks under a competitive linear threshold model

Arastoo Bozorgi[a,*], Saeed Samet[b], Johan Kwisthout[c], Todd Wareham[a]

[a] Department of Computer Science, Memorial University, St. John's, NL A1B 3X9, Canada
[b] eHealth Research Unit, Faculty of Medicine, Memorial University, St. John's, NL A1B 3V6, Canada
[c] Donders Institute for Brain, Cognition, and Behaviour, Radboud University, Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 25 September 2016
Revised 19 July 2017
Accepted 24 July 2017
Available online 25 July 2017

Keywords:
Competitive influence maximization
Linear threshold model
Community detection
Influence maximization
Social network

ABSTRACT

The main purpose in influence maximization, which is motivated by the idea of viral marketing in social networks, is to find a subset of key users that maximize influence spread under a certain propagation model. A number of studies have been done over the past few years that try to solve this problem by considering a non-adversarial environment in which there exists only one player with no competitor. However, in real world scenarios, there is always more than one player competing with other players to influence the most nodes. This is called competitive influence maximization. Motivated by this, we try to solve the competitive influence maximization problem by proposing a new propagation model which is an extension of the Linear Threshold model and gives decision-making ability to nodes about incoming influence spread. We also propose an efficient algorithm for finding the influential nodes in a given social graph under the proposed propagation model which exploits the community structure of this graph to compute the spread of each node locally within its own community. The aim of our algorithm is to find the minimum number of seed nodes which can achieve higher spread in comparison with the spread achieved by nodes selected by other competitor. Our experiments on real world and synthetic datasets show that our approach can find influential nodes in an acceptable running time.

1. Introduction

The effect of online social networks (OSNs) in our daily life is undeniable as they have introduced new ways of communication and serve as a medium for propagating news, ideas, thoughts and any type of information. Such information can propagate via links between people, which leads to word-of-mouth advertising and its famous application, viral marketing. In viral marketing, the owner of a product gives free or discounted samples of a product to a group of people to gain a large number of adoptions through the word-of-mouth effect. The influence maximization problem, which is motivated by the idea of viral marketing, was introduced by Kempe et al. [1] as finding a subset $S \subseteq V$ containing of k nodes in a graph $G = (V, E)$, such that the spread of influence from S will be maximized. There exists a huge amount of work on solving the influence maximization problem [1–5]. Most of these works assume that there is only one party trying to find influential users in the social network. However, in the real world, multiple parties typically compete simultaneously with similar products. This is called competitive influence maximization (*CIM*). Recently, several works have tried to solve the *CIM* problem [6–13] by proposing new propagation models which are extensions of Linear Threshold and Independent Cascade models [1] or the Distance-based and Wave-propagation models [14].

In this paper, we examine the *CIM* problem from the follower's perspective and propose a new propagation model called DCM (Decidable Competitive Model) which is an extension of the Linear Threshold model. In DCM, each node has the ability to think about the incoming influence spreads from its neighbors for d timesteps and then decide to be activated by the neighbor with the majority of adoption.

In real social networks, people interact with each other based on common interests and strong ties between themselves. Such strong ties between individuals create community structures in social networks, which in turn allow information to circulate within these networks at a high velocity. We propose an algorithm called Competitive Influence Improvement (CI2) which finds the minimum number of influential nodes within their respective communities. Closely related to our work are [15–19] which also

* Corresponding author.
E-mail addresses: ab1502@mun.ca (A. Bozorgi), saeed.samet@med.mun.ca (S. Samet), j.kwisthout@donders.ru.nl (J. Kwisthout), harold@mun.ca (T. Wareham).

http://dx.doi.org/10.1016/j.knosys.2017.07.029
0950-7051/

exploit community structure within social networks to find influential nodes.

Contribution. Our major contributions in this research are summarized as follows:

- We propose the DCM propagation model, the primary intent of this work, which gives decision-making power to nodes based on incoming influence in a competitive version of the LT propagation model.
- We prove the *NP*-hardness of competitive influence improvement under the DCM model.
- We propose the CI2 algorithm to find the minimum number of the most influential nodes for a competitor C_2. This algorithm uses knowledge of the nodes selected by a competitor C_1 so that C_2 can achieve more influence spread by spending less budget. Computing the spread of seed nodes is done locally inside communities of the input graph, which results in a substantial decrease in running time.
- We conduct experiments using three real and three synthetic datasets to show that CI2 can find influential nodes in an acceptable running time. Synthetic datasets are generated with the same number of nodes and edges but different community structures in order to track the effect of community structure of networks on our approach. Also, we consider the effect of the algorithm which finds the seed nodes for the first competitor on the seed nodes which will be selected by the second competitor by conducting different experiments which use well-known algorithms [15,20,21] to extract the first competitor's seed set.

Organization. In Section 2, we review some background knowledge to enable a better understanding of the upcoming concepts. In Section 3, we describe our Linear-Threshold-based propagation model, prove that competitive influence improvement under this model is *NP*-hard, and propose our CI2 algorithm. Section 4 describes the experiments performed with real and synthetic data to evaluate the proposed approach. Finally, in Section 5, we give our conclusions and directions for future work.

2. Background and related work

In [1], Kempe et al. introduced two propagation models to address the influence maximization problem, the Linear Threshold (LT) and Independent Cascade (IC) models. In both models, a threshold value $\theta \in [0, 1]$ is assigned to each node and each node can be active or inactive. Also, each edge from node u to node v has an influence weight $p_{u,v} \in (0,1]$. At first, all nodes are inactive except the nodes in set S which have been activated before as seed nodes and the propagation process is started from them. In LT, an inactive node u can be activated at time t if $f_v(S) > \theta_v$, where S stands for v's neighbors which are activated at time $t-1$. As is mentioned in [1], the value of f_v is initialized as

$$f_v(S) = \sum_{u \in S} b_{v,u}$$

where $b_{v,u}$ is the weight of edge (v, u). In the LT model, the sum of all edge weights between v and its neighbors should be less than 1 [1].

In IC, the activation process is the same as that in LT except that in IC, an activated node u has only one chance to activate its inactive neighbor v with probability $p_{u,v}$.

Community structure. Communities are subsets of nodes in the graph with more edges between them and fewer edges to nodes in different communities [22]. Community detection is formulated as a clustering problem — that is, given the full graph $G = (V, E)$, partition the vertex set into k subsets $S_1, S_2, \ldots, S_K$, such that $\bigcap_{i=1}^{k} S_i = \emptyset$ and $\bigcup_{i=1}^{k} S_i = V$. A quality metric $Q(S_1, \ldots, S_K)$ is defined over the partitions and a community detection algorithm will try to find a partition that maximizes or minimizes Q depending on its nature. This is for non-overlapping community detection and one can simply remove the constraint $\bigcap_{i=1}^{k} S_i = \emptyset$ to get the overlapping version [23].

Competitive influence maximization. Recently, several works have tried to solve the competitive influence maximization by introducing new propagation models to simulate the competitive manner of the competitors which are mostly an extension of the LT or IC models. Some efforts such as [14,24] look to this problem from the follower's perspective, i.e. they assume that there are two competitors trying to find some influential nodes and the second competitor starts his process with knowledge of the seed nodes selected by the first competitor and tries to find some new seed nodes other than the ones selected by the first competitor to achieve more influence spread. In some other works such as [6,7] one competitor tries to block the effect of the other competitor. In [8], Lu et al. solve the competitive influence maximization problem from the host's perspective, i.e. the owner of the social network is responsible for fairly allocating some specific number of seed nodes to the competitors. In the next section, we explain [8,25] in more details as they are more related to our work.

3. Propagation model and algorithm

In this section, we introduce the DCM propagation model (which is an extension of the LT model), compare DCM with the Weighted-proportional (WPCLT) [25] and *K*-LT [8] models, and prove the *NP*-hardness of competitive influence improvement under DCM. Finally, we introduce the CI2 algorithm to find the influential nodes in a social network under our competitive propagation model.

3.1. DCM propagation model

In the DCM propagation model, each node can be in one of the following states: *inactive, thinking*, $active^+$ or $active^-$. Suppose there are two competitors who try to advertise for their products over a social network. We denote the first competitor with the + sign and the second competitor with the - sign and each node v, picks a threshold value θ_v uniformly at random from [0,1]. Let S_1 be the seed set selected by the first competitor and S_2 be the seed set selected by the second one. At first all nodes except those in the seed set are *inactive*. The activation process of node v is as follows: at time $t > 1$ if the total incoming influence weight from the in-neighbors of v which are active ($N^{in}_{active}(v)$) reaches the threshold value of v, its state changes to *thinking*, which means the state of node v would be changed with probability

$$\sum_{u \in N^{in}_{active}(v)} p_{u,v} \geq \theta_v \tag{1}$$

Node v remains in *thinking* state after this state change for d timesteps and after that, it decides to become $active^+$ or $active^-$ based on the maximum total incoming influence weight from its in-neighbors. Let A^+_{t+d} be the set of in-neighbor nodes of v with state $active^+$ and A^-_{t+d} be the set of in-neighbor nodes of v with state $active^-$ at time $t+d$. The state of node v changes from *thinking* to $active^+$ or $active^-$ as follows:

$$v_{state} = \begin{cases} active^+, & \text{if } \sum_{u \in A^+_{t+d}} p_{u,v} > \sum_{u \in A^-_{t+d}} p_{u,v} \\ active^-, & \text{otherwise} \end{cases} \tag{2}$$

In the WPCLT model, which was proposed by Borodin et al. [25], the state of a node v changes to $active^+$ with probability $\sum_{u \in A^+_{t-1}} p_{u,v} / \sum_{u \in A_{t-1}} p_{u,v}$. This means that a node v would be activated as $active^+$ ($active^-$) at time t with probability equal to the

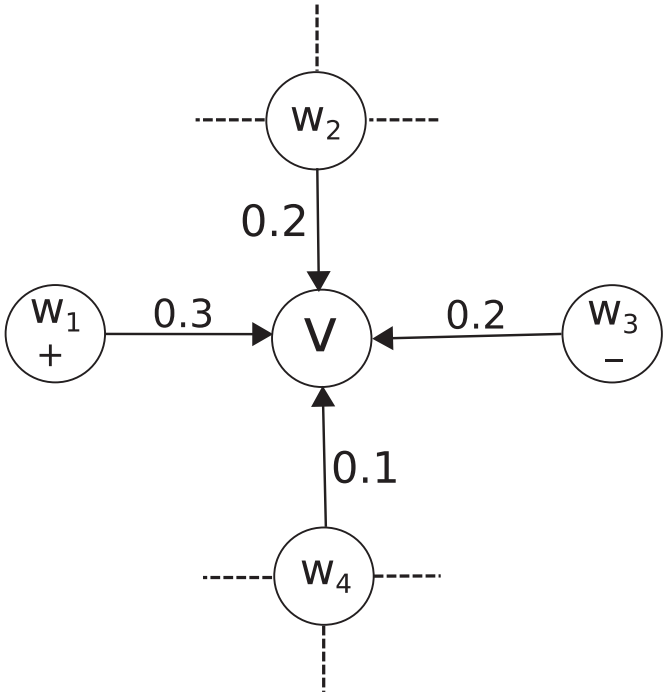


Fig. 1. An example graph substructure.

ratio between the total weight from the in-neighbors with state *active*$^+$ (*active*$^-$) and that from all active in-neighbors. Lu et al. [8] noted that in the WPCLT model, when a node is about to activate, the neighbors which have been activated in all previous timesteps are considered; this, however, does not assure recency, which is when the customer's choice among competing products relies more on recent than old information [26,27]. Hence, Lu et al proposed the *K*-LT model [8] in which the activation probability of node v at time t relies on its in-neighbors which have just been activated at time $t-1$ rather than all past exposures such that the state of a node v changes to *active*$^+$ with probability $\sum_{u\in A^+_{t-1}\setminus A^+_{t-2}} p_{u,v} / \sum_{u\in A_{t-1}} p_{u,v}$.

In both the WPCLT and *K*-LT models, a node cannot decide whether it would be activated or not, while in the real world when people decide to whether to purchase or consume a product, they are influenced by the decisions made by others. Even when individuals seem to be making decisions separately, they are likely to be mindful of the preferences of others [28].

As an example, imagine that the structure in Fig. 1 is part of a social graph in which, nodes w_1 and w_3 have been activated before as *active*$^+$ and *active*$^-$ respectively and nodes w_2 and w_4 haven't been activated yet (we show that nodes w_2 and w_4 are connected to other nodes of the graph by the dotted lines around them). If $\theta_v = 0.25$, at time t_1, node v get influenced in both the WPCLT and *K*-LT models as the total incoming influence weight is bigger than its threshold value and then, node v would be activated as *active*$^+$ as $p_{w_1,v} > p_{w_3,v}$. Now imagine that nodes w_2 and w_4 would be activated as *active*$^-$ by other nodes of the graph at time t_2; This means that in both the WPCLT and *K*-LT models, node v is in *active*$^+$ state at time t_2, while the majority of its neighbors have been activated as *active*$^-$. But in DCM, the state of node v changes from *inactive* to *thinking* at time t_1 and its state remains stable for d timesteps so that it can consider different influence spreads, after which it decides to be activated by the influence spread which is accepted by the majority of its neighbors. This causes the state of node v to change from *thinking* to *active*$^-$ after d timesteps in the DCM model. Therefore, it is reasonable to give the ability to the nodes to think about the incoming influence spread.

3.2. NP-*hardness of competitive influence improvement under DCM*

Let a vertex-labeled and arc-weighted community-structure graph be $G = (V, A, l, p)$ where V is the vertex-set, A is the arc-set, l: $V \to \{inactive, thinking, active^+, active^-\}$ is the vertex-label function, and p: $A \to (0, 1]$ is the arc-weight function. Consider the vertex-labels in G after influence propagation under the DCM model is complete, and let N_- and N_+ be the number of vertices in G with labels *active*$^-$ and *active*$^+$, respectively. We want to select a set of vertices S_+ of size at most k relative to a given set S_- such that S_+ ultimately has more influence than S_- — that is, we want N_+ to be larger than N_-. Let $D(S_-, S_+) = \max(0, N_+ - N_-)$ and $\sigma_D(S_-, S_+)$ be the expected value of $D(S_-, S_+)$ (with this expectation arising from the various θ_v being chosen uniformly at random from the interval [0, 1][1]).

The problem of influence improvement under DCM can now be stated formally as follows:

DCM Competitive Influence Improvement (DCI-Imp)

Input: A community structure graph $G = (V, A, l, p)$, a seed-set S_-, and positive integers d, $k > 0$ and $c \geq 0$.

Output: A seed-set S_+ of size at most k such that $\sigma_D(S_-, S_+) > c$, if such a S_+ exists, and special symbol $\perp$ otherwise.

This problem looks easy, as we are only requiring some fixed amount of improvement (which is the smallest possible, *i.e., any* improvement, when $c = 0$). However, looks can be deceiving.

Theorem 1. *If DCI-Imp is polynomial-time solvable when* $d = 1$ *and* $c = 0$ *then* $P = NP$.

Proof. We first show that DCI-Imp$_D$, the decision version of DCI-Imp (which asks whether or not the requested S_+ exists), is *NP*-hard by a polynomial-time reduction from the following *NP*-hard problem:

Dominating Set [29, Problem GT2]

Input: A graph $G = (V, A)$ and an integer k.

Question: Is there a dominating set in G of size at most k, *i.e.*, is there a subset $V' \subseteq V$, $|V'| \leq k$, such that for each $v \in V$, either $v \in V'$ or $\exists (v, v') \in E$ such that $v' \in V'$?

For a graph $G = (V, E)$, let $N(u)$ be the set of all vertices in G that are neighbors of vertex u in G (including u itself). Given an instance $(G = (V, E), k)$ of Dominating Set, construct the following instance $(G' = (V', A, l, p), S_-, d, k', c)$ of DCI-Imp$_D$:

- $V' = V_1 \cup V_2 \cup V_3$ where $V_1 = \{v^1_1, v^1_2, \ldots, v^1_{|V|}\}$, $V_2 = \{v^2_1, v^2_2, \ldots, v^2_{|V|}\}$, and $V_3 = \{v^3_1, v^3_2, \ldots, v^3_{|V|+(k-1)}\}$.
- $A = A_1 \cup A_2$ where $A_1 = \{(u, v) \mid u \in V_1, v \in V_2, \text{ and } v \in N(u)\}$ and A_2 ensures that each vertex in V_2 has incoming arcs from exactly two distinct vertices in V_3.
- The initial labeling l of V is such that all vertices in V_3 have label *active*$^-$ and all other vertices have label *inactive*.
- p is such that the weight of each arc in A_1 is $1/|V|$ and the weight of each arc in A_2 is $1/4|V|$.
- $S_- = V_3$.
- $d = 1$, $k' = k$, and $c = 0$.

[1] Uniform choice of θ_v is consistent with previous linear-threshold-based models of influence propagation such as that in [1]. However, the results in this section apply relative to θ_v choice under *any* distribution over [0, 1] (including, but not limited to a uniform distribution) as long as that choice is ergodic, *i.e.*, there must be a finite non-zero probability for every $\theta_v \in [0, 1]$ being picked, including 0 and 1 as border cases.

This construction can be done in polynomial time in the size of the given instance of DOMINATING SET.

By the construction of G' above, the only vertices that can change label from *inactive* to *thinking* (and thereafter to either *active*$^+$ or *active*$^-$) under DCM are the vertices in V_2. A vertex v in V_2 will only be able to change label from *inactive* to *thinking* if $\theta_v > 1/2|V|$. Such a *thinking* vertex v will then have final label *active*$^-$ unless there is at least one vertex u in V_1 with label *active*$^+$ that has an arc to v (as the weight of such an arc would outweigh the weights of the two incoming arcs from V_3 and hence force v to have label *active*$^+$). As $|V_2| = |V| + (k-1)$, $D(S_-, S_+)$ can only have value 0 or 1 for a given S_+, with the value of 1 occurring if and only if $\theta_v > 1/2|V|$ for each vertex v in V_2 and the k vertices in S_+ force all vertices in V_2 to have label + under DCM. However, by the construction of G', the vertices in such a S_+ correspond to a dominating set of size k in G. Given that θ_v is drawn uniformly from [0, 1], there is a S_+ such that for some values of θ_v, $D(S_-, S_+) = 1$ and hence $\sigma(D(S_-, S_+)) > c = 0$ if and only if there is a dominating set of size k in the given instance of DOMINATING SET.

The above establishes that DCI-Imp$_D$ is *NP*-hard. To complete the proof, note that any polynomial-time algorithm for DCI-Imp can be used to solve DCI-Imp$_D$ in polynomial time, which, by the definition of *NP*-hardness, would imply that $P = NP$. □

This result shows that if the conjecture $P \neq NP$ is true (which is widely believed within Computer Science [29,30]), the simplest type of competitive influence improvement cannot be computed correctly for all inputs in polynomial time.

One might still hope that this problem is practically solvable in polynomial time. Two senses in which this might be possible are:

1. DCI-Imp is solvable in effectively polynomial time under certain restrictions. For example, there might be an algorithm for DCI-Imp that is exponential-time in general relative to the number k of nodes in S_+ but runs in effectively polynomial time when k is a small constant. Such an algorithm would have runtime $f(k)n^x$ where f is an arbitrary function, n is the input size, and x is a constant. This notion of effective polynomial-solvability is the fixed-parameter tractability underlying Downey and Fellow's theory of parameterized computational complexity [31].
2. DCI-Imp is solvable in polynomial time by a probabilistic algorithm with high probability, *e.g.*, $> 2/3$. This notion of solvability is essentially what many types of stochastic heuristics (in particular, those based on evolutionary computation) promise.

However, it turns out that these types of solvability are also unavailable to us.

Theorem 2. *If DCI-Imp is fixed-parameter tractable relative to parameter k when $d = 1$ and $c = 0$ then $FPT = W[2]$.*

Proof. In the reduction given in the proof of Theorem 1, the size k of the requested dominating set in the given instance of DOMINATING SET is equal to the size k' of S_+ in the constructed instance of DCI-Imp. Hence, this reduction is also a parameterized reduction relative to parameter k' in the constructed instance of DCI-IMP. This result then follows from the $W[2]$-hardness of DOMINATING SET relative to parameter k and the inclusion of *FPT* in $W[2]$ [31]. □

Theorem 3. *If $P = BPP$ and DCI-Imp is polynomial-time solvable by a probabilistic algorithm which operates correctly with probability $\geq 2/3$ then $P = NP$.*

Proof. *BPP* is considered the most inclusive class of problems that can be efficiently solved using probabilistic methods (in particular, methods whose probability of correctness is $\geq 2/3$ and can be efficiently boosted to be arbitrarily close to probability one) [32, Section 5.2]. If DCI-Imp has a probabilistic polynomial-time algorithm which operates correctly with probability $\geq 2/3$, DCI-Imp$_D$ is in *BPP*. However, as $BPP = P$ and DCI-IMP$_D$ is *NP*-hard by Theorem 1 above, the definition of *NP*-hardness then implies that $P = NP$. □

These results show that if, in addition to $P \neq NP$, the conjectures $FPT \neq W[2]$ and $P = BPP$ are also true (both of which are widely believed within Computer Science (see [31,33] and [32, Section 5.2], respectively)), the simplest type of competitive influence improvement cannot be practically computed in either of the senses above (the former relative to small-sized S_+).

To summarize, the results in this section effectively rule out several popular types of efficient algorithms for competitive influence improvement under the DCM model. As such, they also justify the search for and use of heuristic algorithms such as the greedy community-based algorithm described in the remainder of this paper.

3.3. Community-based algorithm

Motivated by the useful characteristics of communities in social networks which we mentioned previously in Section 2, we decided to base our CI2 algorithm for competitive influence improvement on influential nodes in the community structure of input graph G. An overview of CI2 is shown in Fig. 2. At first, the communities of the input graph are extracted; these communities are denoted by labels C_1, C_2 and C_3 in the figure. Then, in each community, the most influential node is selected as a seed candidate. Finally, the node which has the maximum influence spread among candidate nodes is selected as a seed node. The selected seed node for the second competitor is denoted with a + sign in Fig. 2. The CI2 algorithm is explained in more detail in the following section.

Community detection. Many approaches have been proposed to solve the community detection problem in online social networks. MLAMA-Net [34] is an evolutionary algorithm, which solves the community detection problem in a network of chromosomes using evolutionary operators and local searches. In MLAMA-Net, each node includes a chromosome and a learning automaton. Each chromosome explores a community for its corresponding node using evolutionary operators and improves the community by a local search. The learning automaton is responsible for saving the histories of local searches of each node. Very related to MLAMA-Net, Khomami et al. proposed DLACD [35], which extracts the community structure of complex networks based on distributed learning automata.

To find the communities of the input graph, we use the listener-speaker approach introduced in [36] in conjunction with the information diffusion model. First, each node v is considered as a unique community with community label equal to its ID. Then one node is selected as the consumer of the information and receives the community labels from its neighbors. To decide which label will be accepted by the selected node, the weights of the incoming edges are considered. For example, suppose a selected node v has four in-neighbors u_1, u_2, u_3 and u_4, with edge weights 0.1, 0.05, 0.3 and 0.25 respectively. When node v considers the labels from its neighbors, it will weight each label as $0.1/(0.1 + 0.05 + 0.3 + 0.25)$ for node u_1, $0.05/(0.1 + 0.05 + 0.3 + 0.25)$ for node u_2, $0.3/(0.1 + 0.05 + 0.3 + 0.25)$ for node u_3 and $0.25/(0.1 + 0.05 + 0.3 + 0.25)$ for node u_4. Then, the selected node accepts one label from the collection of the received labels from its neighbors based on a specified listening rule, such as the popularity of the observed labels in the ongoing step. This process is then repeated and at each step, one new node is selected as the consumer of the information. The main reason for using the approach of Xie et al. [36] to find graph communities is its ability (verified by experiments done in [36]) to efficiently find high-quality communities.

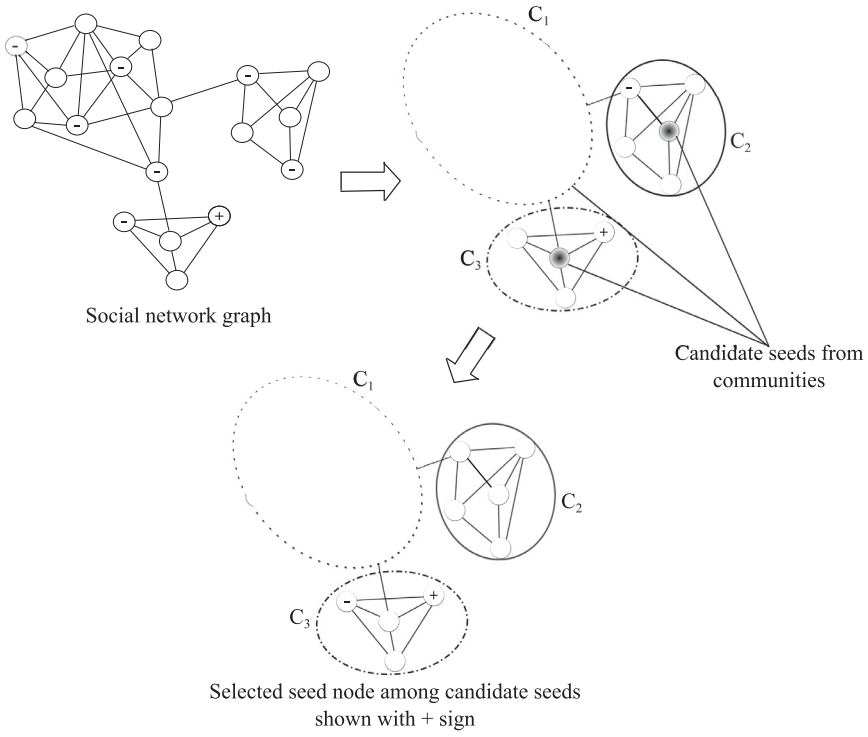


Fig. 2. An overview of the CI2 community-based algorithm.

Seed selection. After constructing the community structure from graph G, we need to find the minimum number of nodes for the second competitor which achieve higher influence spread than the influence spread achieved by the nodes selected by the first competitor. The spread value of node v is the number of nodes which can be accessed and activated by node v and the spread of nodes in set S is the sum of spreads of each node in the set.

In each community C_i, we locally run the simple greedy algorithm [1] which uses DCM as its propagation model to find the most influential node in C_i and store the node ID and its spread value in candidate seed set S'. Note that the node which is selected as a candidate node in this step should be different from the nodes which have been selected for the first competitor. The size of S' is equal to the number of communities and in each step, this set is updated to hold the new candidate seeds of each community. Among the candidate seeds, the one which has the maximum marginal gain is selected and added to S_2. S_1 and S_2 are seed sets of the first and second competitors respectively such that

$$S_2 = S_2 \cup arg\ max_{v \in S'}(\delta(S_2 \cup \{v\})) \quad (3)$$

Stop criterion. The above seed selection process is continued until the influence spread achieved by nodes in S_2 reaches the influence spread achieved by nodes in S_1. The steps of our community-based algorithm are shown in Algorithm 1.

Algorithm 1 The CI2 community-based algorithm.

Require: $G = (V, E), S_1$
Ensure: S_2
1: $S_2 \leftarrow \{\}$
2: Construct the communities of the input graph G and store them in *Communities Set*
3: **while** $\delta(S_2) \leq \delta(S_1)$ **do**
4: $S' \leftarrow \{\}$
5: **for** each $C_i \in CommunitiesSet$ **do**
6: call simple greedy algorithm to find most influential node $s_i \notin S_1$ in community C_i
7: $S' = S' \cup \{s_i\}$
8: $S_2 = S_2 \cup arg\ max_{v \in S'}(\delta(S_2 \cup \{v\}))$
9: return S_2

4. Evaluations

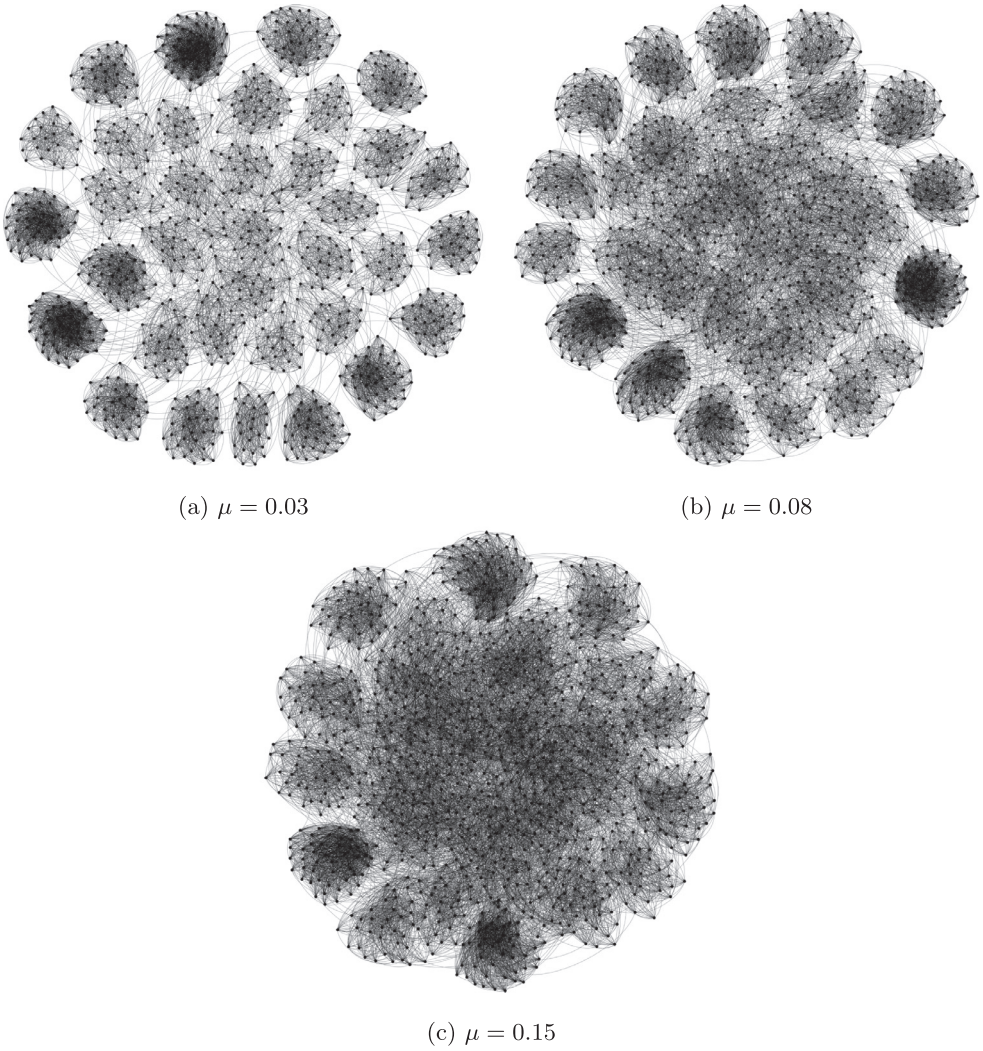
To evaluate the efficiency of our community-based algorithm in finding high-quality seed sets in an acceptable running time, we have done our experiments on three real-world datasets. These experiments show that there is a trade-off between running time and the quality of seed nodes selected by a competitor by changing parameter d, the number of timesteps a node can think about the incoming influence spread. To track the effect of community structure of networks on our approach, we also have used three synthetic datasets with the same numbers of nodes and edges but different community structures. Our code is implemented in C++ and all experiments were run on a Linux (CentOS 7.0) machine with a 3.6GHz Intel Core i7 CPU and 16GB of memory.

4.1. Experiments setup

Dataset. The real-world datasets that we have used in our experiments are available from the SNAP library on the Stanford University website[2] and their specifications are shown in Table 1. Using the community detection algorithm described in Section 3.3, we found the communities in these datasets; the specifications of these communities are shown in Table 2. In both Tables 1 and 2, the # sign indicates the number of elements.

To generate our synthetic datasets, we used LFR benchmark [37], which specifies the heterogeneity of the networks by the distributions of node degrees and community sizes. The node degrees and community sizes are taken from power law distributions with exponents γ and β respectively. By assigning three different values

[2] http://snap.stanford.edu/



(a) $\mu = 0.03$ (b) $\mu = 0.08$

(c) $\mu = 0.15$

Fig. 3. Three networks generated by LFR benchmark with 1000 nodes and different mixing parameter values.

Table 1
Specifications of real standard datasets.

	NetHEPT	Slashdot	Amazon
#nodes	15K	82K	262K
#edges	62K	948K	1.2M
Average out-degree	4.12	9.8	9.4
Maximum out-degree	64	1527	425
#connected components	1781	1	1
Largest component size	6794	82K	262K

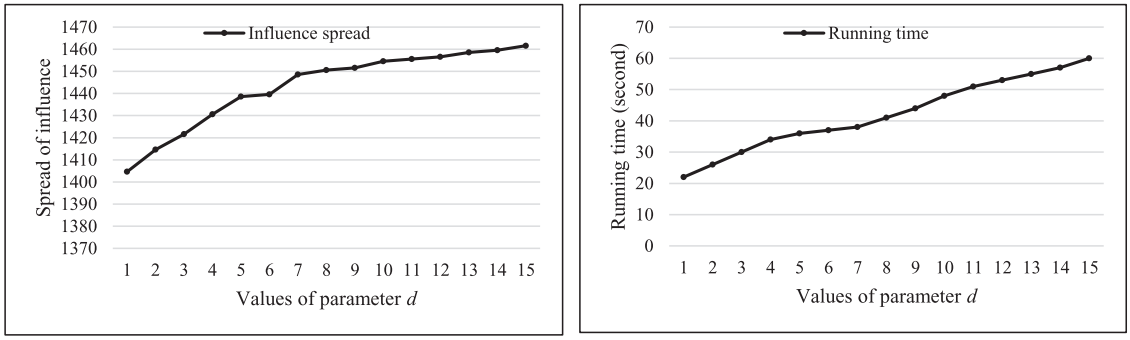
(0.03, 0.08 and 0.15) to the mixing parameter μ, setting $N = 1000$ (the number of nodes), $\gamma = 2$ and $\beta = 1$, and varying the in-degree of nodes between 0 to 50 with average 15 and the community size between 20 and 50, we generated three different datasets, which are visualized in Fig. 3. Note that the mixing parameter determines the fraction of one node's links to other nodes inside its community and nodes outside its community. More specifically, each node shares a fraction of $1 - \mu$ of its links with the nodes inside its community and a fraction of μ with nodes belonging to other communities.

Algorithms. We ran the implementation of our CI2 algorithm with the above described datasets as well as the following algorithms:

- The greedy approximation algorithm [1], which uses Monte Carlo (MC) simulations to compute the spread of a node within a factor of $(1 - 1/e - \epsilon)$ for any $\epsilon > 0$. In this algorithm, MC simulations were performed 10,000 times to compute the spread of the seed sets.

Table 2
Specification of communities in the real standard datasets.

	NetHEPT	Slashdot	Amazon
#Communities	2901	639	32674
#Nodes in the biggest community	407	80K	2852
#Edges in the biggest community	2938	700K	7169
Average out-degree in the biggest community	6.2	11.67	2.52
Maximum out-degree in the biggest community	48	1407	5



(a): Spread of influence (b): Running Time

Fig. 4. The effect of parameter d on influence spread and running time.

- The INCIM algorithm [15], which computes the spread value of each node such that it is very close to that computed by Monte Carlo simulations. This algorithm finds the influence of each node as a combination of its local and global influences to track the effect of each node in its community and also, the effect of each community in the input graph.
- The IPA algorithm [20], which selects as seed node the node which has maximum influence propagation probability in each iteration. Based on the recommendations in [20], we set parameter $threshold = 0.005$.
- The LDAG algorithm [21], which computes the spread of each node in its belonging DAG (Directed Acyclic Graph) locally and achieves good results in quality of seed nodes. Based on the recommendations in [21], we set parameter $\theta = 1/320$.
- The HighDegree algorithm [1], which selects as seed node the node which has maximum out-degree.

The results of these algorithm runs are discussed in the next section. Two characteristics of these algorithms are worth noting. First, both the INCIM and IPA algorithms use the idea of communities to find influential nodes and like LDAG have reasonable running times and find good quality nodes. Second, though the HighDegree and greedy approximation algorithms are now almost 15 years old and may thus appear to be obsolete, they are still very commonly used in comparisons involving recently-proposed approaches to competitive and non-competitive influence maximization [11–13,38,39].

4.2. Experimental results

Setting parameter \boldmath d. As we mentioned in previous sections, in the DCM propagation model, each node can think about incoming influence spread for d timesteps and then decides to be activated based on the majority of its neighbor's adoptions. When $d = 1$, information propagates the same as in the LT propagation model, where nodes can think for only one timestep about incoming influence spread, and we have the best running time. As the value of parameter d is increased, nodes have more time to think about incoming influence, which allows the selection of seed nodes with higher quality; however, this also results in increased running time, too. In Fig. 4, we can see the changes in influence spread and running time associated with values of parameter d from 1 to 15. We did this experiment on the NetHEPT dataset with a seed set of size 50.

As we can see in Fig. 4(a), as the value of parameter d changes from 1 to 7, there is a remarkable increase in influence spread. However, changing the value of parameter d from 8 to 15 provides a marginal increase in influence for a great cost of increased running time (see Fig. 4(b)). We performed the same experiment on the Amazon and Slashdot datasets and got essentially the same results. Hence, we set the value of parameter d to 7 in the remainder of our experiments.

Table 3
Differences between the influence spread values computed by MC and CI2 (in percentage).

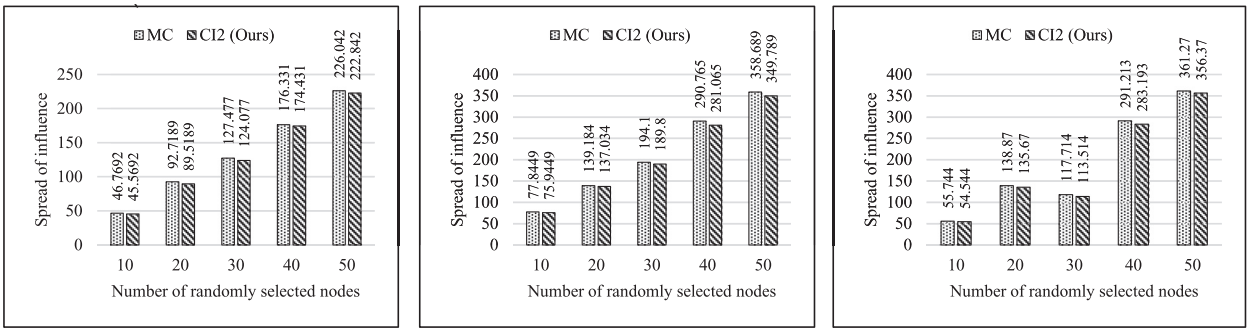
Datasets	Seed set size				
Used	10	20	30	40	50
Amazon	2.07%	2.63%	2.51%	1.81%	2.13%
NetHEPT	2.44%	1.54%	2.21%	3.33%	2.48%
Slashdot	2.15%	2.30%	3.56%	2.75%	1.35%

Efficiency of proposed algorithm. To study how efficiently we compute the spread of nodes by extracting seed nodes from communities, we randomly selected 5 different set of nodes as seed sets of size 10, 20, 30, 40 and 50 from each of the Amazon, NetHept and Slashdot datasets and ran CI2 and Monte Carlo simulations to compute the spread of these seed sets. The results are shown in Fig. 5.

The differences between the values computed by MC and CI2 are shown in Table 3. The results in Fig. 5 and Table 3 show that the values which are computed by CI2 are very close to the values computed by Monte Carlo simulations which computes the spread of a node with a good approximation guarantee.

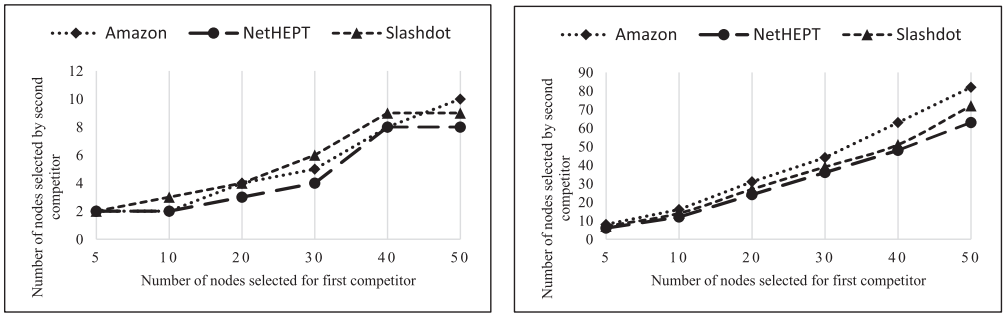
In these runs, calculating the spread of nodes locally inside the communities they belong to seems to cause a huge decrease in running time. To verify this, we used our CI2 algorithm to find a seed set of size 50 from the NetHEPT dataset by (1) considering existing communities and calculating the spread values locally inside communities and (2) calculating the spread of each node in the whole graph without considering their own community. In the former case, CI2 finds the seed nodes in approximately 22 s, while it finds such seed nodes in approximately 70 min in the later case. This clearly shows the effect of localizing the spread calculations in running time, which is the result of considering community structure in the CI2 algorithm.

Seed selection. To simulate the competitive condition from the follower's perspective, we chose some seed nodes randomly and activated them for the first competitor as negative and ran CI2 to select the minimum number of nodes with higher influence spread for the second competitor. The nodes selected for the second competitor should be different from the ones selected for the first competitor. We also did the same process by running the greedy approximation algorithm and the INCIM [15], IPA [20], LDAG [21] and HighDegree [1] algorithms with different values for k as their budgets. The generated seed sets are of size 5, 10, 20, 30, 40 and 50. The minimum number of nodes selected by CI2 to defeat the first competitor in each case is shown in Fig. 6.

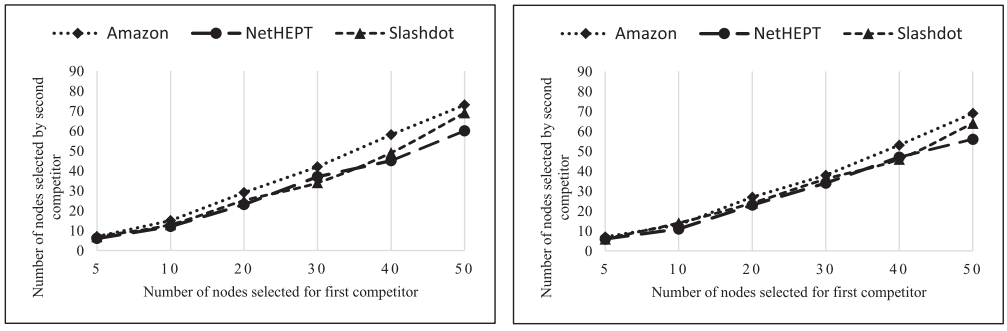


(a): Amazon (b): NetHEPT (c): Slashdot

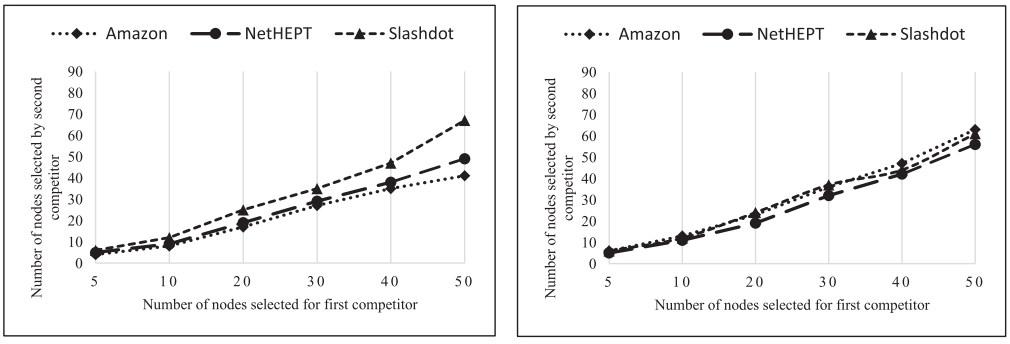
Fig. 5. The influence spreads achieved by different randomly selected seed sets.



(a): Randomly seed nodes selected for first competitor (b): Actual seed nodes selected for first competitor using MC



(c): Actual seed nodes selected for first competitor using INCIM (d): Actual seed nodes selected for first competitor using LDAG



(e): Actual seed nodes selected for first competitor using IPA (f): Actual seed nodes selected for first competitor using HighDegree

Fig. 6. Minimum numbers of nodes which need to be selected by the second competitor.

As we can see in Fig. 6, the minimum number of nodes which is required to be selected by the second competitor to achieve higher influence spread depends deeply on how the seed nodes are selected by the first competitor. In Fig. 6(a), in which the seed nodes of the first competitor are selected randomly, fewer nodes in each set are required to defeat the first competitor. However, when we extract the actual seed nodes by running the algorithms mentioned above, in each set of nodes, more nodes must be selected by the second competitor. For example, in Slashdot dataset in Fig. 6(b), 72 seed nodes must be selected to achieve higher influence spread than the spread achieved by an actual seed set of size 50, while only 13 nodes need to be selected to achieve higher influence

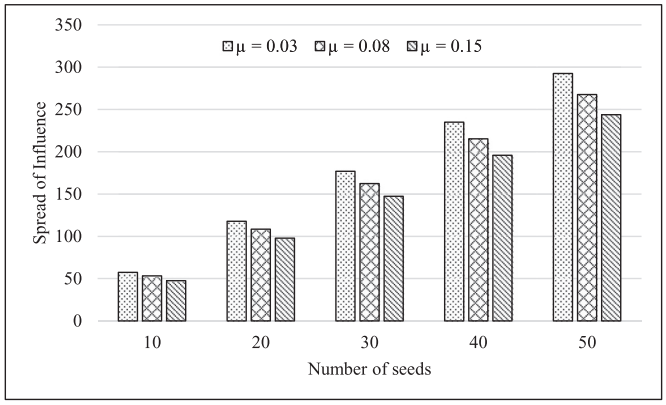


Fig. 7. Influence spread achieved on LFR networks with different mixing parameter values.

spread when the nodes in the set of size 50 are selected randomly. Also, the algorithm which is used to extract the actual seed nodes for the first competitor affects the number of seed nodes that need to be selected by the second competitor, as different algorithms achieve different levels of quality in their seed node extraction. In Figs. 6(c–f), the number of seed nodes that need to be selected to defeat the first competitor are 69, 67, 64 and 61 if the seed sets of size 50 are extracted by the INCIM, IPA, LDAG and HighDegree algorithms respectively from the Slashdot dataset. These figures tell us that as seed nodes are selected with higher quality for the first competitor, more seed nodes other the selected ones must be selected by the second competitor.

The effect of community structure on seed selection. To study how the structure of communities can affect the quality of seed nodes, we ran CI2 on three synthetic LFR networks [37] which vary their community structures by assigning different values to the mixing parameter in LFR benchmark. If the value of the mixing parameter is smaller, the communities are loosely connected to each other and there are few links between nodes in different communities. The results of our runs on LFR networks in Fig. 7 show that CI2 acts is better at finding seed nodes in networks whose community structures are more prominent. This is demonstrated by the observation that in the first network with the smallest mixing parameter, the influence spread achieved by the extracted seed set is higher than the two other networks with larger mixing parameter values. One way to help CI2 act better in networks with less prominent community structure is to consider the effect of border nodes [15] on influence spread computations. Border nodes have at least one link to nodes in other communities, which allows the spread of influence from a border node's own community to others and vice versa. As the main point of this paper is to propose the DCM propagation model and our aim of using community structure in CI2 algorithm is to improve the running time of finding seed nodes, we will address the issue of border nodes in future work.

5. Conclusion

In this paper we studied competitive influence maximization from the follower's perspective and introduced the Decidable Competitive Model (DCM), an extended version of the LT model, for influence propagation in a competitive fashion. To find the influential nodes in a social network graph, we proposed an efficient algorithm which extracts the communities of the input graph and finds the most influential node in each community as a seed candidate. Then the final seed nodes are selected from the set including seed candidates. The size of the final seed set should be as small as possible, i.e. we assign the seed nodes to the second competitor so as to achieve higher influence spread comparing with the spread achievement of the first competitor's seed set by spending less budget. The ability of nodes to think about incoming influence in the DCM propagation model simulates a realistic situation in which a node's tendency is toward the spread of influence which has been adopted by the majority of their neighbors after d timesteps. Adding parameter d to simulate the thinking ability of nodes results in finding influential nodes with higher quality; moreover, by calculating the spread values of each node locally inside its community, we achieved an acceptable running time. The results of our experiments on different real and synthetic datasets proof the efficiency of our proposed algorithm and propagation model.

There are several promising directions for future research. First, faster algorithms than the greedy algorithm used here could be used inside communities to find influential nodes. Second, the effect of border nodes on the quality of seed nodes should be investigated in networks with different community structures. Finally, the effect of temporal evolution of networks on influence maximization should be analyzed.

References

[1] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of SIGKDD, ACM, 2003, pp. 137–146.
[2] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost–effective outbreak detection in networks, in: Proceedings of SIGKDD, ACM, 2007, pp. 420–429.
[3] H. Yu, S.-K. Kim, J. Kim, Scalable and parallelizable processing of influence maximization for large-scale social networks, in: Proceedings of ICDE, IEEE Computer Society, 2013, pp. 266–277.
[4] A. Goyal, W. Lu, L.V. Lakshmanan, Simpath: an efficient algorithm for influence maximization under the linear threshold model, in: Proceedings of ICDM, IEEE, 2011, pp. 211–220.
[5] W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in: Proceedings of ICDM, IEEE, 2010, pp. 88–97.
[6] X. He, G. Song, W. Chen, Q. Jiang, Influence blocking maximization in social networks under the competitive linear threshold model, in: Proceedings of ICDM, SIAM, 2012, pp. 463–474.
[7] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, Y. Yuan, Influence maximization in social networks when negative opinions may emerge and propagate, in: Proceedings of ICDM, SIAM, 2011, pp. 379–390.
[8] W. Lu, F. Bonchi, A. Goyal, L.V. Lakshmanan, The bang for the buck: fair competitive viral marketing from the host perspective, in: Proceedings of SIGKDD, ACM, 2013, pp. 928–936.
[9] S. Bharathi, D. Kempe, M. Salek, Competitive influence maximization in social networks, in: Internet and Network Economics, Springer, 2007, pp. 306–311.
[10] C. Budak, D. Agrawal, A. El Abbadi, Limiting the spread of misinformation in social networks, in: Proceedings of the 20th International Conference on World Wide Web, ACM, 2011, pp. 665–674.
[11] W. Liu, K. Yue, H. Wu, J. Li, D. Liu, D. Tang, Containment of competitive influence spread in social networks, Knowl. Based Syst. 109 (2016) 266–275.
[12] M.A.M.A. Kermani, S.F.F. Ardestani, A. Aliahmadi, F. Barzinpour, A novel game theoretic approach for modeling competitive information diffusion in social networks with heterogeneous nodes, Physica A 466 (2017) 570–582.
[13] H. Wu, W. Liu, K. Yue, J. Li, W. Huang, Selecting seeds for competitive influence spread maximization in social networks, in: International Conference of Young Computer Scientists, Engineers and Educators, Springer, 2016, pp. 600–611.
[14] S. Shirazipourazad, B. Bogard, H. Vachhani, A. Sen, P. Horn, Influence propagation in adversarial setting: how to defeat competition with least amount of investment, in: Proceedings of CIKM, ACM, 2012, pp. 585–594.
[15] A. Bozorgi, H. Haghighi, M.S. Zahedi, M. Rezvani, Incim: a community-based algorithm for influence maximization problem under the linear threshold model, Inf. Process. Manage. 52 (6) (2016) 1188–1199.
[16] C. Kim, S. Lee, S. Park, S.-g. Lee, Influence maximization algorithm using markov clustering, in: Database Systems for Advanced Applications, Springer, 2013, pp. 112–126.
[17] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 1039–1048.
[18] M. Hosseini-Pozveh, K. Zamanifar, A.R. Naghsh-Nilchi, A community-based approach to identify the most influential nodes in social networks, J. Inf. Sci. 43 (2) (2017) 204–220.

[19] Y. Zhao, S. Li, F. Jin, Identification of influential nodes in social networks with community structure based on label propagation, Neurocomputing 210 (2016) 34–44.
[20] J. Kim, S.-K. Kim, H. Yu, Scalable and parallelizable processing of influence maximization for large-scale social networks? in: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE, 2013, pp. 266–277.
[21] W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in: Data Mining (ICDM), 2010 IEEE 10th International Conference on, IEEE, 2010, pp. 88–97.
[22] R.D. Luce, A.D. Perry, A method of matrix analysis of group structure, Psychometrika 14 (2) (1949) 95–116.
[23] P. Hu, S.S.M. Chow, W. Cheong Lau, Secure friend discovery via privacy-preserving and decentralized community detection, 2014.
[24] T. Carnes, C. Nagarajan, S.M. Wild, A. Van Zuylen, Maximizing influence in a competitive social network: a follower's perspective, in: Proceedings of the Ninth International Conference on Electronic Commerce, ACM, 2007, pp. 351–360.
[25] A. Borodin, Y. Filmus, J. Oren, Threshold models for competitive influence in social networks, in: Internet and Network Economics, Springer, 2010, pp. 539–550.
[26] G. Peng, J. Mu, Technology adoption in online social networks, J. Prod. Innovation Manage. 28 (s1) (2011) 133–145.
[27] T. Hogg, G. Szabo, Diversity of user activity and content quality in online communities, in: Third International AAAI Conference on Weblogs and Social Media, 2009.
[28] W. Wood, T. Hayes, Social influence on consumer decisions: motives, modes, and consequences, J. Consum. Psychol. 22 (3) (2012) 324–328.
[29] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of *NP*-Completeness, W.H. Freeman, San Francisco, CA, 1979.
[30] L. Fortnow, The status of the P versus NP problem, Commun. ACM 52 (9) (2009) 78–86.
[31] R. Downey, M. Fellows, Parameterized Complexity, Springer, Berlin, 1999.
[32] A. Wigderson, P, NP and mathematics — A computational complexity perspective, in: Proceedings of ICM 2006: Volume I, EMS Publishing House, Zurich, 2007, pp. 665–712.
[33] R. Downey, M. Fellows, Fundamentals of Parameterized Complexity, Springer, Berlin, 2013.
[34] M.R. Mirsaleh, M.R. Meybodi, A michigan memetic algorithm for solving the community detection problem in complex network, Neurocomputing 214 (2016) 535–545.
[35] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, Distributed learning automata-based algorithm for community detection in complex networks, Int. J. Modern Phys. B 30 (8) (2016) 1650042.
[36] J. Xie, B.K. Szymanski, X. Liu, Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process, in: Proceedings of ICDM, IEEE, 2011, pp. 344–349.
[37] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.
[38] H.-C. Ou, C.-K. Chou, M.-S. Chen, Influence maximization for complementary goods: Why parties fail to cooperate? in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 1713–1722.
[39] C.V. Pham, D.K. Ha, D.Q. Ngo, Q.C. Vu, H.X. Hoang, A new viral marketing strategy with the competition in the large-scale online social networks, in: 2016 IEEE International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), IEEE, 2016, pp. 1–6.