



TI-SC: top-k influential nodes selection based on community detection and scoring criteria in social networks

Hamid Ahmadi Beni¹ · Asgarali Bouyer¹

Received: 25 June 2019 / Accepted: 1 February 2020 / Published online: 11 February 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

Influence maximization is a classic optimization problem to find a subset of seed nodes in a social network that has a maximum influence with respect to a propagation model. This problem suffers from the overlap of seed nodes and the lack of optimal selection of seed nodes. Kempe et al. have shown that this problem is an NP-hard problem, and the objective function is submodular. Therefore, some heuristic and greedy algorithms have been proposed to find a near-optimal solution. However, the greedy algorithm may not satisfy the accuracy of a given solution and high time-consuming problem. To overcome these problems, the TI-SC algorithm is proposed for the problem of influence maximization. The TI-SC algorithm selects the influential nodes by examining the relationships between the core nodes and the scoring ability of other nodes. After selecting each seed node, the scores are updated to reduce the overlap in selecting the seed nodes. This algorithm has efficient performance in high Rich-Club networks. The Rich-Club phenomenon causes overlapping of the influence spread among the seed nodes in most of the other methods so that the TI-SC algorithm reduces this overlapping. Furthermore, the discovered communities with low expansion are not considered in the seed node selection phase, and this is useful for reducing computational overhead. Experimental results on both synthetic and real datasets show that the proposed TI-SC algorithm significantly outperforms the state-of-the-art algorithms in terms of efficiency in both small and large-scale datasets.

Keywords Social network · Viral marketing · Influence maximization · Seed node · Community detection

1 Introduction

Social Network Analysis (SNA) studies the structure and information flow in the social network. The social network is a set of nodes and links with a specific type of relationships and interactions, such as friendship, like or dislike, love, etc. One of the most important issues in social networks is the investigation of finding influential nodes. If an influential node is identified individually without considering the position of other influential nodes in the network, it is defined an influential spreader detection problem (Berahmand et al. 2018, 2019). However, identifying the set of influential nodes with regard to the topological effects of these nodes in relation to each other is called the influence maximization problem (Samadi and Bouyer 2019).

An example of the applications of influence maximization is viral marketing. Viral marketing through social networks has now become imperative for maximizing the awareness of a new product and increase their revenue. In viral marketing, selecting the most influential people is more important. The goal of influence maximization is to find the best influential people that maximize diffusion. Diffusion is a process in which information on the network spread from a node to other nodes. In the diffusion process, complete influence time is very important. The complete influence time is the time duration that it takes to achieve complete influence (Ni et al. 2017).

. Due to their application in business, researchers have attracted influence maximization problem in recent years. Domingos and Richardson introduced the influence maximization problem for viral marketing (Domingos and Richardson 2001). Later on, Kempe et al. (2003) developed and formulated the influence maximization. According to their formulation, k seed nodes in a social network graph has best influence maximization if their influence spreading, the number of the activated nodes, is maximized based on the diffusion models of node activation. They considered two

✉ Asgarali Bouyer
a.bouyer@azaruniv.ac.ir; asgar.bouyer@gmail.com

Hamid Ahmadi Beni
h.ahmadi@azaruniv.ac.ir

¹ Azarbaijan Shahid Madani University, Tabriz, Iran

activation model, independent cascade, and linear threshold models (Domingos and Richardson 2001). As an example, influence maximization results from this fact that a commercial company provides a free sample of the product to a set of influential individuals to reach large advertising in a cascade manner. In both models, each node has two active and inactive states. In an active state, the intended node adopts a new product, idea, or information, while the inactive node does not adopt a new product, idea, or information. Each node in the independent cascade model has the probability p_{uv} , that correspond to the probability of extending of a node influences to other nodes. The activation process is done only once time for each node. If node v is not activated by node u , it will not try to activate u in later steps. The existence of a single chance of node activation is one of the features of this model. In the linear threshold model, each edge $e \in E$ contains an influence weight. Each node $v \in V$ has a threshold limit θ_v , which is selected with a uniform distribution within the range $[0,1]$. Each edge (u, v) has an influence weight $w(u, v)[0,1]$. The total weights of all input edges v are at most 1. Each inactive node v will be added to the total active nodes if $\sum_{u \in S_{t-1} \cap N^{in}(v)} w(u, v) > \theta_v$. It means that the total weights of the active neighbors is larger than θ_v . The influence maximization problem is an NP-hard and submodular for both IC and LT models. Kempe et al. proved that the function of the influence under both IC and LT models is monotone and submodular concerning the number of seed nodes (Kempe et al. 2003).

Recently, the community-based influence maximization algorithms are studied (Banerjee et al. 2019a, b; Bozorgi et al. 2017; Chen et al. 2014; Hosseini-Pozveh et al. 2017; Huang et al. 2019; Qiu et al. 2019; Shang et al. 2017; Wang et al. 2010). First a community detection algorithm is performed for extracting dense parts of network (Berahmand and Bouyer 2019). In the second phase, the proper seeds is selected from suitable communities. However, the existing community-based influence maximization algorithms demonstrate several major drawbacks: (a) They have no effective methods to reduce the search space the for selecting seed nodes in the large networks, (b) They suffer from the problems of the overlap of seed nodes, (c) They do not consider the role of core nodes in the influence spread. So, to solve these problems, this paper proposes a new algorithm, called TI-SC, to solve the problem of influence maximization under the IC model, with an emphasis on the time efficiency and spreading ability. The new algorithm is much more efficient in terms of time and influence spread. It can be categorized in community-based influence maximization methods. The TI-SC algorithm limits and controls the influence spread computation in discovered communities. It inspires the scoring criteria from the real world. The proposed algorithm can easily run in networks with millions of nodes and edges and solve the problem of traditional influence maximization algorithms. TI-SC algorithm comprises four phases: 1.

Community detection, 2. Merge of communities, 3. Selecting the first influential node (seed node), and 4. Updating the scoring criteria. In the first step of TI-SC algorithm the Louvain algorithm is used for detecting communities. After the community detection process, if there is relationship between the core nodes of communities, these communities merges. In the next step, the first seed node is selected according to the ability scoring of other nodes. Also, according topological structure of communities reduce the search space for selecting seed nodes. Finally, In this step, update scoring criteria to choose the remaining seed nodes.

To sum up, our major contributions in this algorithm are:

1. We study influence maximization problem under the background of community structure and reduce search space for selecting seed node.
2. We propose the scoring ability of other node which reduce the overlap of seed node.
3. We merge communities that have a similar information diffusion structure by the relationship between core nodes.
4. The experiment result on the synthetic and real-world networks show that the proposed algorithm TI-SC performs better than the base algorithms in term of influence spread. The TI-SC algorithm is more time-efficient than base algorithms.

The rest of the paper is organized as follows. Section 2 contains a valuable review of the literature, Sect. 3 provides a detailed description of the proposed method, Sect. 4 deals with the experimental evaluation for the proposed algorithm, and Sect. 5 draws the conclusion of the present study.

2 Related work

The problem of influence maximization has been studied based on the two important LT and IC models (Kempe et al. 2003). Even though some new models have been provided with regard to these two models in recent years. The IC model was first suggested and explored by Kempe et al. (2003). In an independent cascade model, each edge $(u, v) \in E$ includes the influence probability $p(u, v) \in [0,1]$. The influence probability represents the possibility of activation. The attempt for activation in the IC model is made once; that is, if the node v could not activate the node u , it does not have another chance to activate this node u once again. The termination condition is that no other nodes are activated at time t , and no changes are made in the set of active nodes. The proposed IC model by Kempe deals with positive beliefs and opinions, whereas the IC-N and IC-OC models are based on positive and negative opinions (Ma et al. 2015; Wang et al. 2016). The key difference in diffusion between IC model with IC-N and IC-OC is

that in standard IC model, with adding a new node to the seed set of influential nodes, the influence power of the seed set is increased. Nevertheless, in IC-N and IC-OC diffusion model, blocking a node with negative opinion may also help to influence spread. Other researchers suggested the LAIC model as another development of IC (Liu et al. 2012). For the LAIC model, when a node u is activated in step t , it activates inactive neighbors at step $t + \delta_t$ with the probability $\mathcal{P}_{uv} \mathcal{P}_u^{lat}(\delta_t)$, which δ_t is the delay effect and \mathcal{P}_u^{lat} is a distributed delay. The DICM model introduced for dynamic networks (Wang 2016). The only difference between IC and the DICM models is that the activated nodes can retry to activate an inactive node at a given time.

In the linear threshold model, each edge $(u, v) \in E$ has the influence weight $w(u, v) \in [0, 1]$, which is also shown with w_{uv} . These weights for each $v \in V$ are as follows: $\sum_{u \in N^{in}(v)} w(u, v) \leq 1$, where $N^{in}(v)$ is the input neighbors of node v . Each node v has a threshold θ_v , which is uniformly selected from the interval $[0, 1]$. With regard to the initial activation set S_0 , the active random set S_t for $t > 1$ is generated as follows: All activated nodes in step t will remain active. The node v is activated if the total weight of the active neighbors is at least greater than θ_v . In other words, $\sum_{u \in S_{t-1} \cap N^{in}(v)} w(u, v) > \theta_v$, then, v includes in the set S_t . Ruan et al. developed a new model based on the linear threshold model (Ruan et al. 2015). Moreover, other researcher provided the DLT model for dynamic networks (Wang 2016).

In recent years, the issue of influence maximization has been studied in two categories: greedy-based methods and heuristic methods. Heuristic methods can be examined in several sub-sections, including topological-based methods and community detection methods. We discuss these sections with more details below.

2.1 Greedy algorithm-based methods

The greedy algorithm has been put forward by Kempe et al. so that each node is initially added to the seed set, then k influential nodes with maximum diffusion are selected using the Monte Carlo simulation (Kempe et al. 2003). Although the greedy algorithm guarantees optimal approximation. If the set s^* has more than k members that maximize f -value, we have the following relation for $f(s)$:

$$f(s) \geq \left(1 - \frac{1}{e}\right)f(s^*) \quad (1)$$

However, the greedy algorithm is inefficient because it spends a lot of computational time to choose influential nodes. In the following, the CELF algorithm presented to improve the greedy algorithm and used lazy evaluations for improvement (Leskovec et al. 2007). Lazy evaluation is 700 times fast in the influence maximization problem due to avoid of unnecessary evaluations. However, the CELF algorithm has a high

running time due to the use of Monte Carlo simulation. So, the CELF++ algorithm improved lazy evaluation for the influence maximization problem (Goyal et al. 2011a). This is the modified CELF algorithm that targets decreasing the number of the quantifications of diffusion in computation. However, Memory usage in this algorithm is low, but it doesn't guarantee approximation. The NewGreedyIC algorithm presented to improve the greedy algorithm (Chen et al. 2009). This algorithm generates G' graph R times for each graph G , so that each edge with the probability $1-p$ is eliminated, and each edge with p probability remains in the graph. Then, it calculates the available nodes from each node and chooses k influential node. So, the NewGreedyIC algorithm is much faster than the greedy algorithm but its influence spread is less than the greedy algorithm. The staticGreedy algorithm developed to improved the Greedy algorithm that keeps submodularity property during the selection of the seed nodes, which entails two static snapshot and greedy selection (Cheng et al. 2013). Also, the StaticGreedyDU algorithm was proposed, which is 2-7 times faster than the staticGreedy algorithm. The running time of the StaticGreedy algorithm depends on the number of seed nodes. The LUGreedy algorithm is based on uncertainty that this algorithm uses a parametric space $\theta = x_{e \in E}[l_e, r_e]$ for the probable influence on the edges (Chen et al. 2016). So, this algorithm with uniform sampling and adaptive sampling methods to effectively reduce the uncertainty on parameters and improve the robustness of the influence maximization task but parameter uncertainty may greatly affect influence maximization performance.

2.2 Heuristic methods

Heuristic methods are based on two topology-based and community detection methods.

2.2.1 Topology-based methods

Topology-based algorithms are faster than greedy algorithms. Some topology-based method gives special importance for a node's degree in selecting k -influential nodes (Samadi and Bouyer 2019). For example, The High-Degree algorithm arranges nodes by a degree in descending order and selects k nodes with the highest degree as the influential ones (Chen et al. 2009). However, this algorithm does not have acceptable performance, and it is not suitable for an influence maximization problem. So, the Distance algorithm is a distance-based algorithm to find influential nodes (Chen et al. 2009). With regard to this algorithm, the graph nodes are arranged ascendingly based on the mean distance, and then k influential nodes are chosen. As such, this algorithm does not perform well at running time and influence spread. In the following, the singleDiscount and DegreeDiscount algorithms presented to improve the High-Degree algorithm (Chen et al. 2009). In the

singleDiscount algorithm, k influential node is chosen according to the degree of a node that degree of each node is reduced by the number of seed nodes in the neighbor of the node. Consequently, the running time of the SingleDiscount algorithm is much shorter than that of algorithm Distance, and also the influence spread is longer than that of algorithm Distance, but the performance of the SingleDiscount algorithm is needed to improve the influence spread. In the DegreeDiscount algorithm, k influential nodes are chosen according to degree discount and seed nodes in the neighborhood node. As such, algorithm DegreeDiscount has a low running time, but it does not guarantee approximation. The k -core algorithm was presented with respect to inner shells (Kitsak et al. 2010). Using this algorithm, core and periphery nodes are initially detected in the graph. According to the hypothesis of this algorithm, nodes in the core enjoy a higher influence spread. Hence, algorithm k -core has a low running time, but this algorithm has a low influence spread. The SIMPATH algorithm presented to improved the CELF algorithm (Goyal et al. 2011b). This algorithm contributes to the counting of simple paths in order to calculate the influence path; however, counting all the simple paths is an NP-hard problem. Therefore, the parameter η is used to limit influence computations in the neighborhood of the node. However, memory usage and running time in this algorithm are low, but it does not guarantee approximation. In the following, the MIA algorithm is another method that uses the structure of a graph, such as a tree, to determine influence and prevent the Monte Carlo simulation (Wang et al. 2012). It also uses a tree structure and criterion θ to calculate local influence. However, memory usage in this algorithm is high, but the running time is very low. IRIE algorithm hat has been developed by Jung et al. is more appropriate than the MIA algorithm in terms of runtime and optimal use of memory (Jung et al. 2012). Lawyer proposed a measure, named the expected force that is a node property derived from local topology (Lawyer 2015). The CI algorithm presented based on localization measures (Morone et al. 2016). This algorithm calculates the influence spread for each node in radius l . It is useful for the Spars graph, but the parameter l may greatly affect influence maximization performance. Li et al. developed a new method to measure the importance of a node using its location in a network and dynamic activities (Li et al. 2018). The LIR algorithm presented to avoid the Rich-Club Phenomenon (Liu et al. 2017). According to this algorithm, influential nodes are selected using the degree of neighbor nodes. However, this algorithm is very fast, but it does not guarantee approximation. The HybridRank algorithm presented based on two centralities of eigenvector and coreness (Ahajjam and Badir 2018). However, it should be noted in the HybridRank algorithm that if the node with the highest HC is in the neighborhood of the seed node, it avoids selecting that node as the next seed. The MATI algorithm is an extension of the SIMPATH algorithm (Rossi et al. 2018). It is an influence

maximization algorithm under both the linear threshold and independent cascade modes. Therefore, the MATI algorithm is much faster than the SIMPATH algorithm. The HO centrality is derived by two factors and one topology factor. Activity rank and spread rank are important in the HO centrality (Xin et al. 2019). Activity rank and spread rank are factors in the HO centrality that characterize the activity level and spread ability, respectively.

2.2.2 Community detection-based methods

In community-based algorithms, community detection is performed as the first step to the influence maximization problem. The CGA algorithm presented on mobile social networks, which involves two steps: 1. Detection of communities using information diffusion, 2. Dynamic programming (Wang et al. 2010). This algorithm provides approximation guarantees for influence maximization. Moreover, this algorithm is more than an order of magnitudes faster than the Greedy algorithm for finding top- K influential nodes on a large real-world mobile social network. But this algorithm does not limit the search space for seed nodes. Other researcher presented the CIM algorithm that this algorithm contains three steps. 1. Community detection is detected by the similarity criterion of communities. 2. Generating candidate nodes. 3. Selecting final seed nodes (Chen et al. 2014). This algorithm efficiently selects the number of seeds to maximize influence spreads. However, in this algorithm, the search space for seed nodes selection is not efficiently limited. Borgs et al. used community detection and the red-black tree to find influential nodes (Borgs et al. 2014). This algorithm does not efficiently select the number of seeds to maximize influence spreads. Moreover, this algorithm has a low running time. Hosseini-Pozveh et al. developed a community detection-based algorithm with three steps: 1. Community detection, 2. Selection of the seed nodes, and 3. Identification of the target set (Hosseini-Pozveh et al. 2017). Shang et al. provided the CoFIM algorithm that first examines diffusion and influence among the communities, and then explores diffusion within each community (Shang et al. 2017). Thus, it is much more efficient in terms of both time and memory usage, but this algorithm suffers from the overlap of seed nodes. The CI2 algorithm suggested based on the new DCM model, which chooses influential nodes based on the graph structure (Bozorgi et al. 2017). Seed nodes are optimally selected in this algorithm, but the search space for seed nodes selection is not limited. Other researcher proposed the ComBIM algorithm (Banerjee et al. 2019a). With regard to this algorithm, the community budget is determined according to the value of each node. Then, based on the budget transfer in each community and the degree of nodes in the community, the influential node is selected. Therefore, this algorithm is very fast, but

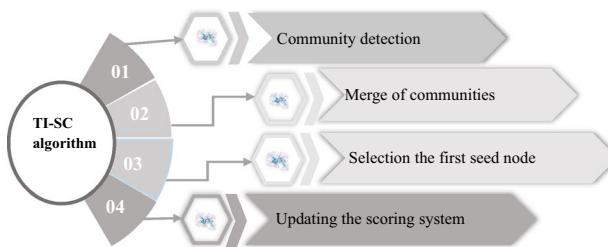


Fig. 1 Overview of the TI-SC algorithm

the parameter budget may greatly affect influence maximization and running time performance. The CTIM algorithm presented to improve the Greedy algorithm (Huang et al. 2019). The algorithm initially uses the Comprehensive Latent Variable model to obtain favorite topics and distribution of the community members from each user. Then, it uses favorite topics to gain influence power in each community. It also computes the user to influential user power using the community-to-community influential power and distribution of the members of each user's community. Next, it chooses k influential nodes using the Divide-and-conquer algorithm. The PHG algorithm presented to improve the Greedy algorithm (Qiu et al. 2019). This algorithm finds key nodes in each community to construct a candidate set by detecting community structure. The search space for seed nodes selection is not limited in this algorithm. Therefore, the running time increases as the candidate nodes increase. Other researcher suggest an original approach to influence maximization using a fuzzy-logic based model and community detection (Atif et al. 2019). This algorithm is efficient and scalable, but accuracy is still an issue.

3 The proposed method

In this section, we present a detailed description of our proposed algorithm (TI-SC) that has 4 phases: (a) community detection, (b) community merging, (c) selecting the primary seed node, and d) updating the scoring criteria. An Overview of the TI-SC algorithm is shown in Fig. 1.

3.1 Community detection

Detecting the structure of the community helps to solve the influence maximization problem. A community is a group of nodes that are strongly linked to each other (Gmati et al. 2018). Social networks are composed of several large and small communities. Discovered communities play a determinant role in the influence maximization problem.

Communities cause the seed nodes with maximum diffusion are chosen for each social network. In the first step of TI-SC algorithm for each graph $G = (V, E)$, the Louvain algorithm is used for detecting communities $C = \{C_1, C_2, C_3, \dots, C_n\}$. Algorithm Louvain [1] is one of the optimal algorithms that attempt to maximize the modularity function using Eq. (2). The algorithm is implemented in two steps:

1. At first, it uses the modularity function ΔQ for obtaining communities for each graph.

$$\begin{aligned} \Delta Q = & \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] \\ & - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \end{aligned} \quad (2)$$

- where \sum_{in} is the sum of all the weights of the links inside the community i , \sum_{tot} represents the sum of all the weights of the links to nodes in the community i , k_i is the total weight of the nodes i , $k_{i,in}$ indicates the total weight of the link from the node i to the nodes within C , and m is the total weight of all the links in the network.
2. Then, it combines the obtained communities with each other to maximize the modularity function. These steps continue until the algorithm achieves the maximum value of the modularity function, and other communities do not change.

3.2 Merge of communities

After the community detection process, suppose that we have n communities $C = \{C_1, C_2, C_3, \dots, C_n\}$, where for each community $C_i = (V_c, E_c)$, $v \in V_c$, and $e \in E_c$, and v is the number of the nodes within the community C_i and e represents the number of the edges within the community C_i . Now, the k -core algorithm [13, 33] is used to determine each node position for each community C_i . This algorithm assigns integer K_s to each node, and each node position is determined by the successive layers in the graph. A small amount of K_s is defined for marginal nodes, and the core nodes have the largest value of K_s . In the k -core algorithm, the first-order nodes are initially deleted and included in shell 1. Of course, when the first-order nodes are deleted, it is possible to create new first-order nodes so that these nodes are also included in the shell 1. Therefore, the $K_s=1$ is assigned to the nodes in shell 1. Then, the second-order nodes are placed in shell 2, and $K_s=2$ is allocated to the nodes in shell 2. However, in a case of deletion of the second-order nodes, new first- and second-order nodes may be created so that the new nodes are included in their shells. The same procedure continues for the remaining nodes so that all nodes of the graph are assigned to their K_s . The core nodes (with

highest Ks value) have the maximum influence compared to the margin nodes. If $V_{co} = \{v_{co1}, v_{co2}, v_{co3}, \dots, v_{con}\}$ is the core nodes within the community; therefore, $\Gamma(V_{co})$ represents the neighbors of the core nodes within a community.

Now, we examine the two communities C_i and C_j . The core nodes for these two communities are $V_{coi} = \{v_{co1}, v_{co2}, v_{co3}, \dots, v_{con}\}$ and $V_{coj} = \{v_{co1}, v_{co2}, v_{co3}, \dots, v_{con}\}$. If $\Gamma(V_{coi}) = V_{coj}$ (or vice versa $\Gamma(V_{coj}) = V_{coi}$), the communities C_i and C_j become $C_i \cup C_j$. In other words, if there is a relationship between the core nodes of the communities, the maximum influence spread will be observed among these communities. Moreover, such communities have a similar information diffusion structure. For this reason, these communities are combined.

Figure 2a, c depict an example of the two Ego-Facebook network and Santa Fe Institute Collaboration network before the merge of communities step so that nodes with similar colors show that the nodes are within the same community. As seen in Fig. 2, algorithm 1 has been used for discovering 14 communities for the Ego-Facebook network and 7 communities for Santa Fe Institute Collaboration network before the merge of communities step. Figure 2b, d indicate the merge of communities. Before the merge of communities step, four communities have been merged in the Ego-Facebook network and Santa Fe Institute Collaboration network.

Algorithm 1: merge of the community (G)

Input: Network G(V,E)
Output: Union of community
1: initialize $C \rightarrow \emptyset$;
// phase (i): community detection
2: $C = \{C_1, C_2, \dots, C_h\}$ // Using algorithm Louvain
// phase (ii):union of communities
3:**for each** c_i in C **do**
4: find V_{co} //Using algorithm k-core
5:**end for**
6: **for each** c_i in C **do**
7: **for each** c_j in C do
8: **if** $\Gamma(V_{coi}) = V_{coj} \mid \Gamma(V_{coj}) = V_{coi}$ **then**
9: $c_c = c_i \cup c_j$
10: $C_c.add(c_c)$
11: **else:**
12: $C_c.add(c_i, c_j)$
13: **end if**
14: **end for**
15:**end for**
16: return : $C_c = \{c_{c1}, c_{c2}, \dots, c_{cn}\}$

Algorithm 1 has been provided for step 1,2 of the algorithm TI-SC. This algorithm aims to achieve communities and a merge of communities. In line 1–2 of algorithm 1, communities are discovered via algorithm Louvain and the

core nodes V_{co} for each c_i community is chosen by algorithm K-core in lines 3–4. Then, the neighborhood of the core nodes is compared in lines 6–12. If $\Gamma(V_{coi}) = V_{coj}$ (or vice versa $\Gamma(V_{coj}) = V_{coi}$), both communities are combined with regard to line 9.

3.3 Selection the first seed node

At this phase of the TI-SC algorithm, the first seed node is selected according to the scoring criteria that its idea is derived from the real world. In the real world, scores of people who are in the one-hop (first-level neighbors) and two-hop (second-level neighbors) environments of the person A have high importance because the set of people who are in the one-hop and two-hop environments has a very high knowledge about person A. Therefore, other people who do not know person A trust the scores given by the set of one-hop and two-hop people, and so converge their scores to the scores given by set of one-hop and two-hop people. In social networks, as in the real world, it is very important to try to achieve the highest score, such as the presidential elections. Therefore, people in one-hop and two-hop levels tend to converge the scores of their neighborhood to their own score. In the TI-SC algorithm, the score of each node u is calculated concerning the nodes in one-hop and two-hop distance from the node u. Moreover, it should be noted that each node in the one-hop and two-hop environments can only give one score to the intended node. Nodes with the greatest scores are chosen as the influential ones in the graph network. Accordingly, the node scoring criteria is locally computed in each community by this phase. The node with the highest score is chosen as the first seed node. If $C_c = \{c_{c1}, c_{c2}, \dots, c_{cn}\}$ is the merge of communities through algorithm 1, $v \in \Gamma_{c_c}(u)$ shows neighbors of node u within the community c_{ci} . According to the following relation, each node v gives a score to the node u, if both of them is in the same community:

$$score_{v \rightarrow u} = p \left(\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)} \right) \quad (3)$$

In addition, in the real world, individuals' opinions depend on a random parameter. Therefore, p is considered as a random parameter, which is a random number with a uniform distribution between [0,1]. The environment $N_u^{(1)} \setminus N_o^{(1)}$ around node u is considered as the one-hop, which contains only the set of nodes v as the scorer nodes. $DN_u^{(1)}$ is the total number of the edges inside $N_u^{(1)} \setminus N_o^{(1)}$ environment and external edges from the environment $N_u^{(1)} \setminus N_o^{(1)}$. Then, the environment $N_u^{(2)} \setminus N_o^{(2)}$ is considered all nodes around the set nodes v, except node u. $DNN_u^{(1)}$ is the total number of the edges in $N_u^{(2)} \setminus N_o^{(2)}$ and the external edges from the

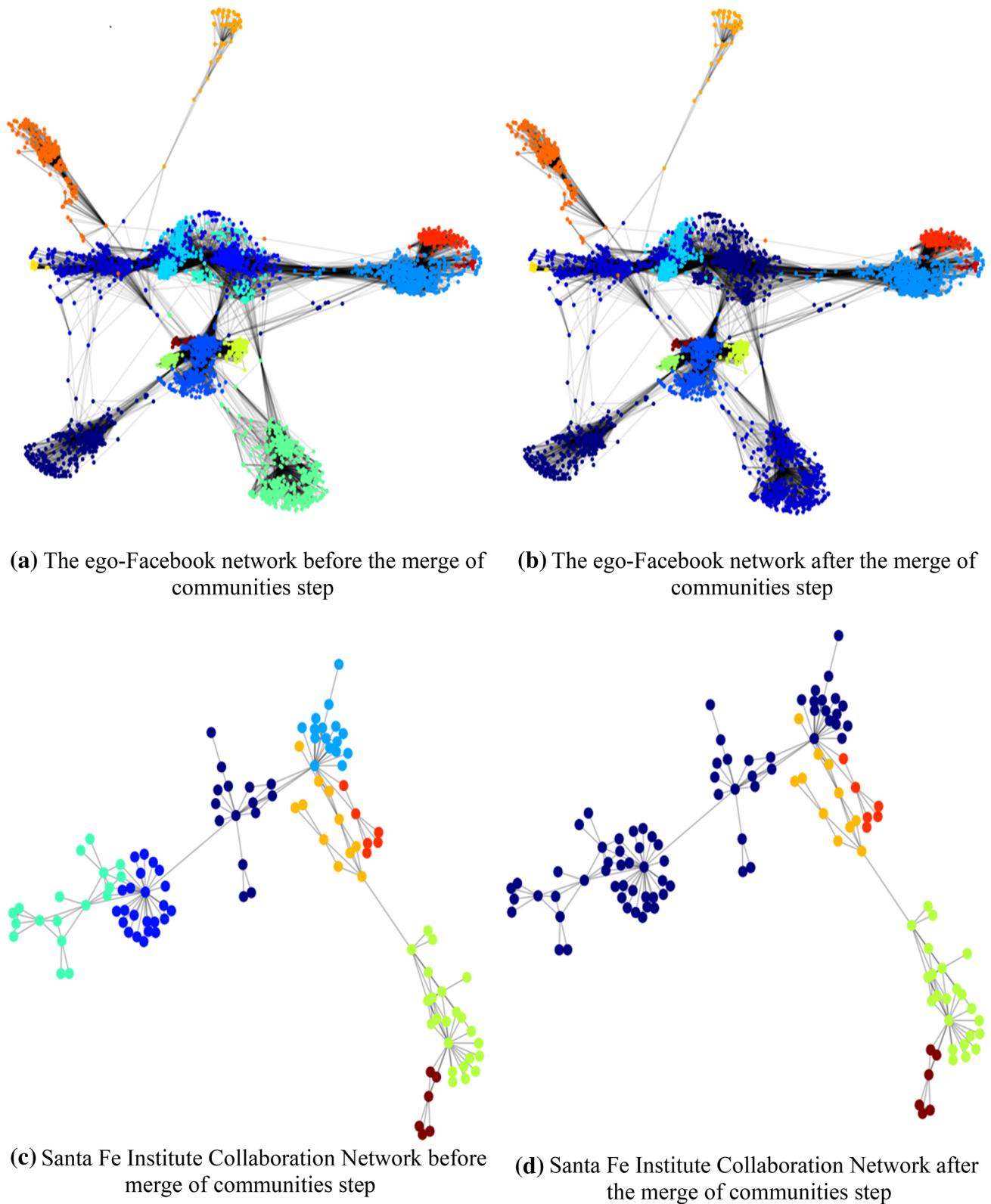


Fig. 2 The ego-Facebook network and Santa Fe Institute Collaboration network before and after the merge of communities step. Nodes with similar colors show that they are within the same community

environment $N_u^{(2)} \setminus N_o^{(2)}$. If $\Gamma_{c_c}(u) > 1$, then the Score for node u is defined as:

$$\text{score} = \sum_{v \in \Gamma_{c_c}(u)} \text{score}_{v \rightarrow u} \quad (4)$$

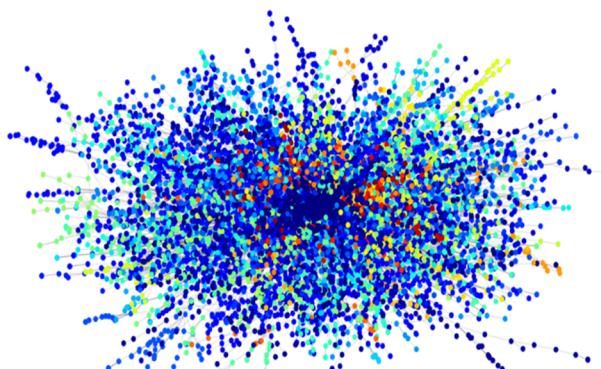
In Eq. 4, the given scores by each neighbor of node u are calculated for each $v \in \Gamma_{c_c}(u)$. When the scores have been calculated for each node within the community, the node with the highest score in each community is chosen and included in a top-score array. Then, the influence spread $\sigma(score_{top[i]})$ is calculated for each top-score array and the highest node with the maximum $\sigma(score_{top[i]})$ is chosen as the first seed node. In other words, it is formulated in Eq. 5.

$$S = S \cup \{\text{argmax}_{u \in V \setminus S} \{\sigma(score_{top[i]})\}\} \quad (5)$$

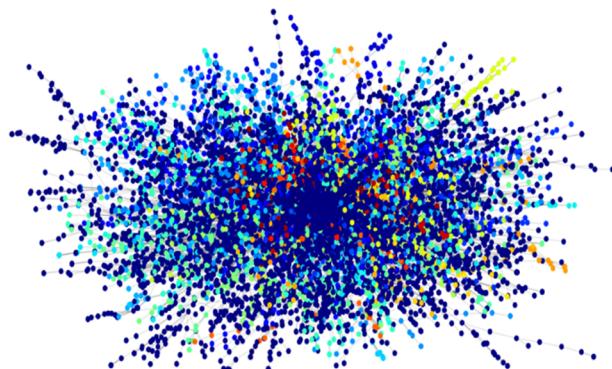
However, since there is a large number of communities in big datasets and the comparisons of the influence spread $\sigma(score_{top[i]})$ are very time-consuming; thus, the criterion θ has been used to control and eliminate additional comparisons. Experiments conducted on different datasets showed that most of the nodes have very low influence spreading in communities. These nodes lengthen the arrays and cause high computational overhead. With regard to our experiments, these communities have a certain topological structure. According to this topological structure, the two criteria θ, φ , are defined. The criterion φ is calculated according to the following relation:

$$\varphi = \frac{n_{NC}}{n_{EC} + 1} \quad (6)$$

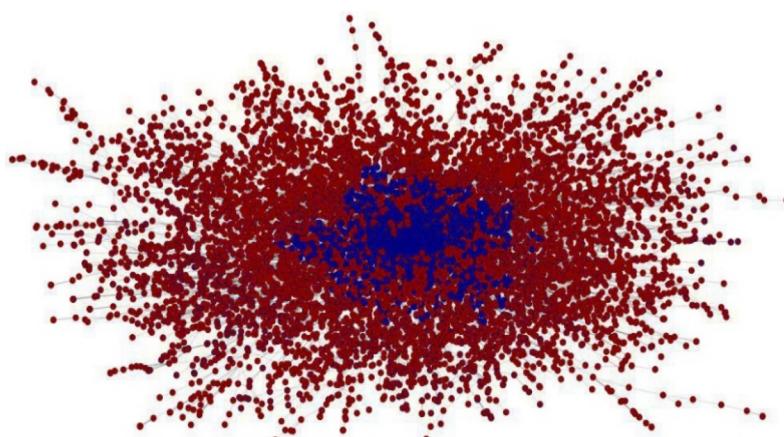
Where n_{NC} and n_{EC} represent the number of nodes and edges in community c_i , respectively. It is possible that



(a) Phase 1: community detection step



(b) Phase 2: the merge of communities step



(c) Phase 3: first seed node selection and detection of unsuitable communities in the influence spread calculations using θ, φ

Fig. 3 The effect of two criteria θ, φ on the M-FO115 network. **a** Community detection step: the nodes with similar colors are in the same community. **b** The merge of communities, and **c** shows the red

nodes in unsuitable communities with low influence spread, and blue nodes are in suitable communities for influence spread calculations (color figure online)

$n_{EC} = 0$ for some communities (just one node). The value of 1 is added to the denominator so that φ does not proceed to the infinity. If $\varphi > \theta$ for the community c_i , the nodes of the community c_i are identified as the nodes with low influence spread and the community c_i is not participated in the comparisons of influence spread of $\sigma(S \cup v_{ms})$.

Moreover, communities are not updated. The value of θ has been set in the experiments section. Figure 3 depicts the effect of the two criteria θ, φ in the fire forest model. Communities have been specified with the Louvain algorithm in Fig. 3a, in which the nodes with similar colors are in the same community. Figure 3b indicates the merge of communities in the fire forest model. Figure 3c portrays the effect of the two criteria θ, φ . Red nodes are in the communities that are not suitable for influence spread and cause computational overhead. Blue nodes are suitable for calculations of influence spread. As seen in Fig. 3c, numerous red nodes (nodes with low influence) are in the network that only causes computational overhead in the influence spread process. Consequently, these red nodes are ignored by the TI-SC algorithm that has significant improvement in the running time of the algorithm.

Based on the above discussion for phase 3, we have proposed Algorithm 2 for choosing the first seed node. In algorithm 2, a set of seed nodes are defined as an array, titled S in line 1. If the condition $\varphi < \theta$ is established in line 3, it is participated in the comparisons of influence spread, and the score calculation is performed for it. For line 4–12, the Score is calculated for each node u in the community c_{ci} , and the score calculated for each node u in the community c_{ci} is added to the SC array. In line 13–18, the maximum score is calculated for each community c_{ci} . In line 19, the node with the maximum score for each community is added to the top-score array. In line 22–30,

the influence spread is calculated for each node in the top-score array via the Monte-Carlo simulation and $R = 1000$. In line 28, the amount of influence spread calculated for each node in the top-score array and it is added to the Inf array. Then, in line 31–36, the maximum influence spread is selected from the top-score array. $Len(Inf)$ is the length of the Inf array. In line 37, a node with maximum influence spread is added to the seed set.

3.4 Updating the scoring criteria

In the real world, if the person A chooses the individual B as the most influential and important person via scoring, he/she definitely given a lower priority to the individual C as the next important person. When the seed node is selected, some changes should be made in the scoring criteria for choosing the next seed nodes. Assume that node v is a neighborhood of the first seed node. In this case, node v gives lower scores to its other neighborhood nodes for choosing the second, third, and... seed nodes. Since it gives higher priority to the first seed node, therefore it rationally gives a lower score to other its neighbors in the next steps. Accordingly, each node v is recorded its scores for the seed node, and it should update its scoring criteria to choose the remaining seed nodes. There is a weak relationship between the selection of the next seed nodes and Eq. (3). The algorithm TI-SC solves the problem of the Rich-Club phenomenon well. The Rich Club phenomenon causes overlapping of the influence spread among the seed nodes so that the algorithm TI-SC reduces the overlapping. In other words, if a node is selected as a seed node, it means that all one-hope and two-hope nodes do not potential to choose as seed nodes due to assign low scores by their neighbors.

Table 1 Summary of real-world datasets and Synthetic networks

Dataset	DBLP	Amazon	Ego-Facebook	Email	M-Fo115	M-Fo120
Node	317 k	335 k	4 k	1 k	10 k	10 k
Edge	1 M	926 k	88 k	5 k	23 k	25 k
Max Degree	343	549	1045	71	155	229
Min Degree	1	1	1	1	1	1
Max community size	22,263	12,369	548	209	1315	1645
Min community size	5	13	19	25	2	2
Parameter θ	0.7	0.4	0.09	0.3	0.4	0.3

Algorithm 2: Select first seed (G)

Input: $C_c = \{c_{c1}, c_{c2}, \dots, c_{cn}\}$

Output: Select first seed

// SC is list of Score for each node in community.

// Score_top is list of max score.

// Inf is list of Influence for each node with max score.

// S is seed set.

- 1: **initialize** $S \rightarrow \emptyset$;
- 2: **for each** c_{ci} in C **do**
- 3: **if** $\varphi < \theta$:
- 4: **for each** $v \in \Gamma_{c_i}(u)$ in c_{ci} **do**
- 5: **if** $\Gamma_{c_i}(u) = 1$:
- 6: $score_{v \rightarrow u} = p(\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)})$
- 7: $SC.add(score)$
- 8: **end if**
- 9: **if** $\Gamma_{c_i}(u) > 1$:
- 10: $score = \sum_{v \in \Gamma_{c_i}(u)} score_v$
- 11: $SC.add(score)$
- 12: **end if**
- 13: **for each** $score$ in SC **do**
- 14: $max = score[0]$
- 15: **if** $score[i] > max$:
- 16: $max = score[i]$
- 17: **end if**
- 18: **end for**
- 19: $Score_top.add(u)$ // u is node with max score in the community.
- 20: **end for**
- 21: **end for**
- 22: **for each** $score$ in $Score_top$ **do**
- 23: **for** $j=1$ to R **do**
- 24: **count** $\leftarrow 0$
- 25: simulate diffusion process on graph G with $Score_top$
- 26: $n_a \leftarrow$ the number of activate nodes after the diffusion ends
- 27: $count \leftarrow count + n_a$
- 28: $Inf.add(count/R)$
- 29: **end for**
- 30: **end for**
- 31: **for** i in $len(Inf)$ **do**
- 32: $max = inf[0]$
- 33: **if** $Inf[i] > max$:
- 34: $max = Inf[i]$
- 35: **end if**
- 36: **end for**
- 37: $S.add(u)$ // u is node with max Inf .
- 38: **return** S

At this step, the seed node does not give a score to any of its neighboring nodes. If $v \in \Gamma_S(S)$ are the neighbors of the seed set, then the scores that each node v gives to $u \in \Gamma(v)/S$ is updated by Eq. 7 as follows:

$$score_{v \rightarrow u} = p \left(\left(\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} - \frac{1}{\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} + \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)}} \right) \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)} \right) \quad (7)$$

In Eq. 7, it should be noted that if a seed node exists in the environment $N_u^{(1)} \setminus N_o^{(1)}$ and $N_u^{(2)} \setminus N_o^{(2)}$, then the value of

$DNN_u^{(2)}$ and $DN_u^{(1)}$ in the equation are 0. After updating each node in the community c_i , the node with the highest score in each community is selected. Then, the influence spread of $\sigma(S \cup v_{ms})$ for each community is calculated by the Monte Carlo simulation and $R = 1000$, and the node v_{ms} with the highest $\sigma(S \cup v_{ms})$ is identified as seed among all communities and is added to the S set. However, since the process of seed selection is very time-consuming, the parameters θ and φ is used to eliminate additional comparisons.

Algorithm 3: Update scoring criteria (G)

Input: $C_c = \{c_{c1}, c_{c2}, \dots, c_{cn}\}$, S
Output: Seed set

- 1:**for** $j=1$ **to** k **do**
- 2: **for each** c_{ci} **in** C_c **do**
- 3: **if** $\varphi < \theta$:
- 4: **for each** $v \in \Gamma_S(S)$ **in** c_{ci} **do**
- 5: $score_{v \rightarrow u} = p((\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} - \frac{1}{\sum_{N_u^{(1)} \setminus N_o^{(1)}} DN_u^{(1)} + \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)}}) \sum_{N_u^{(2)} \setminus N_o^{(2)}} DNN_u^{(2)})$
- 6: $SC.add(score)$
- 7: **end for**
- 8: **for each** $score$ **in** SC **do**
- 9: $max = score[0]$
- 10: **if** $score[i] > max$:
- 11: $max = score[i]$
- 12: **end if**
- 13: **end for**
- 14: $I_{uc_{ci}} = \sigma_{c_{ci}}(S \cup \{u\})$ // u is node with max score.
- 15: **end if**
- 16: **end for**
- 17: $S = S \cup \{\text{argmax}_{u \in V \setminus S} \{I_{uc_{ci}}\}\}$
- 18:**end for**
- 19:**return** S

Algorithm 3 is provided with regard to the above discussion for step 4 of the algorithm TI-SC, which aims to update the scoring criteria and choose the seed nodes. S and $C_c = \{c_{c1}, c_{c2}, \dots, c_{cn}\}$ are the inputs of the algorithm 3, which is initially $|S| = 1$ and the obtained seed node from algorithm 2 are included to S . According to experimental evaluations, k is set to value 30 in line 1. Line 2–16 is implemented for each c_{ci} community, so that if the condition $\varphi < \theta$ is established in line 3, it will be participated in the comparisons of influence spread, and the score updates will be performed for it. In line 4–13, the score is updated for each community, and the node with the maximum score is chosen for each community. Then, in line 14 for each c_{ci}

community, the influence spread of the node with the highest score is calculated along with the set S using the Monte Carlo simulation and $R = 1000$. In line 17, the value of $I_{uc_{ci}}$ obtained for each community is compared, and the node u with the highest value of $I_{uc_{ci}}$ is identified as the seed node and is added to the S set.

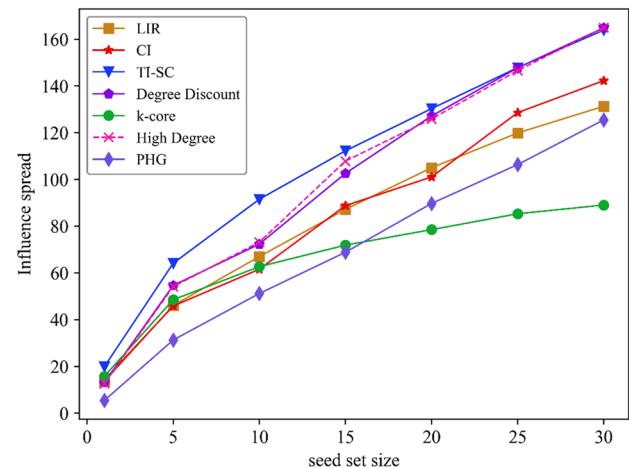


Fig. 4 Influence spreads of different algorithms on the DBLP

3.5 Analyzing of time complexity

In the analyzing of time complexity, n is the number of nodes in the network, m is the number of edges in the network, n' is the number of nodes in communities which are not identified as the low influential nodes spread; And these communities are participated in comparing and selecting the influential spreaders for $\sigma(S \cup v_{ms})$, and k is the number of seed nodes. For the TI-SC algorithm, the time complexity is separately computed in four parts: At first, the time complexity of Louvain algorithm is $O(n \log n)$. At the second, the time complexity of community merging step is $O(m)$. For selecting the first seed node, the required computational time is $O(Rn)$ where R denotes the number of Monte Carlo simulations. Fourth, the time complexity of updating the scoring criteria and selecting another seed node is $O((k-1)Rn')$. Consequently, the total time complexity is: $O(n \log n + m + Rn + (k-1)Rn') \approx O(m)$.

4 Experimental results and analysis

We first introduce the datasets, baseline methods, and evaluation metrics, then conduct experiments to evaluate our methods about the scope of information propagation on these datasets.

4.1 Dataset

In order to verify whether the TI-SC algorithm is better than other compared baseline methods, we use both real networks and synthetic networks to evaluate the performance.

4.1.1 Real-world networks

We first evaluate the performance of our community-based influence maximization algorithm on four real-world datasets. The largest datasets (Amazon, DBLP) contains 300 thousand nodes and about 1 million edges. Two other data (Ego-Facebook, Email) are in medium size. Three datasets (Amazon, DBLP, and Ego-facebook) are downloaded from the SNAP website,¹ and the Email dataset is downloaded from the KONECT website.² Table 1 summaries the statistical properties of these datasets.

- **DBLP** The DBLP online library is a large list of papers in computer science(Yang and Leskovec 2015). The data-

set provides a co-authorship network among research workers. If two authors have collaborated on at least one paper, an edge is created between them that this edge is undirected. The network consists of 317 K nodes and 1 M edges.

- **Amazon** The Amazon dataset was collected by crawling the Amazon website (Yang and Leskovec 2015). If a product i is frequently co-purchased with product j , then the network contains an undirected edge between them. The network contains 335 K nodes and 926 K edges.
- **Ego-Facebook** This dataset consists of “friends’ lists” from Facebook (Leskovec and Mcauley 2012). The dataset includes profiles, circles, and ego networks, and it contains 4 K nodes and 88 K edges.
- **Email** The email dataset is a network of University at Rovira i Virgili (URV), including faculty, researchers, technicians, managers, administrators, and graduate students (Guimera et al. 2003). Each edge represents that at least one email was sent. The network contains 1 K nodes and 5 K edges.

4.1.2 Synthetic networks

We also evaluate the performance of our TI-SC algorithm through synthetic networks. We use the forest fire model (Barabási and Albert 1999) to synthetic networks, due to its following advantages:

1. Existing the community structures in the forest fire model: This model can generate networks with configurable ground truth community structure.
2. Power-law degree distribution: The forest fire model generates networks whose degree follows a power-law distribution. The power-law distribution is a common property shared by most real-world networks, making our results more practicable. In the Forest Fire model, the network becomes denser with increasing mean degree; in other words, the larger the network grows, the smaller the network diameter.

A random graph M-Fo115 generated from the forest fire model with connecting probability 0.115 that it contains 10 K nodes and 23 K edges and also a random graph M-Fo120 generated from the forest fire model with connecting probability 0.120 that it contains 10 K nodes and 29 K edges.

4.2 Baseline algorithms

We compare the performance of our TI-SC algorithm with six baseline algorithms, which include three state-of-the-art algorithms having been successfully applied on large-scale networks and one node centrality based heuristic

¹ <http://snap.stanford.edu/data/>.

² <http://konect.cc/>.

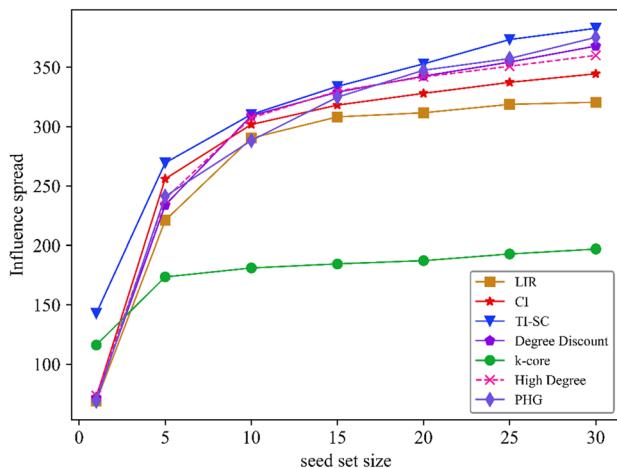


Fig. 5 Influence spreads of different algorithms on the Ego-facebook

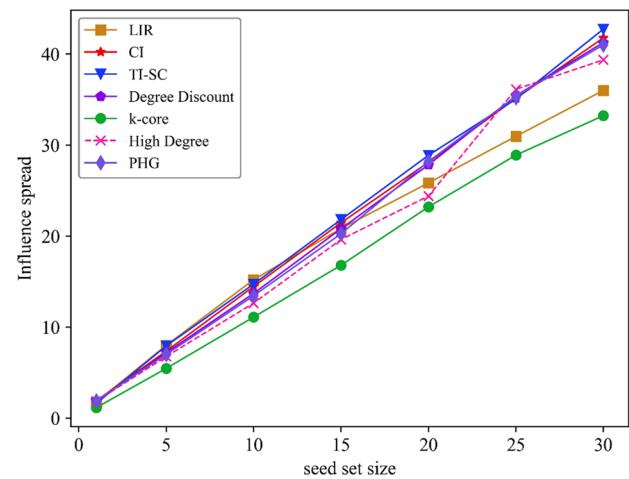


Fig. 7 Influence spreads of different algorithms on the Email

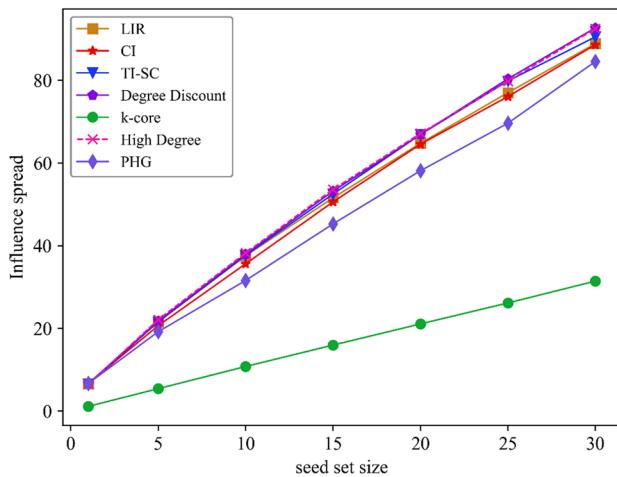


Fig. 6 Influence spreads of different algorithms on the Amazon

algorithms. We do not use the general greedy algorithm due to its extremely low time efficiency in handling large-scale networks.

K-core algorithm The k-core algorithm was presented with respect to inner shells(Ahajjam and Badir 2018). With regard to this algorithm, core and periphery nodes are detected in the graph. According to the hypothesis of the algorithm, nodes in the core mostly have higher influence spread than others.

DegreeDiscount In the DegreeDiscount algorithm, k influential nodes are chosen according to the equation $d_v - 2t_v - (d_v - t_v)t_v p$, where d_v and t_v respectively are the degree of the node v and the number of the seed nodes in the neighborhood of the node v (Chen et al. 2009). P is the influence probability from the node u to node v .

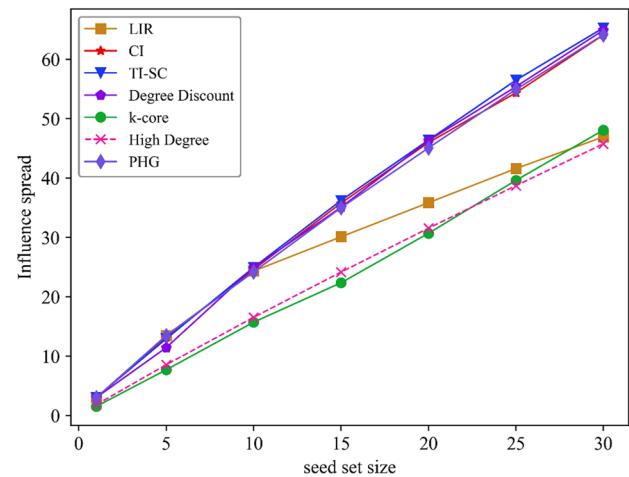


Fig. 8 Influence spreads of different algorithms on the M-FO115

Collective Influence (CI) algorithm The CI algorithm presented based on localization measures (Morone et al. 2016). In the first step of this algorithm, the value of $CI_l(i) = (k_i - 1) \sum_{j \in \partial B(i,l)} (k_j - 1)$ is calculated for each node, where k_i is the degree of the node i , and $\partial B(i,l)$ is a circle with radius l and its center is the node i . With regard to the values obtained for each node in the graph, the max-heap tree is created in the second step. In the third step, the node with the maximum value (root node) of the max-heap tree is eliminated. Then, the heapify algorithm is called to reconfigure. In the fourth step, the CI score for neighbors of the eliminated node is updated, and the second to fourth steps of algorithm proceed as long as the value of $\lambda(l,q) = (\frac{\sum_i CI_l(i)}{N(K)})^{1/(l+1)}$ equals 1. (K) is the mean degree.

LIR The algorithm LIR was presented to avoid the Rich-Club Phenomenon (Liu et al. 2017). According to this

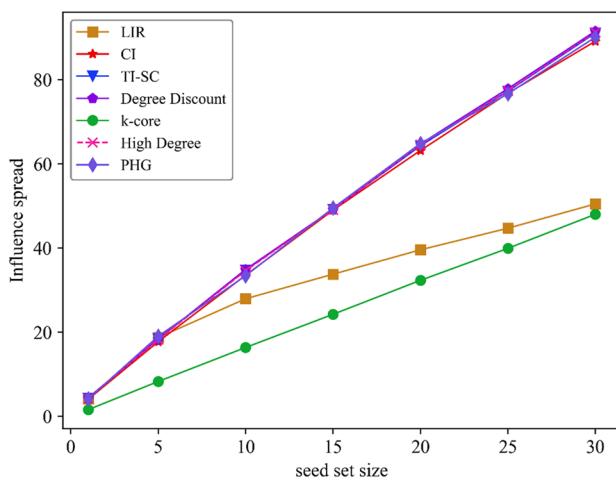


Fig. 9 Influence spreads of different algorithms on the M-FO120

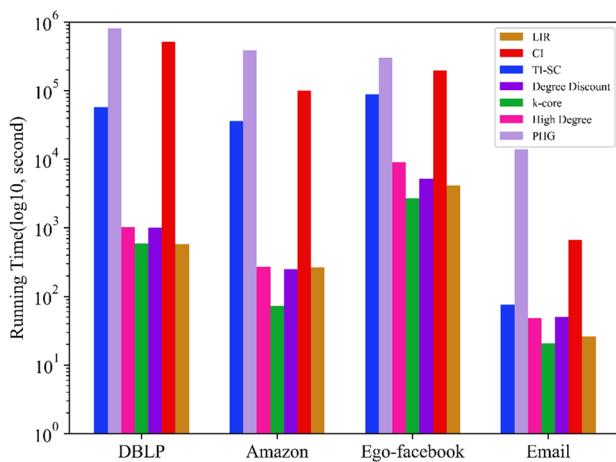


Fig. 10 Running time of different algorithms on four real-world datasets ($k=30$)

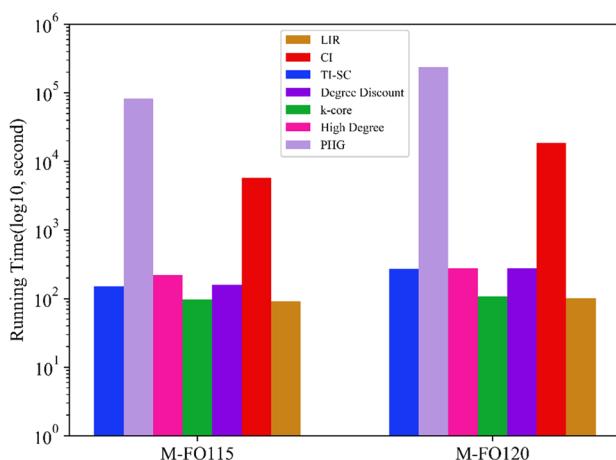


Fig. 11 Running time of different algorithms on two synthetic networks ($k=30$)

algorithm, the relation $LI(v_i) = \sum_{v_j \in N(v_i)} Q(d_j - d_i)$ is first computed for each node, where d_i is the degree of node v_i , and $N(v_i)$ is neighbors of node v_i . Then, nodes with $LI=0$ are arranged descendingly based on their degree. Hence, k influential nodes with $LI=0$ are chosen. It should be noted that in the LIR algorithm, the number of the nodes with $LI=0$ in two Email and Ego-facebook datasets was 6 and 5, respectively. We need at least 30 nodes with $LI=0$ with regard to the number of seed nodes. Therefore, if the number of the nodes with $LI=0$ is less than required number (e.g. 30 nodes), it may need the other required influential nodes with $LI=1$ in the LIR algorithm. Needless to say that the nodes with $LI=1$ are more important than the nodes with $LI=0$.

PHG The PHG algorithm presented to improved the Greedy algorithm (Qiu et al. 2019). First, this algorithm finds key nodes in each community to construct a candidate set by detecting community structure. Second, it finds the most potential influence nodes from a candidate set by combining the influence weight of nodes. Finally, it greedily selects the nodes with maximization marginal gain.

High Degree The High-Degree algorithm arranges the nodes descendingly by degree, and selects k nodes with the highest degree as the influential ones (Chen et al. 2009).

4.3 Evaluation metrics

Similar to other state-of-the-art algorithms (Chen et al. 2009; Cheng et al. 2013; Goyal et al. 2011b; Liu et al. 2017; Wang et al. 2012), we use two metrics to evaluate the performance of our algorithm.

- **Influence spread** Given seed set, influence spread is defined as the number of expected active nodes with the Monte-Carlo simulation in IC and LT models. It is used to evaluate the accuracy of an influence maximization algorithm. Algorithm with Higher influence spread value has more accurate than others. We repeat 1000 Monte-Carlo simulations to compute the influence spread for any given seed set.
- **Running time** Running time is defined as the time for selecting k seed nodes. In our experiments, k is set to 30.

4.4 Experiment setup

- **Diffusion model** we use the Independent cascade model. The propagation probability from node u to v is $p_{uv} = 0.01$. The influence spread value is computed by repeating 1000 Monte-Carlo simulations.
- **Experimental environment** The experiments are carried out on a computer with 2.5 GHz Intel core i5 CPU-

Table 2 Impact of θ criterion on the influence spread and runtime of the TI-SC algorithm on the Ego-facebook dataset

	$\theta=0.02$	$\theta=0.03$	$\theta=0.04$	$\theta=0.05$	$\theta=0.07$	$\theta=0.09$	$\theta=0.1$	$\theta=0.3$
K=1	144.27	144.845	144.059	142.35	141.183	144.1639	144.737	142.8589
K=5	180.184	178.936	188.668	264.5589	271.9869	271.1339	269.2329	269.4199
K=10	182.184	184.715	207.0699	296.842	307.7399	307.353	306.0139	310.2455
K=15	187.137	188.126	219.0189	314.9919	327.047	330.084	328.945	333.8794
K=20	191.435	192.594	226.0495	335.054	343.476	350.479	346.294	352.8541
K=25	192.785	196.4559	234.8455	348.275	359.856	363.6139	364.3159	373.3629
K=30	197.354	197.8149	240.8655	359.0039	373.803	378.575	375.4241	382.8219
Time	2050.615	2448.113	4809.297	2108.5239	33,061.842	40,016.797	53,393.106	88,789.59

Table 3 Impact of θ criterion on the influence spread and runtime of the TI-SC algorithm on M-FO115 dataset

	$\theta=0.4$	$\theta=0.5$	$\theta=0.6$	$\theta=0.72$	$\theta=0.75$	$\theta=0.8$	$\theta=0.85$	$\theta=0.9$
K=1	3.0009	3.0639	3.1029	3.0539	3.0219	3.0679	3.0159	3.0589
K=5	12.5979	12.7609	12.8559	12.9049	12.7689	12.8929	12.7369	12.9399
K=10	24.7119	24.8859	25.0919	25.1939	25.2679	24.9099	24.7379	24.9129
K=15	36.2759	35.8269	35.969	36.2459	36.027	36.2399	36.086	35.78
K=20	46.3229	46.3589	46.4419	46.2069	46.4719	46.2059	46.4309	46.2749
K=25	56.7859	56.5099	56.2279	56.6669	56.3589	56.4669	56.5179	56.6559
K=30	66.138	65.6179	65.96	65.52	65.5689	64.063	64.25	65.635
Time	150.895	186.839	239.028	2310.73	2808.07	5911.179	9884.32	11,343.446

Table 4 Impact of θ criterion on the influence spread and runtime of the TI-SC algorithm on M-FO120 data set

	$\theta=0.3$	$\theta=0.4$	$\theta=0.5$	$\theta=0.6$	$\theta=0.7$	$\theta=0.8$	$\theta=0.85$	$\theta=0.9$
K=1	4.2619	4.0459	4.3929	4.3119	4.1209	4.5289	4.3509	4.2719
K=5	18.3809	19.0639	18.8359	19.0449	18.7429	18.8329	18.8069	19.0669
K=10	34.7759	34.928	34.69	35.1829	34.912	34.6629	34.8219	35.0169
K=15	49.1719	48.7619	49.5279	49.4729	48.9129	49.1149	48.8859	49.6009
K=20	64.3309	64.168	62.8949	63.277	63.6429	63.095	63.701	63.777
K=25	77.3089	77.0329	77.4969	78.3599	76.9019	77.0889	76.7269	76.9319
K=30	91.1279	90.4819	90.9419	91.5889	90.9769	90.7679	90.5119	90.3059
Time	274.572	285.959	297.124	468.906	2774.493	7538.773	12,092.164	13,864.680

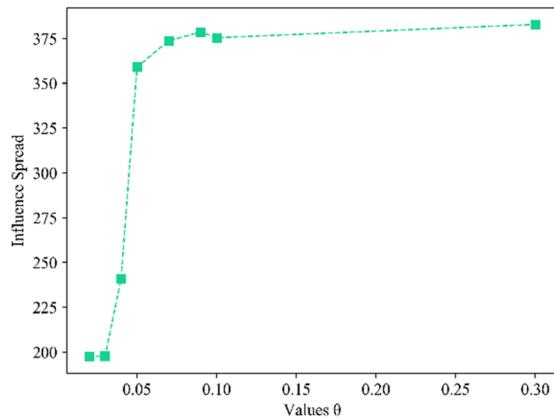
3230 M and 16 GB memory. All codes are programmed in python and the programs are in single process and single thread.

4.5 Result

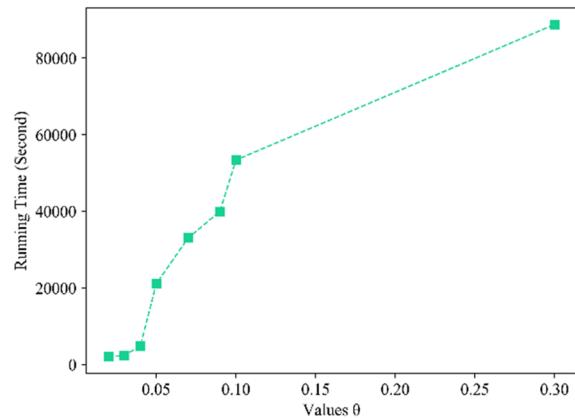
4.5.1 Influence spread

We first compare the influence spread of different algorithms on four real-world datasets where the x-axis represents the number of seed nodes while the y-axis represents the overall influence spread. From the results of the four real-world datasets, we see that our TI-SC algorithm always outperforms other compared methods in terms of influence spread. The k-core algorithm shows the worst performance on all networks. The High Degree and DegreeDiscount methods, though performs well on some datasets, cannot provide any

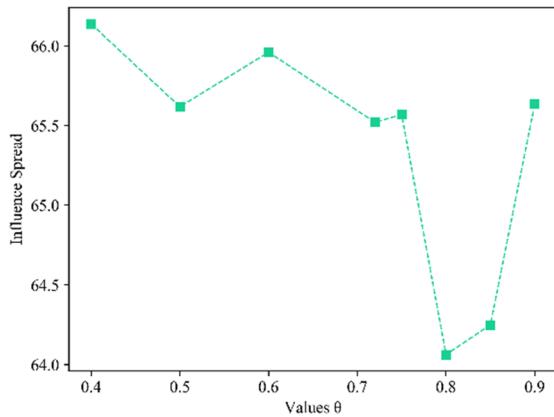
performance guarantee. For example, Figs. 4 and 5 show that their influence spread values is weak in datasets DBLP and Ego-Facebook. It shows the significant gaps of influence spread value between DegreeDiscount and TI-SC algorithms in Fig. 4. On the DBLP dataset, the influence spread value ($k=10$) of DegreeDiscount is 72.342, while the TI-SC algorithm is 91.529. It is revealed that the TI-SC has better performance than PHG and DegreeDiscount in Fig. 5. On the Ego-Facebook dataset, the influence spread value ($k=30$) of DegreeDiscount and PHG is 367.86 and 375.1619, respectively. However, this values is 382.8219 for the TI-SC algorithm. In addition, Fig. 6 represents that TI-SC, High Degree and DegreeDiscount algorithms exhibit the same influence spread value. When k is small (e.g., $k \leq 15$), we see in Fig. 6 that the influence spread values of the LIR algorithm are the same with TI-SC, High Degree, and DegreeDiscount algorithms. In Fig. 7, it is obvious that the influence spread



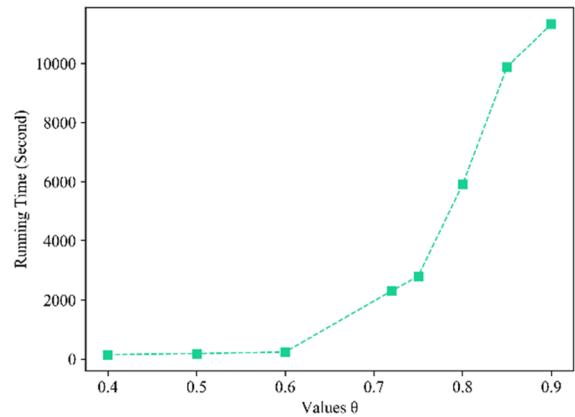
(a) Impact of θ criterion on the influence spread of the TI-SC algorithm on the Ego-Facebook data set in $K=30$



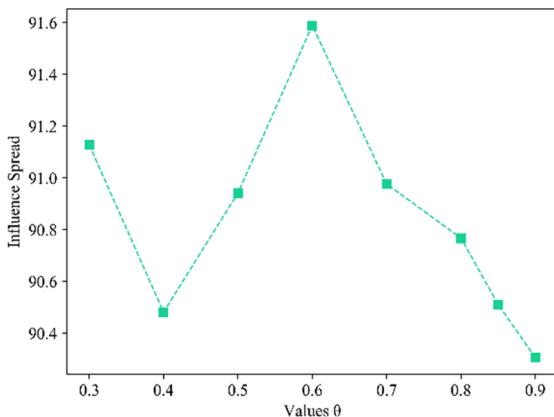
(b) Impact of θ criterion on the runtime of the TI-SC algorithm on the Ego-Facebook data set in $K=30$



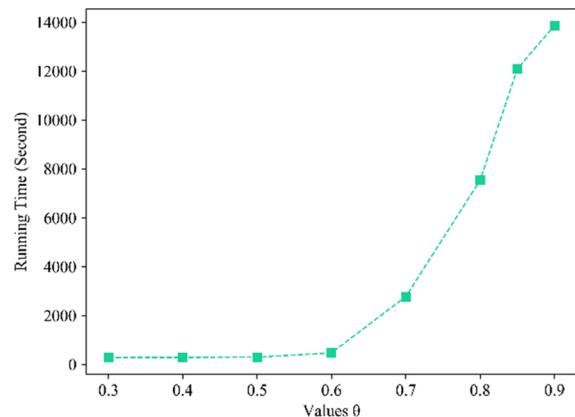
(c) Impact of θ criterion on the influence spread of the TI-SC algorithm on the M-FO115 data set in $K=30$



(d) Impact of θ criterion on the runtime of the TI-SC algorithm on the M-FO115 data set in $K=30$



(e) Impact of θ criterion on the influence spread of the TI-SC algorithm on the M-FO120 data set in $K=30$



(f) Impact of θ criterion on the runtime of the TI-SC algorithm on the M-FO120 data set in $K=30$

Fig. 12 Impact of θ parameter on the influence spread and runtime of the TI-SC algorithm in $k=30$

values of DegreeDiscount, CI and PHG algorithms are lower than our TI-SC algorithm. For example, on the Email data-set with $k=30$, the TI-SC algorithm achieves an influence

spread value of 42.9783, while the value of DegreeDiscount, CI and PHG are 41.1737, 41.2561 and 41.0019, correspondingly. Overall, from the results on the real-world networks,

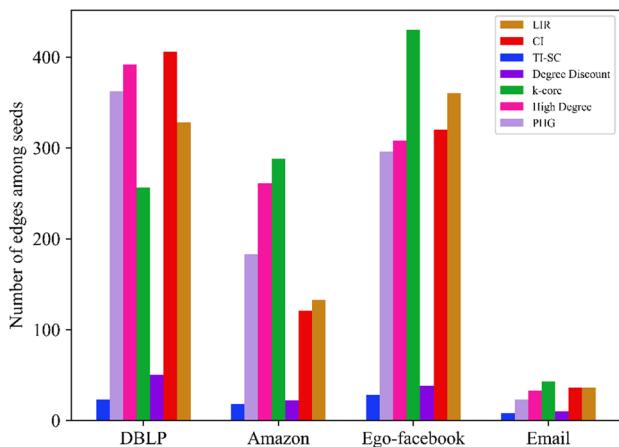


Fig. 13 The number of edges that connect two seeds on four real-world datasets

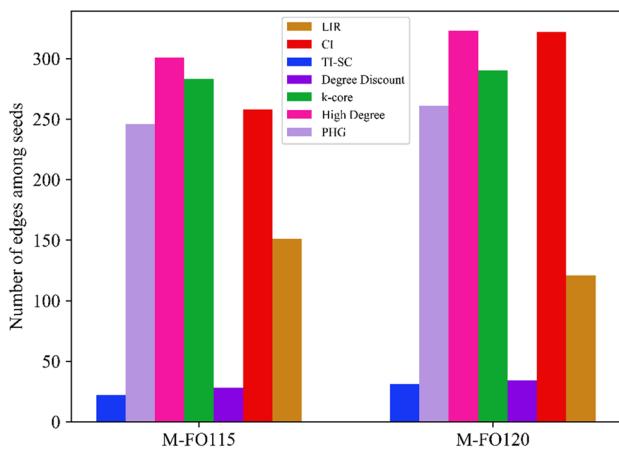


Fig. 14 The number of edges that connect two seeds on two synthetic networks

our TI-SC algorithm shows its effectiveness in finding top influential seed nodes as compared with the state-of-the-art algorithms. Furthermore, the DegreeDiscount and CI methods have the second and third best performance, respectively.

We compare the influence spread of different algorithms on two synthetic networks. It is obvious in Fig. 8 that the influence spread values of TI-SC algorithm are best. Figure 9 shows that TI-SC, High Degree and DegreeDiscount algorithms exhibit the same influence spread value.

4.5.2 Running time metric

Figures 10 and 11 shows the running time of different algorithms on the four real-world datasets and two synthetic networks. The running time is the time of selecting $k=30$

seed nodes. As mentioned above, The DegreeDiscount, High Degree, LIR, and k-core, though run fast, but cannot provide any performance guarantee in terms of influence spread. The worst running time is for the PHG algorithm. From the results, we see in Fig. 11 that the TI-SC algorithm is very competitive in its time efficiency, requiring less than some second on most datasets.

4.5.3 Impact of θ factor

According to Table 1, the value of the factor θ has been determined in each data set, which has a significant effect on the computational overhead reduction. If the θ -value is set with a large value, it increases the computational overhead, and if the θ -value is set to a very small value, computational overhead and influence spread will be decreased. Therefore, an appropriate value should be set for this parameter. This study demonstrates the impact of the factor θ on the computations. The results are analyzed only on the three M-FO115, M-FO120, and Ego-Facebook data sets.

According to Tables 2, 3, and 4, θ -values affected the influence spread calculations. As mentioned above, the nodes within communities with a φ -value greater than θ are known as the nodes with low influence spread, and they are not involved in selecting the influence spread $\sigma(S \cup v_{ms})$. Moreover, these communities is filtered and not updated. K in Tables 2, 3, and 4 displays the number of the seed nodes, and the numbers in the columns θ represent the influence spread in the number of different seeds with regard to different values of θ .

Based on the obtained results in Table 2, when the θ -values are small, low influence spread is provided in the number of different seeds; however, the algorithm is run relatively fast. For example, the algorithm runtime is 2050.615 when $\theta=0.02$, which is an acceptable runtime for the TI-SC algorithm; however, the influence spread of the nodes is low. This means that when the value of θ is very small, communities with influential nodes are eliminated from the influence spread calculations which consequently declines the influence spread. Therefore, the θ -value should be set to an appropriate value to tradeoff runtime and influence spread metrics. In the value of $\theta=0.3$, the TI-SC algorithm provides the maximum influence spread in a majority of K-values; however, the runtime of the TI-SC algorithm will prolong at this value of θ . For instance, for the Ego-Facebook dataset, an appropriate value of θ can be set to 0.05. Table 3 illustrates that when the values of θ increases, the influence spread will be remained constant at different K-values. However, as seen in the Time line, when θ -values increase, the time also increases. This means that there are many communities with little influential nodes that increase computational overhead, whereas they do not affect the increase of influence spread. Therefore, $\theta=0.4$

is the best value for the M-FO115 dataset in this experiment to balance the runtime and influence spread. The runtime equals to 150.895 at $\theta=0.4$, providing that its influence spread is approximately the same with the influence spread of different K-values in different seeds. As seen in Table 4, when the values of θ increase, the influence spread remains constant in different K-values. However, when θ -values increase, the time also increases. Therefore, the best value for the M-FO120 dataset is $\theta=0.3$ in order to balance between runtime and influence spread.

Figure 12 depicts the impact of θ -values on the influence spread and runtime of the TI-SC algorithm in Ego-facebook, M-FO115, and M-Fo120 datasets at $K=30$. Figure 12a, b respectively portray the impact of θ -value on the influential spread and runtime of the TI-SC algorithm in the Ego-facebook data set. As shown in the figures, when θ increases, the time and influence spread will be increased. The maximum influence spread and runtime will be at $\theta=0.3$, whereas the minimum runtime and influence spread will be at $\theta=0.02$. Figure 12c, d respectively depict the impact of θ on the influence spread and runtime of the TI-SC algorithm in the M-FO115 data set. As shown in Fig. 12d, the runtime of the TI-SC algorithm increases for large values of θ . Based on Fig. 12c, there is no difference in the influence spread for different values of θ , and the influence spread in different θ s differs a few percent. Figure 12e, f show the impact of θ on the influence spread and runtime of the TI-SC algorithm in the M-FO120 data set. As shown in Fig. 12f, the runtime of the TI-SC algorithm also increases with the increase of θ -values. The lowest runtime is obtained at $\theta=0.3$ and the highest runtime is belonged to $\theta=0.9$. According to Fig. 12e, there is no significant difference in the influential spread for different values of θ , and the influence spread in different θ s differs a few percent.

4.5.4 Rich-Club phenomenon

The effect of Rich-Club phenomenon is shown in Figs. 13 and 14. Figure 13 display the number of edges that connect seeds in real-world networks (rich-club phenomenon). For instance, there are 23 edge between the selected seeds by the TI-SC algorithm on the DBLP dataset. It is obvious that the lowest Rich-Club coefficient is belonged to the TI-SC algorithm. There also exist 18 edge between the selected seeds by the TI-SC algorithm on the Amazon dataset that the TI-SC has the best result but K-core algorithm has the worst tolerance. In addition, in Ego-Facebook dataset, there are 28 edge between the selected seeds for our algorithm. Like Amazon, the K-core is the worst algorithm but TI-SC is the best method. In the Email dataset, the TI-SC algorithm and other algorithms is not large. However, the performance

of our algorithm and DegreeDiscount algorithm are very close in terms of influence spread. Consequently, the TI-SC algorithm efficiently tolerates the rich-club problem in comparison to other methods. The main disadvantages of the TI-SC is that its performance is remarkably not better than other algorithms in the networks with low Rich-Club coefficient. In such a network, all algorithms nearly has same performance.

In Fig. 14, we show the number of edges that connect seeds in synthetic networks. There are 22 edge between the selected seeds by the TI-SC algorithm on the M-FO115 and M-FO120 datasets that HighDegree algorithm is the worst algorithm but the TI-SC method is the best.

5 Conclusions

In this paper, we proposed an efficient community-based algorithm combined with a scoring measure for selecting top-K influential nodes. The proposed TI-SC algorithm consists of four steps: 1. Community detection, 2. Merge of communities, 3. Computing scoring criteria Selection of the first influential node, and 4. Updating scoring criteria. TI-SC algorithm solved the influence maximization in near-linear time complexity. Due to the filtering and reduction process in the updating phase, this algorithm has fast convergence to an optimized set of K seed nodes. The proposed algorithm is evaluated based on the independent cascade model. In the TI-SC algorithm, a scoring criteria inspired by the real-world has been used to calculate the influence spread of each node. To avoid the thousands of additional comparisons for influence computations, we have reduced the search space, which improves the runtime compared to similar algorithms. In the TI-SC algorithm, the scoring criterion reduces the overlap of seed nodes, and this leads to the selection of K -node with optimal influence spread. The most important implications of the TI-SC algorithm in the real-world is to find politically influential nodes with a scoring criterion, help to design an efficient advertisement strategy and a better immunization strategy. Since the real world social networks usually has higher Rich Club coefficient, our proposed algorithm remarkably outperforms other algorithms. The experimental results on both large-scale and medium-scale real-world networks and also two synthetic networks indicate that our algorithm is considerably more efficient than other algorithms in terms of influence spread and runtime metrics. We believe that the proposed algorithm can provide a deep insight into future studies of influence maximization.

References

- Ahajam S, Badir H (2018) Identification of influential spreaders in complex networks using HybridRank algorithm. *Sci Rep* 8:11932
- Atif Y, Al-Falahi K, Wangchuk T, Lindström B (2019) A fuzzy logic approach to influence maximization in social networks. *J Ambient Intell Humaniz Comput* 1–17
- Banerjee S, Jenamani M, Pratihar DK (2019a) ComBIM: a community-based solution approach for the budgeted influence maximization problem. *Expert Syst Appl* 125:1–13
- Banerjee S, Jenamani M, Pratihar DK (2019b) Maximizing the earned benefit in an incentivized social networking environment: a community-based approach. *J Ambient Intell Humaniz Comput* 1–17
- Barabási A-L, Albert R (1999) Emergence of scaling in random networks. *Science* 286:509–512
- Berahmand K, Bouyer A (2019) A link-based similarity for improving community detection based on label propagation algorithm. *J Syst Sci Complex* 32:737–758. <https://doi.org/10.1007/s11444-018-7270-1>
- Berahmand K, Bouyer A, Samadi N (2018) A new centrality measure based on the negative and positive effects of clustering coefficient for identifying influential spreaders in complex networks. *Chaos, Solitons Fractals* 110:41–54
- Berahmand K, Bouyer A, Samadi N (2019) A new local and multi-dimensional ranking measure to detect spreaders in social networks. *Computing* 101:1711–1733
- Borgs C, Brautbar M, Chayes J, Lucier B (2014) Maximizing social influence in nearly optimal time. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms. SIAM, pp 946–957
- Bozorgi A, Samet S, Kwisthout J, Wareham T (2017) Community-based influence maximization in social networks under a competitive linear threshold model. *Knowl-Based Syst* 134:149–158
- Chen W, Wang Y, Yang S (2009) Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 199–208
- Chen Y-C, Zhu W-Y, Peng W-C, Lee W-C, Lee S-Y (2014) CIM: community-based influence maximization in social networks. *ACM Trans Intell Syst Technol* 5:25
- Chen W, Lin T, Tan Z, Zhao M, Zhou X (2016) Robust influence maximization. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 795–804
- Cheng S, Shen H, Huang J, Zhang G, Cheng X (2013) Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In: Proceedings of the 22nd ACM international conference on information and knowledge management. ACM, pp 509–518
- Domingos P, Richardson M (2001) Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 57–66
- Gmati H, Mouakher A, Gonzalez-Pardo A, Camacho D (2018) A new algorithm for communities detection in social networks with node attributes. *J Ambient Intell Humaniz Comput* 1–13
- Goyal A, Lu W, Lakshmanan LV (2011a) Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th international conference companion on World wide web. ACM, pp 47–48
- Goyal A, Lu W, Lakshmanan LV Simpath (2011b) An efficient algorithm for influence maximization under the linear threshold model. In: 2011 IEEE 11th international conference on data mining. IEEE, pp 211–220
- Guimera R, Danon L, Diaz-Guilera A, Giralt F, Arenas A (2003) Self-similar community structure in a network of human interactions. *Phys Rev E* 68:065103
- Hosseini-Pozveh M, Zamanifar K, Naghsh-Nilchi AR (2017) A community-based approach to identify the most influential nodes in social networks. *J Inf Sci* 43:204–220
- Huang H, Shen H, Meng Z, Chang H, He H (2019) Community-based influence maximization for viral marketing. *Appl Intell*, pp 1–14
- Jung K, Heo W, Chen W (2012) Irie: scalable and robust influence maximization in social networks. In: 2012 IEEE 12th international conference on data mining. IEEE, pp 918–923
- Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 137–146
- Kitsak M, Gallos LK, Havlin S, Liljeros F, Muchnik L, Stanley HE, Makse HA (2010) Identification of influential spreaders in complex networks. *Nat Phys* 6:888
- Lawyer G (2015) Understanding the influence of all nodes in a network. *Sci Rep* 5:8665
- Leskovec J, Mcauley JJ (2012) Learning to discover social circles in ego networks. In: Advances in neural information processing systems. pp 539–547
- Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance N (2007) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 420–429
- Li X, Guo J, Gao C, Zhang L, Zhang Z (2018) A hybrid strategy for network immunization. *Chaos, Solitons Fractals* 106:214–219
- Liu B, Cong G, Xu D, Zeng Y (2012) Time constrained influence maximization in social networks. In: 2012 IEEE 12th international conference on data mining. IEEE, pp 439–448
- Liu D, Jing Y, Zhao J, Wang W, Song G (2017) A fast and efficient algorithm for mining top-k nodes in complex networks. *Sci Rep* 7:4330
- Ma H, Zhu Y, Li D, Kim D, Liang J (2015) Improving the influence under IC-N model in social networks. *Discrete Math Algorithms Appl* 7:1550037
- Morone F, Min B, Bo L, Mari R, Makse HA (2016) Collective influence algorithm to find influencers via optimal percolation in massively large social media. *Sci Rep* 6:30062
- Ni Y, Shi Q, Wei Z (2017) Optimizing influence diffusion in a social network with fuzzy costs for targeting nodes. *J Ambient Intell Humaniz Comput* 8:819–826
- Qiu L, Jia W, Yu J, Fan X, Gao W (2019) PHG: a three-phase algorithm for influence maximization based on community structure. *IEEE Access* 7:62511–62522
- Rossi M-EG, Shi B, Tziortziotis N, Malliaros FD, Giatsidis C, Vazirgiannis M (2018) MATI: an efficient algorithm for influence maximization in social networks. *PLoS ONE* 13:e0206318
- Ruan Z, Iniguez G, Karsai M, Kertész J (2015) Kinetics of social contagion. *Phys Rev Lett* 115:218702
- Samadi N, Bouyer A (2019) Identifying influential spreaders based on edge ratio and neighborhood diversity measures in complex networks. *Computing* 101:1147–1175
- Shang J, Zhou S, Li X, Liu L, Wu H (2017) CoFIM: a community-based framework for influence maximization on large-scale networks. *Knowl-Based Syst* 117:88–100
- Wang X-G (2016) A new algorithm for the influence maximization problem in dynamic networks or traffic sensor networks. *Multimedia Tools Appl* 75:4833–4844
- Wang Y, Cong G, Song G, Xie K (2010) 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, pp 1039–1048

- Wang C, Chen W, Wang Y (2012) Scalable influence maximization for independent cascade model in large-scale social networks. *Data Min Knowl Disc* 25:545–576
- Wang Q, Jin Y, Lin Z, Cheng S, Yang T (2016) Influence maximization in social networks under an independent cascade-based model. *Phys A* 444:20–34
- Xin Y, Gao C, Wang Z, Zhen X, Li X (2019) Discerning influential spreaders in complex networks by accounting the spreading heterogeneity of the nodes. *IEEE Access* 7:92070–92078
- Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. *Knowl Inf Syst* 42:181–213

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.