Physica A 592 (2022) 126841

Contents lists available at ScienceDirect

Physica A

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

Influence maximization problem by leveraging the local traveling and node labeling method for discovering most influential nodes in social networks

Asgarali Bouyer *, Hamid Ahmadi Beni

Azarbaijan Shahid Madani University, Tabriz, Iran

ARTICLE INFO

Article history:
Received 2 August 2021
Received in revised form 17 November 2021
Available online 4 January 2022

Keywords:
Social networks
Influence maximization
Node labeling
Influence spread
Independent cascade model

ABSTRACT

The influence maximization problem has gained particular importance in viral marketing for large-scale spreading in social networks. Developing a fast and appropriate algorithm to identify an optimized seed set for the diffusion process on social networks is crucial due to the fast growth of networks. Most fast methods only focus on the degree of nodes while ignoring the strategic position of nodes in the networks. These methods do not have the required quality in finding a seed set in most networks. On the other hand, many other methods have acceptable quality, but their computational overhead is significant. To address these issues, the main concentration of this paper is to propose a fast and accurate method for the influence maximization problem, which uses a local traveling for labeling of nodes based on the influence power, called the LMP algorithm. In the proposed LMP algorithm, first, a travel starts from a node with the lowest influence power to assign a ranking-label for this node and its neighbor nodes in each step based on their diffusion capability and strategic position. The LMP algorithm uses node labeling steps to reduce search space significantly. Three ranking-labels are used in the proposed algorithm, and nodes with the highest ranking-label are selected as candidate nodes. This local and fast step strictly reduces the search space. Finally, the LMP algorithm selects seed nodes based on the topology features and the strategic position of the candidate and connector. The performance of the proposed algorithm is benchmarked with the well-known and recently proposed seed selection algorithms. The experimental results are performed on real-world and synthetic networks to validate the efficiency and effectiveness. The experiments exhibit that the proposed algorithm is the fastest in comparison with other state-of-the-art algorithms, and it has linear time complexity. In addition, it can achieve a good tradeoff between the efficiency and time complexity in the influence maximization problem.

1. Introduction

Diffusion of information, ideas, innovations, and viral marketing through social networks is inevitable in the world of social media [1,2]. The diffusion process in social networks has a significant impact on the economy and human life

* Corresponding author.
E-mail addresses: a.bouyer@azaruniv.ac.ir (A. Bouyer), h.ahmadi@azaruniv.ac.ir (H.A. Beni).

https://doi.org/10.1016/j.physa.2021.126841
0378-4371/

with many applications in social networks [3,4]. Therefore, maximizing diffusion has great importance in the recent decade [2,5–19]. However, there are many challenges in identifying the influential nodes for maximizing diffusion [5,7,20–22]. Influence maximization (IM) as a stochastic optimization problem was first proposed by Domingos [23]. It is a discrete stochastic optimization problem that its goal is to select k influential nodes (seed set) that can spread the influence on the maximum number of individuals [1]. The nodes in the seed set are initial nodes to spread of influence to other nodes. In the IM problem, nodes have two active and inactive states in the diffusion process [5]. In each diffusion step, active nodes attempt to change inactive nodes' state to active nodes based on a diffusion model. There are two classic progressive models: the independent cascade (IC) model and the linear threshold (LT) model. In the IC model, each edge has an influence probability that each active node v executes an activation attempt over the inactive node u. If node v successfully active the node u, the state of node u is changed to "newly activated". In this model, if v does not activate u, it will not try to activate u in the next diffusion steps. Also, in the linear threshold model, each node $v \in V$ is associated with a threshold $\theta_v \in [0, 1]$, and each edge is associated with an influence weight $w \in [0, 1]$. If the total weight of the edge from its active in-neighbors is at least θ_v, then the state of the inactive node is changed to "active" [5]. Both the IC and LT models share two critical properties in the influence maximization problem. The influence spread function on these models is monotone and submodular [5,24]. A set function $f : 2^V \rightarrow R$ is called submodular if for each STV and $v \in V$, $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$. A set function $f : 2^V \rightarrow R$ is called monotone if each subsets STV, $f(S) \leq f(T)$.

Most of the proposed IM algorithms recently embraced two major challenges [11–15,25–35]. The first problem is that these algorithms do not have time-efficient [36]. To solve this challenge, several algorithms were proposed such as simulation-based [6,7,37,38], proxy-based [8,37], sketch-based [7,39–41], community-based [9,11,12,29,42–44], etc. The second challenge is that some algorithms cannot identify an acceptable seed set. Several algorithms were proposed to solve this challenge, such as path-based [10], topic-based [45,46], etc. However, most of them are not adequate due to weak performance and computational overhead, and some others cannot attain a proper tradeoff between efficiency and time complexity. To address these problems, we propose a new algorithm that performs a local travel from weak or periphery nodes to assign a ranking-label based on nodes' influence power. In the LMP algorithm, the strategic position of the node is specifically important for ranking and diffusion. Therefore, this algorithm selects seed nodes based on a strategic position for diffusion and network topology features. This method includes three general phases. At first, nodes are labeled based on their influence power in a local travel step. In fact, These labels are three different ranking labels with values 0, 1, and 2. Next, nodes with the highest ranking-label and some middle nodes with the middle ranking-label are selected to create a sub-graph. Some nodes with middle ranking-label can play a significant role in the influence spread. all connected nodes in sub-graph are considered as candidate set. At the end, seed nodes are selected among influential candidate nodes of sub-graph based on their strategic position of diffusion. In some algorithms [7,26–28], two nodes with similar topological features are randomly selected as seed. But in the LMP algorithm, if two nodes have the same topology features, the node with suitable strategic position in sub-graph is selected as final seed. Our major contributions in this algorithm are:

1. We proposed a new algorithm based on node labeling that labels diffusion based on node Influence Power. That is why nodes with great influence spread select.
2. We use a special position among the important nodes for finding seed nodes. Nodes with gatekeeper and bridge property are important among important nodes and play a significant role in influence spreading.
3. The experimental results on real-world and synthetic networks demonstrate that the proposed algorithm LMP performs better than state-of-the-art influence maximization algorithms in terms of influence spread and time complexity.

The remainder of this paper is structured as follows. Section 2 reviews some related studies on the influence maximization problem. In Section 3, a detailed description of the proposed algorithm is proposed. Section 4 presents the experimental results and evaluations. Finally, Section 5 expressions a conclusion of this paper and future directions.

2. Related work

The influence maximization problem has been divided into six categories: simulation-based, sketch-based, proxy-based, heuristic-based, meta-heuristic, and community-based. We discuss these sections with more details below.

2.1. Simulation-based algorithms

The Greedy algorithm was first proposed to influence the maximization problem by Kempe et al. [5]. This algorithm is dependent on the Monte Carlo simulation. This algorithm requires to repeat evaluations of the influence spread. The main drawback of this algorithm is its inefficiency. Then, the CELF (Cost-Effective Lazy Forward selection) algorithm was proposed to improve the Greedy algorithm's running time [6]. This algorithm uses the submodularity feature and lazy evaluations. This algorithm does not calculate the influence spread for each node. Then, Liqing et al. proposed the TSIM (two-stage selection for influence maximization in social networks) algorithm that utilizes the DegreeDiscount approach and lazy evaluation technique [47]. Its efficiency of the influence spread is a bit satisfactory, but its time complexity

in comparison to other algorithms is not efficient. Then, Ding et al. proposed R-Greedy, D-Greedy algorithms under the Realistic Independent Cascade (RIC) model [48]. In the R-greedy algorithm, influential nodes select from the candidate set based on a certain criterion considering both the influence and the candidate nodes' acceptance probability. Also, D-Greedy reduces the time complexity of R-greedy. Aghaee et al. proposed HEDVGreedy (High degree Expected Diffusion Value Greedy) based on the Greedy and High Degree algorithms. This algorithm reduces search space with nodes with a high degree [34].

2.2. Sketch-based algorithms

Chen et al. presented NewGreedyIC [7]. The NewGreedyIC algorithm generates R random graphs $G^{'}$ that R is the number of Monte Carlo simulations. Then, the influence spread is calculated on the random graphs $G^{'}$. The influence spread of the NewGreedyIC algorithm is like the Greedy algorithm, but its running time is much less than the Greedy algorithm. Then, StaticGreedy reduces the NewGreedyIC's running time. This algorithm creates Snapshots based on the independent cascade model. After that, the influence spread calculates based on reachable nodes [39]. Tang et al. proposed the TIM (Two-phase InfluenceMaximization) algorithm based on the triggering model [49]. This algorithm uses a lower-bound of the maximum expected influence spread, and this algorithm creates RR sets and selects nodes that cover a large number of RR sets. Then, algorithm IMM proposed that consists of two phases. In phase 1, the IMM (Influence Maximization via Martingales) algorithm randomly samples the graph. After that, this algorithm uses the greedy algorithm for maximum coverage [50].

2.3. Proxy-based algorithms

Selecting nodes with maximum degrees as seed nodes may have an effect on the influence spread. Therefore, DegreeDiscount was proposed as a degree-based algorithm considering the effect of neighbors' degree for selecting a high degree node as a seed node to improve the influence spread [7]. Narayanam et al. proposed the SPIN (ShaPley value based Influential Nodes) algorithm for solving the influence maximization problem using the concept of Shapley value [8]. This algorithm has a good running time, but the selected seed set's quality is not fitting. Then, Liu et al. defined the time constrained influence maximization problem in social networks [51]. They proposed the MISP (Marginal Discount of Influence Spread Path) algorithm for calculating the activation probability of a node as a seed. The MISP algorithm achieves the efficiency of influence spread like the Greedy algorithm. Also, the running time of this algorithm remains constant for different values of k seed nodes.

2.4. heuristic-based algorithms

Kitsak et al. used the k-shell decomposition analysis to find core nodes and select seed set among core nodes using a new measure [52]. This algorithm does not guarantee optimal approximation of seed set because the best spreaders necessarily are not among core nodes. Morone et al. proposed the CI (Collective Influence) algorithm that finds the influential nodes via optimal Percolation [26]. This algorithm calculates the influence spread by removing nodes. However, the CI algorithm does not guarantee optimal approximation. To further improve the CI algorithm, Zhang et al. presented the VoteRank algorithm based on voting ability that avoids the Rich-club phenomenon [27]. This algorithm has a proper running time. But it does not have suitable quality in selecting the seed set. Then, Liu et al. presented the LIR (local index rank) algorithm that selects influential nodes by the local index [28]. This algorithm computes the local index (LI) of each node and selects all nodes with $LI = 0$. Then, nodes with $\mathrm{LI} = 0$ rank by their degree. Finally, this algorithm selects k node with the highest degree as the seed nodes. Like the VoteRank algorithm, LIR has a suitable running time, but the influence spread is limited. Nguyen et al. presented the ProbDegree algorithm that selects seed nodes based on the number of neighbors and the effect of two-hop away neighbors [53]. This algorithm outperforms state-of-the-art algorithms in terms of influence spread, but its computational overhead is considerable. Then, the RNR (Reversed Node Ranking) algorithm is proposed based on rank [32]. This algorithm uses the reversed rank for selecting seed nodes because this is based on the idea that the node with a higher rank has stronger influence power. Yu et al. proposed a multi-action credit distribution model to quantify the influence ability of each user that the influence ability is submodular [54]. The multi-action credit distribution model has the effectiveness and efficiency with the streaming algorithm. SFRM (shall-based ranking and filtering method) selects influential nodes based on shell-based ranking [55]. In this algorithm, not all seed nodes are in the center of the graph shell. Some seeds are in the center shell, and the rest of the seeds are in the neighbor shells of the center shell. This algorithm is very fast. Li et al. proposed the DEIM (Dynamic algorithm based on cohesive Entropy for Influence Maximization) algorithm that this algorithm is based on cohesive Entropy for Influence Maximization [56]. Also, it utilizes an entropy calculation to obtain influential nodes. Wang et al. proposed a novel influence maximization based on a sensitive centrality measure [57]. This algorithm has stable performance on graphs with different scales and various structural characteristics.

2.5. Meta-heuristic algorithms

The ADABC (adaptive discrete artificial bee colony) algorithm proposed a three-layer-comprehensive-influence evaluation (TLCIE) model [58]. Then, Zareie et al. proposed the GWIN (Gray Wolf based Influence Maximization) algorithm, which uses the distance between the nodes as a cost function and the gray wolf optimization algorithm to solve the IM problem [59]. Tang et al. proposed a DSFLA (discrete shuffled frog-leaping algorithm) algorithm that selects seed nodes by a discrete shuffled frog-leaping algorithm [16]. This algorithm implements global exploration for seed nodes and local exploitation on each memeplex. DSFLA selects seed nodes effectively. Wang et al. proposed a new model in multiplex networks that have different layers. This algorithm selects seed nodes in each layer based on a memetic algorithm [60]. Also, Wang et al. presented a measure to evaluate the community's robustness. Then, the community robustness enhances using a memetic optimization algorithm to improve influence spread in social networks [61].

2.6. Community-based algorithms

On the other hand, some other researches have focused on IM problems based on community detections. Qiu et al. presented a community-based algorithm that selects the most potential influence nodes by the influence weight of nodes [11]. Also, it obtains a better influence spread. But this algorithm has not a good running time. Beni et al. presented the TI-SC (top-k influential nodes selection based on community detection and scoring criteria in social networks) algorithm based on other node scoring ability, which reduces the overlap of seed nodes [33]. Also, TI-SC outperforms state-of-the-art algorithms in terms of influence spread. However, its time complexity is not acceptable for massive networks. CIM (community-based influence maximization) is a community-based influence maximization with three main steps [42]. CIM initially started with the community detection step. Next, the initial candidate is selected based on discovered communities. Finally, the seed set is selected between candidate nodes. This algorithm has less quality than most of the other community-based algorithms, but it is a bit fast. INCIM is another community-based approach to solve the influence maximization problem [62]. After the community detection step, INCIM calculates each node's influence as a combination of its local and global influences. The influence spread of each node is locally calculated locally in its own community. It uses the LT as a propagation model. Banerjee et al. proposed the ComBIM (community-based solution approach for solving the BIM problem) algorithm that has four steps [31]. At first, community detection is performed to obtain important community structures. Next, budget distribution to divide the total budget among the communities based on community size and density. After that, the initial seeds are selected, and then budget transfer among the community is done to utilize and finalize the seed set. The main drawback of this algorithm is its high time complexity in comparison to the moderate quality of the influence spread. C2IM (Community-based Context-aware Influence Maximization) is a new community-based approach to improve the time-complexity in the IM problem [12]. In C2IM, after the community detection step, it proposes a non-desirable nodes Finder (NDF) technique to identify non-desirable nodes. In the end, they used a newly defined measure to identify influential seed nodes.

3. Proposed algorithm

In this section, we introduce the framework of the LMP algorithm in detail. This algorithm can locally identify the influence spread. The LMP is divided into three main phases:

1. Node labeling based on Node Influence Power (*NIP*)
2. Create a sub-graph and select influential candidate nodes
3. Select final seed set from the candidate set

Phase 1: The proposed algorithm (LMP) uses a local travel form node based on their influence power. The node's influence power (*NIP*) is calculated based on the topological features by Eq. (1). Three essential features are used for computing the node's influence power: the degree, clustering coefficient, and the common neighbors. Based on the *NIP* measure, nodes with a high degree and low common neighbors have high influence power. These nodes can be hubs or semi-hubs, and they are placed in a strategic position. In the LMP algorithm, the travel starts with the nodes with the least influence power. The main idea is that improving the diffusion on social networks depends on the fact that information should always be spread to the nodes with high influence power.

$$NIP\,(v_i) = \deg\,(v_i) - \left(|\Gamma_{v_i} \cap \Gamma_{v_{j,max}}|\right)(1 - CC_{v_i}) \tag{1}$$

where $\deg\,(v_i)$ is the degree of node v_i. high-degree nodes can have higher influence power.Γ_{v_i} are the neighbors of the node v_i, $v_{j,max}$ is the node with the highest degree among the neighbors of the node v_i, $\Gamma_{v_{j,max}}$ are the neighbors of the node $v_{j,max}$, CC_{v_i} is the clustering coefficient of the node v_i. In fact, by Eq. (1), we are looking for nodes with a special position among the important nodes.

Generally, nodes with high degree are not necessarily suitable for diffusion; and their strategic location should be considered for diffusion. So, the clustering coefficient (CC_{v_i}) is considered as a measure to express their posiion. That is why the high clustering coefficient indicates that the node is in the dense region. Experiments proved that the nodes with

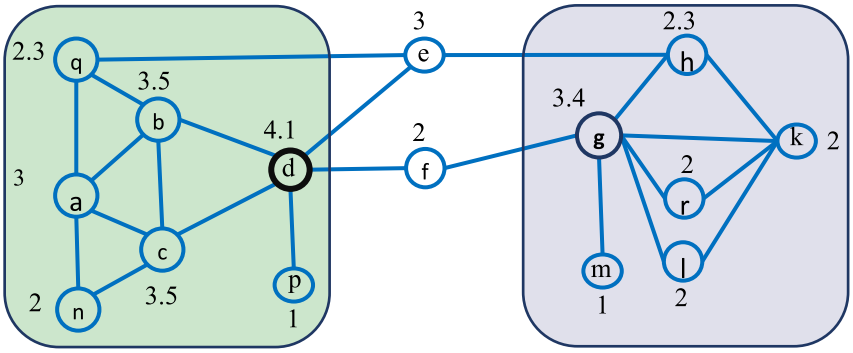


Fig. 1. An example network.

high clustering coefficient necessarily do not have specific strategic position to influence spread because its neighbors in dense part may have similar influence ability. Common neighbors also are important measure in spreading. If two nodes have a lot of common neighbors, they have many overlaps and close diffusion with each other. But, the proposed LMP algorithm considers common neighbors between node v_i and $v_{j,max}$ (e.g $v_{j,max}$ is the node with the highest degree among the neighbors of the node v_i). Consequently, for two nodes node v_x and node v_y with higher degrees than others, if a node v_x and its important neighbor has higher overlap Common neighbors than a node v_y in a same condition, it verifies that node v_x are in better position for influence spread. Because node v_y their important neighbors have close influence ability whereas the node v_x due to having higher degree and low overlapped common neighbors has higher influence ability than its neighbors. Needless to mention that LMP algorithm does not need to consider common neighbors between all v_i's neighbors. Based on the Eq. (1), a high degree node with low number of overlapped common neighbors with its important neighbor has higher spreading ability. Therefore, a combination of these three measures revealed a good ranking of high influence nodes.

For example, nodes with gatekeeper property are important among important nodes and play a significant role in influence spreading. If these nodes are selected as a seed, they can activate numerous nodes in less time. Fig. 1 shows an example network with 15 nodes. In this network, the *NIP* value is computed for each node. The node g has the highest degree in the network, and most of the previously provided algorithms select the node g as a seed node. But the proposed LMP algorithm gives the best importance for node d based on Eq. (1) because $NIP(d) > NIP(g)$. As we have shown in Fig. 1, node g and its important neighbor (node k) has three overlap nodes and it verifies that their influence close together. But node d only has one common neighbor, only one overlap, with its important neighbor (node b or c). It proves that node d has better position than node g. Node d also is the second-best node based on degree property but located between in a strategic position. Because node d as an important node has higher protentional to infect the gray section of the network using e or f nodes, node g is a core node in the network's periphery. Such nodes mostly cannot have optimal influence spread. Node g only has one chance to infect the green section. Therefore, node d has a higher potential to select as a candidate node.

The LMP algorithm uses node labeling steps to reduce search space significantly. This algorithm is one of the fastest influence maximization algorithms among well-known fast algorithms. The purpose of the label is the *flag* field with three values 0, 1, and 2. Therefore, labels can be shown as $flag = 0$, $flag = 1$, and $flag = 2$. All nodes are initially labeled with $flag = 0$. The LMP algorithm helps us to improve the influence spread and optimize information diffusion on social networks. The steps of the LMP algorithm are described step by step using Algorithm1 to 3 as follows. At first, Algorithm 1 is performed on the input network to label all nodes based on their *NIP* value. The steps of Algorithm 1 are shown below.

Algorithm 1: Labeling nodes based on the Node Influence Power (NIP)

Input: Network G (V, E)
Output: Network with labeling nodes G_f (V, E, Flag)

1 - First, calculate the Node Influence Power (NIP) for all nodes using the equation (1).
2 - For each node v_i, consider a $flag$ with an initial value of 0 ($flag_{v_i} \leftarrow 0$).
3- **While** there is a node with flag = 0, **repeat** the following operation.
4- Select a node v_x from the list with $flag_{v_x} = 0$.
5- For all neighbors of the node v_x, if $NIP(v_x) > NIP(v_j)$ then set $flag_{v_j} = 1$ where v_j is a neighbor of the node v_x.
6- Among the neighbors of the node v_x, find the node v_y with the largest NIP value where $NIP(v_y) \geq NIP(v_x)$. If no such v_y node is found, update the flag of the node v_x equal to 2 ($flag_{v_x} \leftarrow 2$)
Else:
   A. If $flag_{v_y} = 2$, then update the flag of the node v_x equal to 1 ($flag_{v_x} \leftarrow 2$) and go to step 3 to select a new node (with $flag_v = 0$).
   B. Otherwise, update the flag of the node v_x to $flag_{v_x} \leftarrow 1$ and go to step 7.
7- If a node v_y is found for all neighbors of the node v_y, if $NIP(v_y) > NIP(v_j)$ then set $flag_{v_j} = 1$ where v_j is a neighbor of the node v_y. Then from this node v_y look for a node v_z with the largest NIP value where $NIP(v_z) \geq NIP(v_y)$. If no such v_z node is found, update $flag_{v_y} \leftarrow 2$.
But if there is such a node v_z, then:
   A. $flag_{v_y} \leftarrow 1$.
   B. If $flag_{v_z} = 2$, then go to step 3.
   C. else go to step 8.
8- $flag_{v_z} \leftarrow 2$.
9- For all neighbors of the node v_z, if $NIP(v_z) > NIP(v_j)$ then set $flag_{v_j} = 1$ where v_j is a neighbor of the node v_z.
10- **End while**
11- return G_f

The nodes with $flag = 0$ have the lowest influence. Also, most nodes with $flag = 1$ have low influence spread, and nodes with $flag = 2$ are the nodes with a higher influence spread than others, and these nodes are added to the influential candidate set in the next step because they have a higher potential to select as final seed nodes.

Phase 2: After executing Algorithm 1, the label of all nodes is identified. In the next step, the most important nodes are assigned to the candidate set. Each node in the candidate set may have the required potential to select as the final seed. Selecting the influential candidate nodes has a significant effect on reducing the search space and computational overhead. Algorithm 2 shows the steps of selecting the candidate set.

Algorithm 2: Select the candidate set

Input: Network with labeling nodes (G_f)
Output: Candidate nodes (as a sub-graph *G'* with connected nodes)

1- assign all nodes with $flag_v = 2$ to candidate set.
2- select top *2k* nodes with the highest NIP among nodes with $flag_v = 1$ and add them to a temporary set.
3- From a temporary set, select *k* nodes with the *lowest clustering coefficient* and add them to the candidate set. The *k* shows the number of seed nodes.
4- Create the sub-graph *G'* with only one component. To make just one component, some middle nodes also is added to the sub-graph as connector nodes among candidate nodes.
5- Return subgraph *G'*.

Phase 3: Now, a sub-graph is obtained with *p* candidate nodes. Therefore, the search space is considerably reduced. Candidate nodes potentially have a higher influence than other nodes. Therefore, the *k* node is selected from the candidate set as the final seed node at the final step. In the sub-graph, some connector nodes with $flag = 1$ have a chance to be selected as seeds due to their special position in the graph. It is essential to mention that the number of candidate nodes is a small value. For example, with assuming $k = 50$, in a dataset with more than 150,000 nodes, the candidate set has less than 500 nodes, or in a dataset with more than 50,000 nodes, less than 200 nodes are selected for the candidate set. It is proved that the seed set is quickly finalized by computing the importance of candidate nodes using Eq. (3) and ranking them to select final seed nodes. Needless to mention, all required computing metrics for Eq. (3) are local metrics. The final seed nodes are selected by Algorithm 3 below.

Algorithm 3: Select the final seed nodes

Input: Candidate set
Output: Seed set

1- Set $S_{c,i} \leftarrow NIP(v_{c,i})$ //It was previously computed by equation (1)
2- $S_{c,i} \leftarrow S_{c,i} + \sum_{\forall v_{c,j} \in \Gamma_{v_{c,i}}} NIP_{v_{c,j}}$, where $v_{c,i}$ is the node v_i in candidate set and $v_{c,j}$ is its neighbors.
3- $AVG_{c,i} = S_{c,i}/(\deg(v_{c,i}) + 1)$
4- Compute the squares clustering coefficient (Sq) using equation (2) for node $v_{c,i}$ [63].

$$Sq(v_{c,i}) = \frac{\sum_{u=1}^{deg(v_{c,i})} \sum_{w=u+1}^{deg(v_{c,i})} q_{v_{c,i}}(u,w)}{\sum_{u=1}^{deg(v_{c,i})} \sum_{w=u+1}^{deg(v_{c,i})} \left[a_{v_{c,i}}(u,w) + q_{v_{c,i}}(u,w)\right]}, \quad (2)$$

And $a_{v_{c,i}}(u,w) = \left(deg_{(u)} - \left(1 + q_{v_{c,i}}(u,w) + \theta_{uv_{c,i}}\right)\right)\left(deg_{(w)} - \left(1 + q_{v_{c,i}}(u,w)\right) + \theta_{v_{c,i}w}\right)$

5- Set $T_{v_{c,i}}$ with the number of relations among the neighbors of the node $v_{c,i}$.
6- Calculate the equation (3) to obtain the importance of a candidate node $v_{c,i}$ ($I(v_{c,i})$).

$$I(v_{c,i}) = \deg(v_{c,i}) - \frac{e^{Sq(v_{c,i})} \times T_{v_{c,i}}}{AVG_{c,i}} \quad (3)$$

where e is Neper number.

7- Rank the candidate nodes based on $I(v_{c,i})$ value.
8- Select top k nodes with the highest rank from the list as final seed nodes.

In Eq. (3), the square clustering coefficient is used, which is a local measure and computed by Eq. (2) (see [63]). In Eq. (2), u *and* w are neighbors of the node $v_{c,i}$ and $q_{v_{c,i}}(u, w)$ is the number of common neighbors of u and w except for node $v_{c,i}$. For each pair (u,v), if u and v are connected, $\theta_{uv} = 1$ otherwise, $\theta_{uv} = 0$. After executing Algorithm 3, the k seed nodes are identified. It is necessary to mention that this step is very fast, and the required running time for executing algorithm 3 is less than 1 s because, in large-scale datasets with several hundred thousand nodes, the selected number of nodes for the candidate set is less than 1% of all nodes.

To clarify the steps of the proposed algorithm and make it easier to understand, an example is given step-by-step on a small network. Fig. 2(a) shows an example network with 20 nodes. In this network, a tuple (*NIP*, *flag*) is provided for each node. For all nodes, an array list of nodes is created, and the initial value of the flag field is 0 ($flag_V \leftarrow 0$). It is necessary to mention that each node can be considered as a start node for traveling. In Fig. 2(b), the node v_{19} is randomly selected as the first node to travel. In the array list of Fig. 2(b), the node v_{19} is colored with purple. At the first step of traveling, the node v_{19} must select a proper neighbor to travel. Thus, the node v_1 is selected to travel from node v_{19} because the node v_{19} has only one neighbor (v_1) and *NIP* (v_1) > *NIP* (v_{19}). Therefore, the *flag* of the node v_{19} is changed from 0 to 1. At the second step of traveling, the best neighbor of the node v_1 must be chosen. The nodes v_2, v_4 and v_{18} are neighbors of the node v_1 and since their *NIP* are lower than the *NIP* of node v_1, the flag of the node v_1 is set to 2, and the *flag* of nodes v_2, v_4 and v_{18} is changed from 0 to 1, and traveling is terminated for this node. Therefore, a new node with $flag = 0$ is selected to start a new traveling. Since the *flag* of nodes v_1, v_2, v_4, v_{18} and v_{19} are 1 or 2, therefore they cannot be selected for traveling. These nodes are colored in an array list of Fig. 2(c), and it means that they cannot be selected as starter nodes.

As shown in Fig. 2(c), the node v_{20} is selected to start a new traveling. At the first step of traveling, the node v_9 with larger *NIP* than node v_{20} is selected to travel, and the *flag* of the node v_{20} is set to 1. At the next step, among the neighbors of the node v_9 (nodes v_8 and v_{10}), the node v_8 has larger *NIP* than node v_{10} and also *NIP* (v_8) > *NIP* (v_9), so the *flag* of the node v_9 is set to 1, and its all neighbors with lower *NIP* than *NIP* (v_9) change their *flag* value to 1. Thus, the *flag* of the node v_{10} is also changed to 1. Since we only have permission to travel two steps from the starter node, the traveling is terminated at the node v_8 and the *flag* of node 8 is consequently set to 2.

Fig. 2(d) shows that the next starter node between the nodes with $flag = 0$ is the node v_6. At the first step of traveling from node v_6, since the node v_5 has the highest *NIP* among the neighbors of the node v_6 and *NIP* (v_5) > *NIP* (v_6), the *flag* of the node v_6 changes from 0 to 1. At next, because the node v_5 has higher *NIP* than its neighbors, its flag changes to 2, and the *flag* of its neighbors with lower *NIP* than *NIP* (v_5) is also set to 1. Therefore, the flag of nodes v_3, v_4 and v_7 are changed to 1. Needless to mention that the *flag* of the node v_4 was previously changed to 1, and it is not necessary to rechange.

Fig. 2(e) expressions that the new starter node between the remaining nodes with $flag = 0$ is the node v_{11}. Because it has a higher *NIP* than its neighbor nodes, its *flag* value is set to 2, and the *flag* of all its neighbors is set to 1. Therefore, the *flag* of nodes v_{12}, v_{13}, v_{14}, and v_{16} are changed to 1. Finally, in Fig. 2(f), the last starter node between the remaining

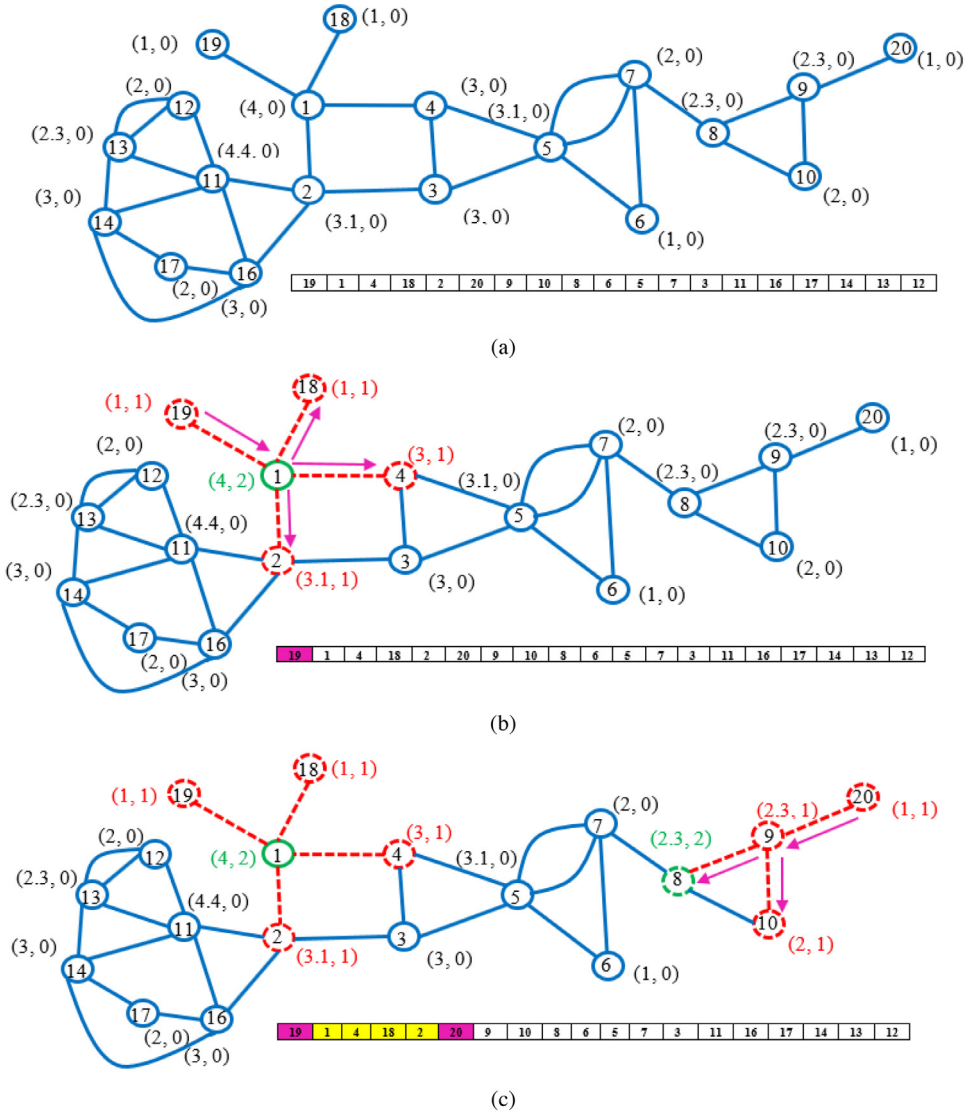


Fig. 2. An example network for labeling nodes based on Algorithm 1.

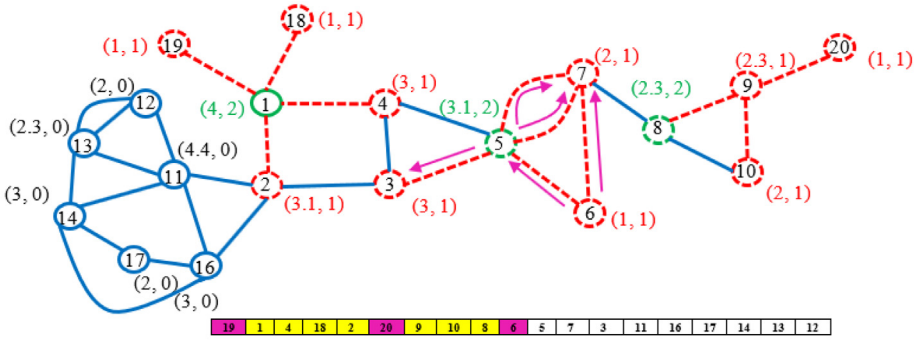
nodes is the node v_{17}. At the first step of traveling from node v_{17}, since the node v_{17} has not any neighbors with *flag* $= 0$, and its *NIP* is lower than the nodes v_{14} or v_{16}, then its *flag* is changed to 1, and the traveling step is terminated.

3.1. Time complexity

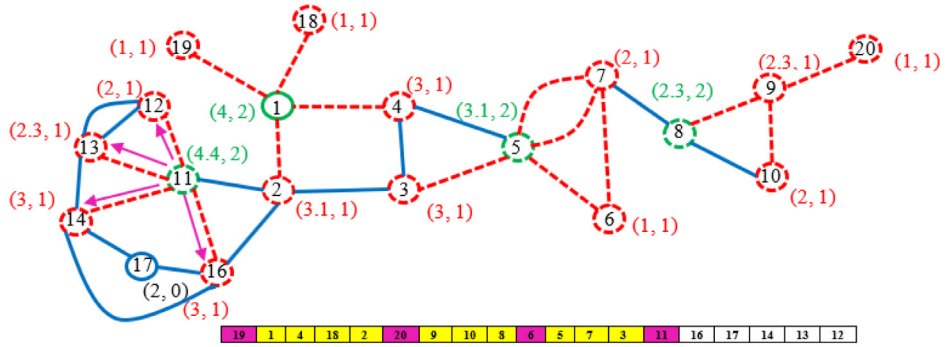
In this section, we analyze the time complexity of the LMP algorithm based on Algorithms 1, 2, and 3. First, we have applied the Algorithm 1 for labeling nodes based on the Node Influence Power (NIP) measure, which requires $O(kn)$ time complexity where n is the number of nodes ad k is the average degree. In next step, Algorithm 2 is implemented for selecting the candidate set. therefore, it requires $O(n)$ time complexity. Finally, we have used the Algorithm 3 for selecting the final seed set. This algorithm has a time complexity of $O(n^{'})$ where $n^{'}$ is the number of candidate nodes and $n^{'} \ll n$.. So, the total time complexity of the LMP algorithm is $O\left(kn + n + n'\right) = O(kn)$. Table 1 shows the time complexity of LMP algorithm in comparison to other fast methods.

4. Performance evaluation

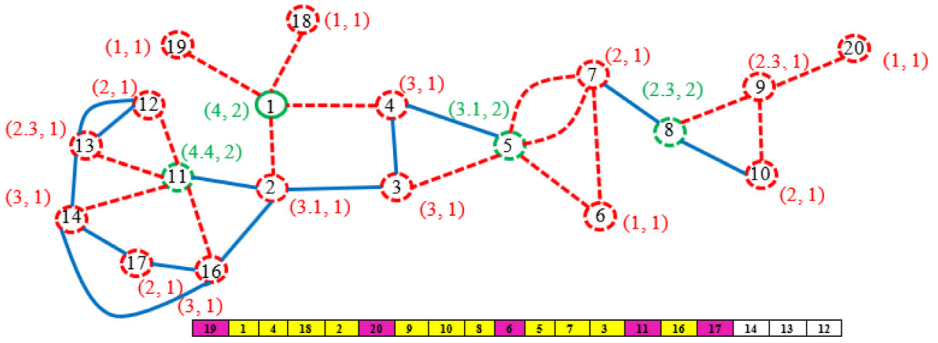
In this section, the performance of the LMP algorithm is evaluated on twelve synthetic real-world networks. We analyze the influence spread as well as the required running time by comparing the LMP algorithm with other state-of-the-art algorithms.



(d)



(e)



(f)

Fig. 2. (*continued*).

4.1. Dataset

We consider several popular social networks, the following ten popular networks and two synthetic networks for our comparisons.

- **Route views**: This dataset describes routers comprising the Internet that can be organized into sub-graphs. This network consists of 6474 nodes and 13 895 edges [64].
- **PGP**: This network is an unweighted and undirected that describes the interaction network of persons of the Pretty Good Privacy algorithm for secure information interchange [65]. Also, this network consists of 10 680 nodes and 24 316 edges.
- **Sister cities**: The sister cities network is an undirected and unweighted network of cities of the world connected by " twin city" relationships, as extracted from WikiData [64]. This network consists of 14 274 nodes and 20 573 edges.

Table 1
Time complexity of the proposed LMP and other different algorithms (m is number of edges ($m' \ll m$), k is the average degree of network, R is the number of repetitions of the Monte Carlo simulation, q is number of reparations for the ranking to get stable, $\overline{D}$ is average node degree in a graph.).

Algorithm	Time complexity
LMP	$O(n + 2k + n')$
CI	$O(m)$
VoteRank	$O(m + klogn + k\frac{m^2}{n^2}R)$
LIR	$O(m + kn'R)$
TI-SC	$O(n \log n + m + Rn + (k-1)Rn')$
PHG	$O(nlogn + n + kn'm')$
ProbDegree	$O(klogn + m)$
HEDVGreedy	$O(mk^2\overline{D})$
SRFM	$O(m + n')$
RNR	$O(kq\,(m + nlogn))$

Table 2
Summary of specific statistical parameters of synthetic and real networks.

Networks	Nodes	Edges	Average degree	Assortativity	Density
Route views	6474	13,895	4.29	−0.17	6.631E−4
PGP	10,680	24,316	4.55	0.23	4.264E−4
Sister cities	14,274	20,573	2.88	0.38	2.019E−4
As-22july06	22,693	48,436	4.21	−0.19	1.837E−4
CAIDA	26,475	53,381	4.03	−0.19	1.523E−4
COND-MAT (1995–2005)	39,577	175,691	8.87	0.18	2.243E−4
RO	41 773	125 826	6.02	0.11	1.442E−4
HU	47 538	222 887	9.37	0.20	1.972E−4
HR	54 573	498 202	18.25	0.19	3.345E−4
Douban	154,908	327,162	4.22	−0.18	2.7267E−5
M-FO115	10,000	23,142	4.62	0.18	4.628E−4
M-FO120	10,000	29,566	5.91	−0.03	5.913E−4

- **As-22july06**: The As-22july06 is a symmetrized snapshot of the Internet structure at the level of autonomous systems for July 22, 2006, that this network consists of 22 693 nodes and 48 436 edges [64].
- **CAIDA**: The CAIDA is the network of autonomous systems from the CAIDA project for January 2004 to November 2007. This network consists of 26 475 nodes and 53 381 edges [64].
- **COND-MAT (1995–2005)**: The COND-MA (1995–2005) co-authorship network is the author's papers submitted to Condense Matter Physics that the data includes papers in the period between 1995-01-01 and April 2005-03-3 [64]. Also, this network consists of 39 577 nodes and 175 691 edges.
- **RO**: The RO was collected from the music streaming service Deezer from Romania that nodes indicate the persons and edges are the friendships [66]. Also, this network consists of 41 773 nodes and 125 826 edges.
- **HU**: The HU was collected from the music streaming service Deezer from Hungary that nodes indicate the persons and edges are the friendships [66]. Also, this network consists of 47 538 nodes and 222 887 edges.
- **HR** The HR was collected from the music streaming service Deezer from Croatia that nodes indicate the persons and edges are the friendships [66]. Also, this network consists of 54 573 nodes and 498 202 edges.
- **Douban**: The Douban network is a Chinese social networking service that users can create content related to film, books, music, recent events, etc. [67]. This network consists of 154 908 nodes and 327 162 edges.
- **M-Fo115:** This synthetic network was created from the forest fire model using probability $p = 0.115$ [68].
- **M-Fo120:** This synthetic network was created from the forest fire model using probability p = 0.12 [68].

Also, several popular models generate synthetic networks. In this paper, these networks are generated by the forest fire model. For building synthetic networks by the forest fire model, a forward burning probability (p) is one parameter. v first chooses a node w uniformly at random and forms a link to w. Next, a random number x is generated that is binomially distributed with a mean $(1-p)^{-1}$. Node v selects x edges with a node with node w. This model has the power-law distribution property, which makes the Synthetic network closer to the real world. Moreover, Table 2 shows the specific features and parameters for both artificial and real-world networks.

4.2. Baseline methods

The LMP algorithm has been compared with nine state-of-the-art algorithms. The list of state-of-the-art algorithms is as follows.

- **CI** [26]: This algorithm finds the influential nodes via optimal Percolation.
- **VoteRank** [27]: This algorithm is based on voting ability that avoids the Rich-club phenomenon.

- **LIR** [28]: This algorithm selects influential nodes by a local index. It is a fast method but low efficiency in seed selection.
- **TI-SC** [33]: This algorithm is fast and reduces the overlap among seed nodes.
- **PHG** [11]: This community-based algorithm selects the most potential influence nodes by the influence weight of nodes.
- **ProbDegree** [53]: This algorithm selects seed nodes based on the number of neighbors and the two-hop away neighbors' effect.
- **HEDVGreedy** [34]: This algorithm selects seed nodes based on the Greedy algorithm.
- **SRFM** [55]: This algorithm selects influential nodes based on shell-based ranking.
- **RNR** [32]: This algorithm selects seed nodes based on the reversed rank information of a node.

4.3. Experiment setup

The LMP algorithm is executed on a desktop computer with 2.5 GHz Intel Core i5 CPU-3230M and 16 GB memory and 64-bit Windows 10 operating system using the python programming language. The independent cascade (IC) model and linear threshold model as the models of diffusion are considered for our experiments. In these experiments, the seed size of k is varied from 1 to 30, while influence probability is $p = 0.01$ for each edge. Also, the threshold and weight for each node are a random number between 0 and 1 in the linear threshold model. We execute the Monte-Carlo simulation on the independent cascade and linear threshold models with 5000 repetitions for each seed set and take the average influence spread on all datasets.

4.4. Performance analysis

The experimental results of the LMP and other comparison algorithms are compared and discussed in terms of influence spread and running time on the same conditions. The influence spread is calculated according to the following equation for node v_i:

$$Influence\ spread_{v_i} = \frac{\sum_{j \in R} A_j}{R} \tag{4}$$

where R is number of Monte Carlo simulation repetitions, and A is the number of nodes activated by the v_i node in each iteration of the Monte Carlo simulation. Fig. 3 depicts the influence spread with the change of seed nodes based on the IC model in real networks. Fig. 3(A) shows the influence spread in the Route views dataset. The influence spread results in Fig. 3(A) indicate that LMP produces the best influence spread. Also, the results of the HEDVEGreedy, TI-SC and PHG algorithms are close to the LMP algorithm. We can find that the influence spread of ten algorithms constantly increases with seed nodes, and the influence spread of LMP and PHG is larger than others. Especially, LIR and ProbDegree have the worst results in both PGP and Route views datasets. In Fig. 3(B), the influence spread of all compared algorithms is shown in the PGP dataset. The influence spread results in Fig. 3(B) shows that LMP performs quite well, and LIR has a much worse influence spread. Fig. 3(C) shows the influence spread in the Sister cities dataset with the increase of seed nodes. The influence spread results in Fig. 3(C) specify that LMP outperforms other compared methods. In addition, VoteRank and TI-SC nearly have the same performance as the second-best rank, and LIR has the worst results. Fig. 3(D) illustrates the influence spread in the As-22july06 dataset. When k is 1 to 15, we see that the influence spread values of the LMP, VoteRank, HEDVGreedy, SRFM, and TI-SC algorithms are nearly the same. However, the influence spread of LMP is better than others in k = 10 to k = 30. Fig. 3(E) and (F) show the influence spread in CAIDA and COND-MAT (1995–2005) dataset with the increase of seed nodes. The influence spread results in Fig. 3(F) and (G) shows that the influence spread values of the LMP algorithm are larger than other compared algorithms. Also, the performance of SRFM, HEDVGreedy, TI-SC, PHG, and VoteRank algorithms are close to the LMP algorithm, but LIR considerably has low performance in these datasets. Fig. 3(G) and (H) show the influence spread of all compared methods in the RO and HU with the increase of seed nodes. The influence spread results in Fig. 3(G), and (H) reveal that LMP outperforms the compared algorithm. However, PHG, VoteRank, CI, and TI-SC have nearly the same performance. In Fig. 3(K) and (L), LMP, like other datasets, has the best rank in terms of influence spread on HR and Douban datasets. However, in both datasets, CI and LIR methods averagely have the lowest influence spread values. Consequently, on all datasets in Fig. 3, LMP can identify the proper seed sets with higher quality in terms of influence spread value.

Fig. 4 depicts the influence spread with the change of seed nodes based on the LT model in real networks. Fig. 4(A) shows the influence spread in the Route views dataset. The influence spread results in Fig. 4(A) indicate that LMP produces the best influence spread. Also, the results of the HEDVGreedy and PHG algorithms are close to the LMP algorithm. We can find that the influence spread of ten algorithms constantly increases with seed nodes, and the influence spread of LMP and HEDVGreedy is larger than others. In Fig. 4(B), the influence spread of all compared algorithms is shown in the PGP dataset. The influence spread results in Fig. 4(B) shows that LMP performs quite well, and ProbDegree has a much worse influence spread. Fig. 4(C) shows the influence spread in the sister cities dataset with the increase of seed nodes. The influence spread results in Fig. 4(C) specify that LMP outperforms other compared methods. CI and LIR have the worst results. Fig. 4(D) illustrates the influence spread in the As-22july06 dataset. When k is 1 to 5, we see that the influence

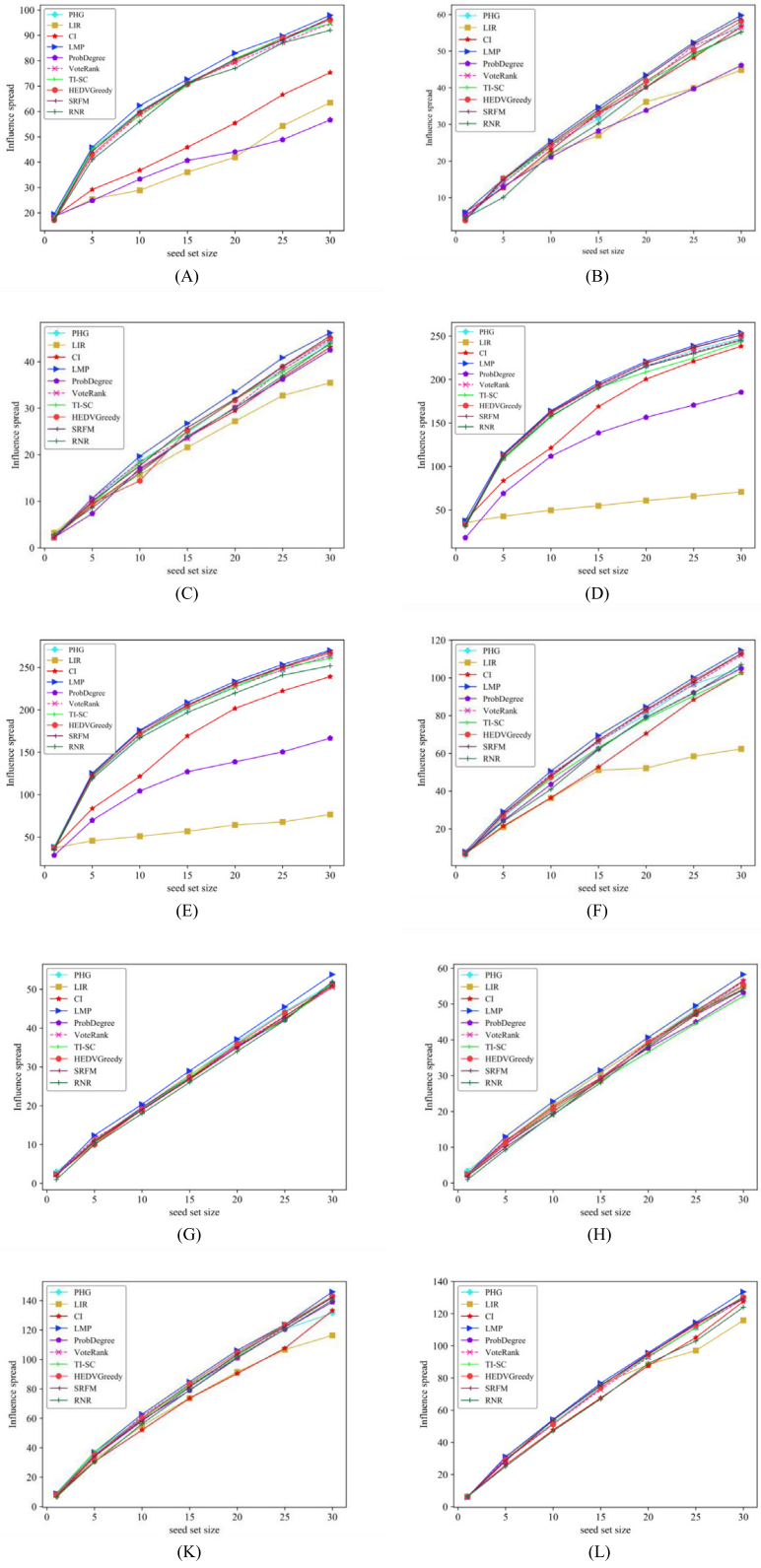


Fig. 3. Influence spread comparison in several real-world datasets based on the IC model. (A) Route views; (B) PGP; (C) Sister cities; (D) As-22july06; (E) CAIDA; (F) COND-MAT (1995–2005); (G) RO; (H) HU; (K) HR; and (L) Douban dataset.

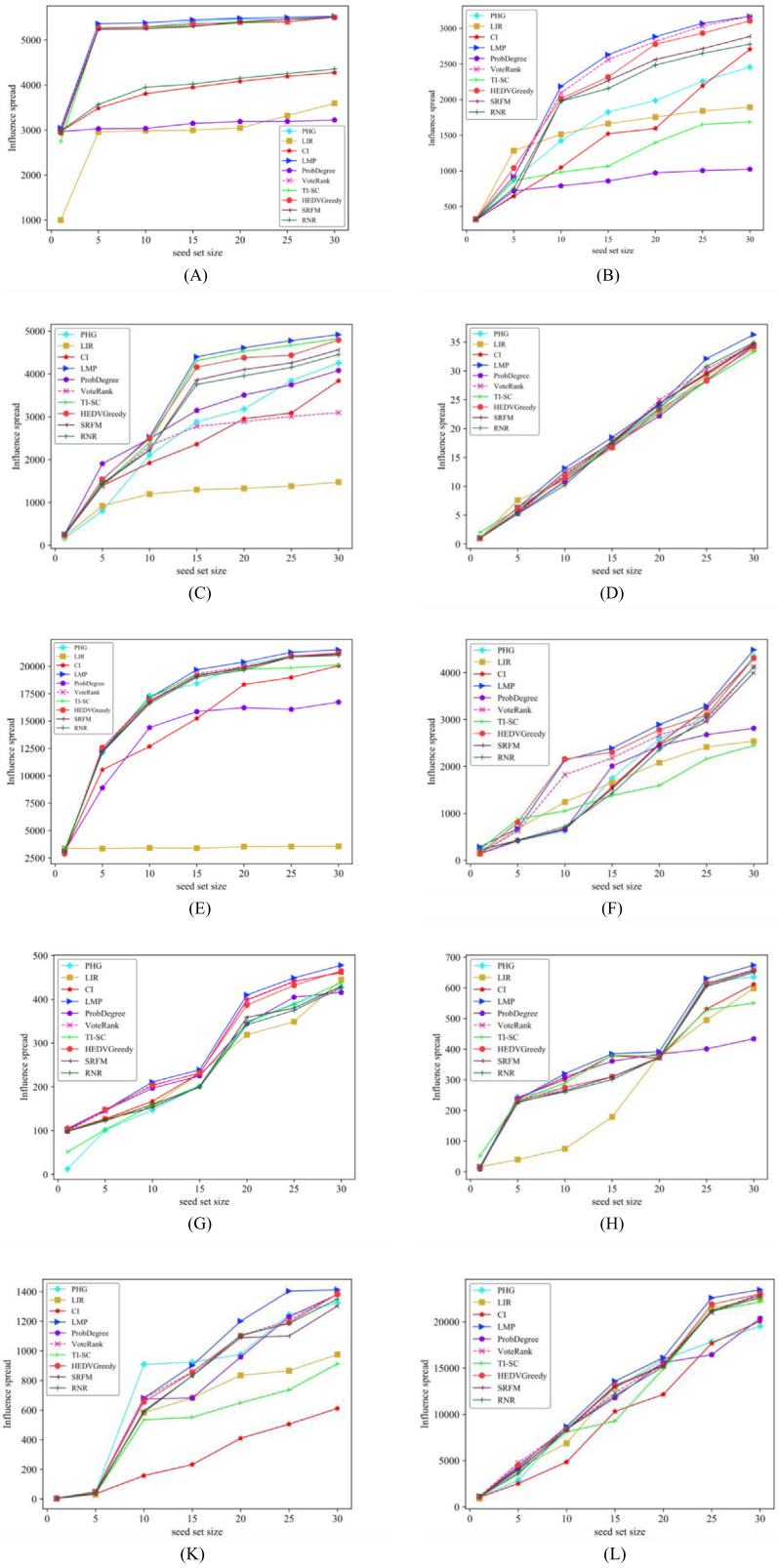


Fig. 4. Influence spread comparison in several real-world datasets based on the LT model. (A) Route views; (B) PGP; (C) Sister cities; (D) As-22july06; (E) CAIDA; (F) COND-MAT (1995–2005); (G) RO; (H) HU; (K) HR; and (L) Douban dataset.

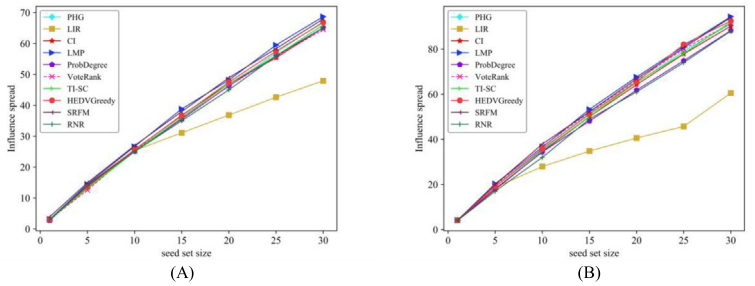


Fig. 5. Influence spread comparison in synthetic datasets under IC model (A) M-FO115; (B) M-FO120.

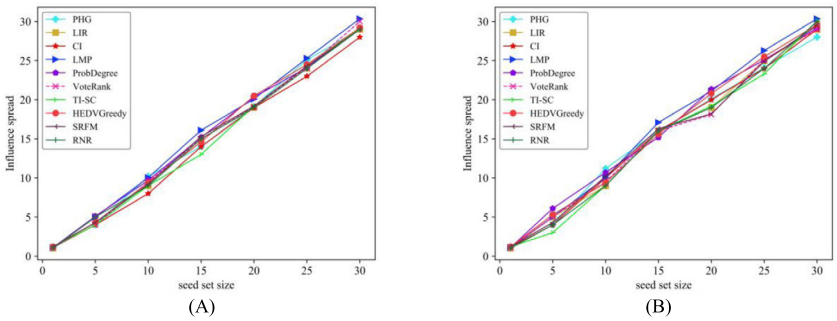


Fig. 6. Influence spread comparison in synthetic datasets under LT model (A) M-FO115; (B) M-FO120.

spread values of the LMP, VoteRank, and HEDVGreedy algorithms are nearly the same. However, the influence spread of LMP is better than others in k = 10 to k = 30. Fig. 4(E) and (F) show the influence spread in CAIDA and COND-MAT (1995–2005) dataset with the increase of seed nodes. The influence spread results in Fig. 4(E) and (F) shows that the influence spread values of the LMP algorithm are larger than other compared algorithms. Fig. 4(G) and (H) show the influence spread of all compared methods in the RO and HU with the increase of seed nodes. The influence spread results in Fig. 4(G), and (H) reveal that LMP outperforms the compared algorithm. However, HEDVGreedy has nearly the same performance. In Fig. 4(K) and (L), LMP like other datasets, has the best rank in terms of influence spread on HR and Douban datasets. However, in both datasets CI has the lowest influence spread values. Consequently, on all datasets in Fig. 4, LMP can identify the proper seed sets with higher influence spread in the LT model.

Fig. 5 depicts the influence spread with the change of seed nodes in synthetic networks based on the IC model. Fig. 5(A) and (B) show the influence spread in the M-FO115 and M-FO120 datasets with the increase of seed nodes. The influence spread results in Fig. 5(A) show that all compared methods have the same performance in k = 1 to 15. However, LMP has a higher performance than other compared methods for k > 15 in both datasets. In addition, the LIR has the lowest performance in both datasets. Because the seed nodes selected by the LIR algorithm have the least influence spread. Fig. 6 depicts the influence spread with the change of seed nodes in synthetic networks based on LT model. Fig. 6(A) and (B) show the influence spread in the M-FO115 and M-FO120 datasets with the increase of seed nodes. The influence spread results in Fig. 6(A), and (B) show that LMP has a higher performance than other compared methods for k > 15 in both datasets.

Table 3 shows the speedup % (in terms of influence spread) of the LMP algorithm over other state-of-the-art algorithms. The speedup is calculated using Eq. (5) as follow:

$$Speedup = ((A - B)/A) \times 100 \tag{5}$$

where A and B are two compared methods.

The experimental results in Table 3 are obtained for three different numbers of seeds (for k = 10, 20, and 30). The evaluation of the speedup measure reveals that The LMP algorithm strangely has the best speedup versus PHG, TI-SC, VoteRank, CI, LIR, HEDVGreedy, SRFM, RNR, and ProbDegree.

On the other hand, we have compared the average influence spread for k = 1 to 30 to show the effectiveness of the proposed LMP algorithm. Table 4 indicates the average influence spread per seed on real-world and synthetic networks based on the IC model. As shown in Table 4, the LMP algorithm averagely has a better influence spread than other algorithms. of course, the PHG algorithm in some datasets such as Route views, As-22july06, and PGP has the second rank its result is close to LMP. In addition, TI-SC has the second rank in RO and HR datasets, and VoteRank has the second rank in Sister cities, CAIDA, COND-MAT (1995–2005), and M-Fo120 datasets. Finally, ProbDegree has the second rank in

Table 3
Speedup % for LMP versus other methods (in terms of influence spread) on artificial and real-world datasets. All experiments are based on the IC model (PD indicates the PorbDegree algorithm, VR indicates the VoteRank algorithm, HG indicates the HEDVGreedy algorithm).

Data sets	Seed size	LMP vs. other algorithms																	
		LMP	PHG	LMP	LIR	LMP	CI	LMP	PD	LMP	VR	LMP	TI-SC	LMP	HG	LMP	SRFM	LMP	RNR
Route views	10	**3.7**	−3.5	**115.7**	−53.6	**69.2**	−40.9	**87.1**	−46.5	**6.1**	−5.7	**6.8**	−6.4	**4.9**	−4.6	**4.1**	−4	**11.7**	−10.1
	20	**3.7**	−3.5	**97.8**	−49.4	**49.7**	−33.2	**88.3**	−46.8	**4.7**	−4.5	**2.6**	−2.6	**3.9**	−3.8	**3.05**	−2.9	**7.7**	−7.1
	30	**1.8**	−1.8	**84.2**	−35.1	**30**	−23	**72.9**	−42.1	**3.4**	−3.3	**3.4**	−3.3	**1.8**	−1.7	**1.1**	−1	**6.4**	−6
PGP	10	**4.3**	−4.1	**15.1**	−13.1	**10.8**	−9.4	**20.9**	−17.3	**6.5**	−6.1	**4.9**	−4.6	**4**	−3.8	**2.2**	−2.2	**15.8**	−13.6
	20	**4**	−3.8	**20.2**	−16.8	**8.3**	−7.7	**28.5**	−22.1	**6**	−5.7	**4.7**	−4.5	**4**	−3.8	**1.2**	−1.2	**8.1**	−7.5
	30	**4.4**	−4.2	**33.2**	−24.9	**5.5**	−5.5	**29.6**	−22.8	**5.8**	−5.5	**8**	−7.4	**2.7**	−2.7	**1.3**	−1.3	**8.3**	−7.7
Sister cities	10	**5.6**	−5.3	**25.2**	−20.1	**17.9**	−15.1	**15.1**	−13.1	**6.5**	−6.1	**7.8**	−7.2	**36.7**	−26	**11.6**	−10.1	**21.5**	−17.7
	20	**5.3**	−5	**23.1**	−18.8	**13.7**	−12.1	**11.1**	−10	**10.7**	−9.7	**5.1**	−4.8	**5.7**	−5.4	**4.7**	−4.5	**11.7**	−10.5
	30	**4.2**	−4.2	**30.3**	−23.2	**7.1**	−6.6	**8.7**	−8.7	**3**	−2.9	**5.7**	−5.3	**2.5**	−2.4	**1.4**	−1.4	**5**	−4.8
As-22july06	10	**1.8**	−1.8	**229.8**	−69.6	**35.2**	−26	**46.7**	−31.8	**1.7**	−1.7	**4.5**	−4.3	**1.5**	−1.4	**0.7**	−0.6	**3.8**	−3.7
	20	**2.4**	−2.4	**263.4**	−72.4	**10.2**	−9.2	**41.1**	−29.1	**2.3**	−2.2	**5.9**	−5.6	**1**	−0.9	**0.9**	−0.9	**2.7**	−2.7
	30	**2.5**	−2.5	**258.8**	−72.1	**6.5**	−6.1	**36.8**	−26.9	**3.1**	−3	**4.3**	−4.1	**0.93**	−0.9	**1.1**	−1	**3.5**	−.3.4
CAIDA	10	**3.6**	−3.5	**246.1**	−71.1	**44.9**	−31	**68.9**	−40.8	**3.3**	−3.2	**2.7**	−2.6	**2.4**	−2.3	**0.7**	−0.7	**5.5**	−5.2
	20	**2**	−1.9	**263.3**	−72.1	**15.7**	−13.6	**68.4**	−40.6	**2.8**	−2.8	**3.1**	−3	**1.5**	−1.5	**1.7**	−1.1	**6.2**	−5.8
	30	**2.3**	−2.3	**253.1**	−71.6	**12.9**	−11.4	**62.2**	−38.3	**2.5**	−2.5	**3.7**	−3.6	**1.2**	−1.2	**0.5**	−0.5	**7.2**	−6.7
COND-MAT (1995–2005)	10	**6.8**	−6.3	**39.4**	−28.2	**38.8**	−27.9	**16.1**	−13.9	**3**	−2.9	**9.1**	−8.3	**6.5**	−6.1	**4.9**	−4.7	**23.4**	−19
	20	**5**	−4.8	**62.1**	−38.3	**20**	−16.7	**7**	−6.5	**2.7**	−2.6	**8.3**	−7.7	**2**	−2	**1.6**	−1.6	**7.1**	−6.6
	30	**9.1**	−8.3	**83.8**	−45.6	**11.7**	−10.4	**9.2**	−8.4	**2.4**	−2.3	**11.8**	−10.5	**1.5**	−1.5	**1.6**	−1.6	**7.1**	−6.6
RO	10	**4.2**	−4	**3.9**	−3.7	**7.4**	−6.9	**3.8**	−3.7	**8.5**	−7.8	**5.5**	−5.2	**7.2**	−6.7	**7.2**	−6.7	**13.2**	−11.6
	20	**1.7**	−1.7	**4.1**	−3.9	**6.1**	−5.8	**5**	−4.7	**4.6**	−4.4	**3.1**	−3	**3.1**	−3	**6.1**	−5.7	**9.2**	−8.4
	30	**4.4**	−4.2	**5.3**	−5	**6.3**	−5.9	**4.4**	−4.2	**6.4**	−6	**4.3**	−4.1	**5.4**	7.2	**5.4**	−5.2	**3.4**	−3.3
HU	10	4.1	**-3.9**	**5**	−4.8	**6.9**	−6.4	**14.5**	−12.7	**14.9**	−13	**13**	−11.5	**9.7**	−8.8	**20.1**	−16.7	**18.8**	−15.8
	20	**3.3**	−3.2	**2.9**	−2.8	**4.3**	−4.1	**8**	−7.4	**3**	−2.9	**11.2**	−10.1	**2.7**	−2.6	**7.1**	−6.6	**5.7**	−5.4
	30	**5.4**	−5.1	**6.1**	−5.7	**3**	−2.9	**9.5**	−8.7	**3.4**	−3.3	**11.6**	−10.4	**5.4**	−5.1	**7.8**	−7.2	**7.4**	−6.9
HR	10	**3.6**	−3.4	**14.1**	−12.4	**20.5**	−17	**6.4**	−6	**1.7**	−1.7	**5.3**	−5.1	**4.4**	−4.2	**7.8**	−7.2	**11.6**	−10.4
	20	**4.8**	−4.6	**16**	−13.8	**17.3**	−14.8	**5.1**	−4.8	**4.9**	−4.7	**3.8**	−3.6	**1.5**	−1.4	**2.5**	−2.5	**4.6**	−4.4
	30	**10.9**	−9.8	**25.4**	−20.2	**9.5**	−8.7	**4.9**	−4.7	**2.3**	−2.2	**2.8**	−2.7	**2.1**	−2.1	**2.7**	−2.6	**4**	−3.8
Douban	10	**4**	−3.8	**1.4**	−1.4	**13.6**	−11.9	**0.7**	−0.7	**5.7**	−5.4	**0.9**	−0.9	**5.9**	−5.6	**0.6**	−0.6	**15.7**	−13.3
	20	**2.8**	−2.7	**8.3**	−7.7	**9.3**	−8.5	**0.7**	−0.7	**3.1**	−3	**2.3**	−2.3	**1.2**	−1.2	**1.1**	−1.1	**7.6**	−7
	30	**3.5**	−3.3	**15.2**	−13.2	**4.6**	−4.4	**2.4**	2.4	**3.4**	−3.3	**2**	−2	**2.5**	−2.5	**3.2**	−3.1	**7.7**	−7.1
M-FO115	10	**3**	−2.9	**30.5**	−23.4	**5.6**	−5.3	**6**	−5.7	**-0.6**	0.6	**4.2**	−4	**1.9**	−1.8	−3.7	**3.8**	**14**	−12.2
	20	**4**	−3.8	**66.6**	−39.9	**5.3**	−5	**9.4**	−8.6	**2.33**	−2.2	**3.5**	−3.4	**3.2**	−3.1	**1.1**	−0.3	**10.8**	−9.7
	30	**3.5**	−3.4	**56**	−35.9	**4.6**	−4.4	**6.9**	−6.5	**3.2**	−3.1	**2.5**	−2.5	**2**	−2	**0.3**	−0.3	**7.2**	−6.7
M-FO120	10	**3**	−2.9	**30.5**	−23.4	**5.6**	−5.3	**6**	−5.7	−0.6	**0.6**	**4.2**	−4	**1.9**	−1.8	−3.7	**3.8**	**14**	−12.2
	20	**4**	−3.8	**66.6**	−39.9	**5.3**	−5	**9.4**	−8.6	**2.3**	−2.2	**3.5**	−3.4	**3.24**	−3.1	**1.1**	−1.1	**10.8**	−9.7
	30	**3.5**	−3.4	**56**	−35.9	**4.6**	−4.4	**6.9**	−6.5	**3.2**	−3.1	**2.5**	−2.5	**2**	−2	**0.3**	−0.3	**7.2**	−6.7

Table 4
The average number of influence spread is based on the IC model (for k = 1 to 30 per seed).

Datasets	Algorithms									
	PHG	LIR	CI	LMP	ProbDegree	VoteRank	TI-SC	HEDVGreedy	SRFM	RNR
Route views	15.3	8.9	10.9	**15.81**	8.19	15.06	15.22	15.1	15.1	14.7
PGP	7.44	6.22	7.33	**7.9**	6.23	7.4	7.4	7.5	7.5	7
Sister cities	5.68	4.8	5.16	**6.2**	5.3	5.6	5.61	5.5	5.8	5.4
As-22july06	39.7	12.66	35.7	**40.9**	28.3	39.7	38.9	40.1	40.3	39.3
CAIDA	42.3	13.3	35.84	**43.55**	26.18	42.38	42.39	42.7	43.03	41.03
COND-MAT (1995–2005)	14.2	9.6	12.63	**15.3**	13.8	14.6	13.8	14.76	14.76	13.7
RO	6.4	6.36	6.22	**6.68**	6.29	6.29	6.3	6.31	6.25	6.1
HU	6.97	6.96	6.94	**7.25**	6.59	6.8	6.5	6.8	6.64	6.59
HR	18	16.1	16.5	**18.9**	18.06	18.3	18.5	18.58	18.28	17.86
Douban	16.5	15.5	15.5	**17.1**	16.7	16.4	16.6	16.67	16.73	15.33
M-Fo115	8.23	6.67	8.23	**8.8**	8.16	8.1	8.2	8.3	8.5	8.1
M-Fo120	11.54	7.7	11.28	**11.91**	10.9	11.6	11.4	11.67	11.75	10.83

Table 5
The average number of influence spread is based on the LT model (for k = 1 to 30 per seed).

Datasets	Algorithms									
	PHG	LIR	CI	LMP	ProbDegree	VoteRank	TI-SC	HEDVGreedy	SRFM	RNR
Route views	1178.3	663.3	892.33	**1192.2**	726.5	1174.7	1164.7	1172.3	1170.1	909.3
PGP	371.2	342.1	334.1	**505.9**	189.6	497.1	265.6	483.6	445.7	437.3
Sister cities	547.4	260.4	526.7	**767.1**	637.6	524.3	746.3	734.6	688.6	675.1
As-22july06	4.05	4.07	4.16	**4.38**	3.97	4.17	4.03	4.05	4.16	4.1
CAIDA	37 778.5	804.5	3269.8	**3846**	3043.5	3801.2	3717.5	3784.3	3766.5	3743.6
COND-MAT (1995–2005)	432.4	358.2	426.5	**538**	372.5	485.9	324.5	520.1	405.8	410.9
RO	54.1	57.6	64.1	**67.8**	61.2	65.9	56.3	65.7	58.2	57.4
HU	82.4	59.4	81.2	**88.4**	71.2	82.4	80.3	82.5	81.4	81.5
HR	181.03	132.3	65.3	**188.5**	166.2	176.2	114.3	174.6	165.3	170.3
Douban	2620.1	2815.6	2288	**2997.2**	2591.8	2877.4	2681.3	2908.7	2862.1	2799.2
M-Fo115	3.41	3.4	3.2	**3.6**	3.4	3.43	3.32	3.45	3.41	3.4
M-Fo120	3.48	3.46	3.46	**3.71**	3.6	3.46	3.39	3.58	3.48	3.43

only Douban Dataset, and LIR has the second rank only in HU datasets. CI does not have the best or second-best rank in any datasets. As shown in Table 5, the LMP algorithm averagely has a better influence spread than other algorithms based on the LT model. of course, the PHG algorithm in some datasets such as Route views, HR, and M-FO120 has the second rank its result is close to LMP. In addition, the HEDVGreedy algorithm has the second rank in RO, HU, CAIDA, COND-MAT (1995–2005), PGP, and HR datasets, and TI-SC has the second rank in the sister cities dataset.

4.5. Running time

As mentioned above, LMP has two main advantages: higher influence spread and the fast-running time. Tables 6 and 7 show the running time of different algorithms on real and synthetic networks based on the IC and LT models. In this experiment, the running time is computed as the required time for selecting $k = 30$ seeds. Due to the considerable time consuming of PHG, and CI, these algorithms are performed only one time. Other algorithms are repeated five times. The average of obtained results is illustrated in Table 4. In all datasets, the LMP algorithm significantly has the lowest running time. Although the LIR is a fast algorithm using local measures, LMP considerably outperforms LIR in all datasets. Needless to mention that LIR has the worst influence spread in 10 of 12 datasets. Consequently, the LMP algorithm has achieved the best and superior influence spread and running time in all evaluated networks.

5. Conclusions

In this paper, the fast and accurate algorithm is proposed to address the influence maximization problem, namely LMP. In the first step of LMP, the node's influence power is computed for nodes, and then a local travel is started for labeling nodes based on their influence power. In the second step, the most important nodes are assigned to the candidate set. At the third step, the final seed nodes are selected from the candidate set based on their diffusion capability and strategic position. LMP algorithm has two main advantages: (1) it is fast, and (2) its performance is higher than other compared methods. The experimental results verify that LMP can enhance efficiency and accuracy on all datasets. LMP algorithm is effective and promising for dealing with influence maximization in social networks due to its fast-running time and suitable accuracy capability. It is the fastest in all datasets with superior performance. In addition, it has the best average influence spread in all datasets. In future work, we are going to develop the LMP based on SPARC implement of parallel computing. Because it has a notable potential to find seed nodes with remarkable time complexity. Furthermore, we

Table 6
Running times on artificial and real-world networks based on the IC model (All times are in seconds).

Datasets	Algorithms									
	PHG	LIR	CI	LMP	ProbDegree	VoteRank	TI-SC	HEDVGreedy	SRFM	RNR
Route views	9856.5	3808	936 959.2	**892.6**	3654.3	9853.9	38 522.3	45,26.3	996.3	8526.2
PGP	517 678.5	267	42 148.35	**133.65**	133.8	446.25	18 932.25	456	257	233.5
Sister cities	746.25	141.1	211.65	**112.65**	268.4	371.85	1416.45	452.6	110.1	529.3
As-22july06	1 425 682	784.3	1 684 917.9	**293.55**	3348.6	3160.2	14 146.2	393.2	774.2	4,56.2
CAIDA	1 944 354.9	739.5	1 814 442.6	**308.7**	1366.35	3166.2	85 072.95	1266	839	2156
COND-MAT (1995–2005)	1 374 922.3	1135	524 715.15	**200.85**	2307.15	2445.3	113 420.4	1526	956	1489
RO	1 944 678	305.4	1 684 913.1	**256.2**	289.8	391.2	3869.7	310.2	389	399
HU	4 665 711.9	370.5	2 764 841.3	**290.7**	355.05	476.1	7460.1	586	856	956
HR	4 017 600.3	1205	3 499 311.9	**580.8**	1406.25	1982.4	13 129.2	1566	2564	596.2
Douban	1 036 943.4	**1283**	90 720	1387.9	1404.75	1581.75	6584.7	1656	1185	1895
M-Fo115	370 370.25	590.8	39 420.9	**346.5**	1195.65	1029.6	679.05	456	652	985
M-Fo120	825 492.5	904.7	65 406.25	**850.8**	1806.35	1967.35	961.1	1256	901.5	1340

Table 7
Running times on artificial and real-world networks based on the LT model (All times are in seconds).

Datasets	Algorithms									
	PHG	LIR	CI	LMP	ProbDegree	VoteRank	TI-SC	HEDVGreedy	SRFM	RNR
Route views	24 641	9521	2 342 398	**2231**	9135.75	24 634.75	96 305.75	11 315.75	2490	21 315.5
PGP	1 553 037.75	801	126 445.5	**236.2**	401	1338.75	56 796.75	2448.2	352.5	3856.2
Sister cities	2238.75	**197.5**	11 158	337.9	809.1	1115.55	4249.35	15,26.2	523	956.2
As-22july06	355.2	**120**	345	223	267	277	333	287	297	321
CAIDA	4 536 828.1	**252.3**	4 233 699	370.3	3188.15	7387.8	198 503.55	8569.9	1956	9885.3
COND-MAT (1995–2005)	4 124 767.05	3406.9	1 574 145	**602.5**	6921.45	7335.9	340 261.2	6586.3	3105	10 489.89
RO	5 834 034	901.3	5 054 739	**576.4**	869.4	1173.6	11 609.1	3852	4777	6526.3
HU	13 997 135.7	1110.1	12 441 785	**737.1**	1065.15	1428.3	22 380.3	984.5	2985	3888.36
HR	12 052 800.9	3615.3	10 497 935	**1292**	4218.75	5947.2	39 387.6	4812.2	7745	8635.9
Douban	3 110 830.2	3850.2	4 272 160	**1463**	4214.25	4745.25	19 754.1	5785.2	2563	8555.8
M-Fo115	35	38	42	**33**	36	34	43	39	36	34
M-Fo120	43	41	44	**36**	38	39	46	41	39	37

will try to reduce the processing and creating of sub-graph in Algorithm 2. The main time-consuming part of LMP is Algorithm 2.

CRediT authorship contribution statement

Asgarali Bouyer: Conceptualization, Methodology, Software, Writing – review & editing, Supervision. **Hamid Ahmadi Beni:** Methodology, Programming in Python, and Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] K. Gong, M. Tang, P.M. Hui, H.F. Zhang, D. Younghae, Y.-C. Lai, An efficient immunization strategy for community networks, PLoS One 8 (12) (2013).
[2] N. Samadi, A. Bouyer, Identifying influential spreaders based on edge ratio and neighborhood diversity measures in complex networks, Computing 101 (8) (2019) 1147–1175.
[3] A. Bouyer, H. Roghani, LSMD: A fast and robust local community detection starting from low degree nodes in social networks, Future Gener. Comput. Syst. 113 (2020) 41–57.
[4] H. Roghani, A. Bouyer, E. Nourani, PLDLS: A novel parallel label diffusion and label selection-based community detection algorithm based on spark in social networks, Expert Syst. Appl. (2021) 115377.
[5] D. Kempe, J. Kleinberg, É. Tardos, 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, 2003, pp. 137–146.
[6] J. Leskovec, et al., Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 420–429.
[7] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 199–208.
[8] R. Narayanam, Y. Narahari, A shapley value-based approach to discover influential nodes in social networks, IEEE Trans. Autom. Sci. Eng. 8 (1) (2010) 130–147.

[9] 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.
[10] A. Goyal, W. Lu, L.V. Lakshmanan, Simpath: An efficient algorithm for influence maximization under the linear threshold model, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 211–220.
[11] L. Qiu, W. Jia, J. Yu, X. Fan, W. Gao, PHG: A three-phase algorithm for influence maximization based on community structure, IEEE Access 7 (2019) 62511–62522.
[12] S.S. Singh, A. Kumar, K. Singh, B. Biswas, C2IM: Community based context-aware influence maximization in social networks, Physica A 514 (2019) 796–818.
[13] A. Talukder, M.G.R. Alam, N.H. Tran, D. Niyato, C.S. Hong, Knapsack-based reverse influence maximization for target marketing in social networks, IEEE Access 7 (2019) 44182–44198.
[14] G. Xie, Y. Chen, H. Zhang, Y. Liu, MBIC: A novel influence propagation model for membership-based influence maximization in social networks, IEEE Access 7 (2019) 75696–75707.
[15] T. Cai, J. Li, A.S. Mian, T. Sellis, J.X. Yu, Target-aware holistic influence maximization in spatial social networks, IEEE Trans. Knowl. Data Eng. (2020).
[16] J. Tang, R. Zhang, P. Wang, Z. Zhao, L. Fan, X. Liu, A discrete shuffled frog-leaping algorithm to identify influential nodes for influence maximization in social networks, Knowl.-Based Syst. 187 (2020) 104833.
[17] S. Tang, J. Yuan, Influence maximization with partial feedback, Oper. Res. Lett. 48 (1) (2020) 24–28.
[18] Z. Aghaee, M.M. Ghasemi, H.A. Beni, A. Bouyer, A. Fatemi, A survey on meta-heuristic algorithms for the influence maximization problem in the social networks, Computing 103 (11) (2021) 2437–2477.
[19] H.A. Beni, S. Azimi, A. Bouyer, A node filtering approach for influence maximization problem in independent cascade model, in: 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), IEEE, 2020, pp. 1–4.
[20] X. Wu, L. Fu, S. Wang, B. Jiang, X. Wang, G. Chen, Collective influence maximization in mobile social networks, IEEE Trans. Mob. Comput. (2021).
[21] Z. Aghaee, H.A. Beni, S. Kianian, M. Vahidipour, A heuristic algorithm focusing on the rich-club phenomenon for the influence maximization problem in social networks, in: 2020 6th International Conference on Web Research (ICWR), IEEE, 2020, pp. 119–125.
[22] H.A. Beni, Z. Aghaee, A. Bouyer, M. Vahidipour, IMT: Selection of top-k nodes based on the topology structure in social networks, in: 2020 6th International Conference on Web Research (ICWR), IEEE, 2020, pp. 84–88.
[23] P. Domingos, M. Richardson, Mining the network value of customers, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 57–66.
[24] D. Kempe, J. Kleinberg, É. Tardos, Influential nodes in a diffusion model for social networks, in: International Colloquium on Automata, Languages, and Programming, Springer, 2005, pp. 1127–1138.
[25] W. Chen, T. Lin, Z. Tan, M. Zhao, X. Zhou, Robust influence maximization, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 795–804.
[26] F. Morone, B. Min, L. Bo, R. Mari, H.A. Makse, Collective influence algorithm to find influencers via optimal percolation in massively large social media, Sci. Rep. 6 (2016) 30062.
[27] J.-X. Zhang, D.-B. Chen, Q. Dong, Z.-D. Zhao, Identifying a set of influential spreaders in complex networks, Sci. Rep. 6 (2016) 27823.
[28] D. Liu, Y. Jing, J. Zhao, W. Wang, G. Song, A fast and efficient algorithm for mining top-k nodes in complex networks, Sci. Rep. 7 (2017) 43330.
[29] J. Shang, S. Zhou, X. Li, L. Liu, H. Wu, CoFIM: A community-based framework for influence maximization on large-scale networks, Knowl.-Based Syst. 117 (2017) 88–100.
[30] S. Ahajjam, H. Badir, Identification of influential spreaders in complex networks using HybridRank algorithm, Sci. Rep. 8 (1) (2018) 11932.
[31] S. Banerjee, M. Jenamani, D.K. Pratihar, ComBIM: A community-based solution approach for the budgeted influence maximization problem, Expert Syst. Appl. 125 (2019) 1–13.
[32] X. Rui, F. Meng, Z. Wang, G. Yuan, A reversed node ranking approach for influence maximization in social networks, Appl. Intell. 49 (7) (2019) 2684–2698.
[33] H.A. Beni, A. Bouyer, TI-SC: top-k influential nodes selection based on community detection and scoring criteria in social networks, J. Ambient Intell. Humaniz. Comput. (2020) 1–20.
[34] Z. Aghaee, S. Kianian, Efficient influence spread estimation for influence maximization, Soc. Netw. Anal. Min. 10 (1) (2020) 1–21.
[35] Z. Aghaee, S. Kianian, Influence maximization algorithm based on reducing search space in the social networks, SN Appl. Sci. 2 (12) (2020) 1–14.
[36] K. Berahmand, A. Bouyer, N. Samadi, A new local and multidimensional ranking measure to detect spreaders in social networks, Computing 101 (11) (2019) 1711–1733.
[37] C. Wang, W. Chen, Y. Wang, Scalable influence maximization for independent cascade model in large-scale social networks, Data Min. Knowl. Discov. 25 (3) (2012) 545–576.
[38] G. Wu, X. Gao, G. Yan, G. Chen, Parallel greedy algorithm to multiple influence maximization in social network, ACM Trans. Knowl. Discov. Data (TKDD) 15 (3) (2021) 1–21.
[39] S. Cheng, H. Shen, J. Huang, G. Zhang, X. Cheng, Staticgreedy: solving the scalability-accuracy dilemma in influence maximization, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, 2013, pp. 509–518.
[40] E. Cohen, D. Delling, T. Pajor, R.F. Werneck, Sketch-based influence maximization and computation: Scaling up with guarantees, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 629–638.
[41] H.T. Nguyen, M.T. Thai, T.N. Dinh, Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 695–710.
[42] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, S.-Y. Lee, CIM: Community-based influence maximization in social networks, ACM Trans. Intell. Syst. Technol. (TIST) 5 (2) (2014) 25.
[43] H. Huang, H. Shen, Z. Meng, Community-based influence maximization in attributed networks, Appl. Intell. (2019) 1–11.
[44] A.M. Samir, S. Rady, T.F. Gharib, LKG: A fast scalable community-based approach for influence maximization problem in social networks, Physica A (2021) 126258.
[45] N. Barbieri, F. Bonchi, G. Manco, Topic-aware social influence propagation models, Knowl. Inf. Syst. 37 (3) (2013) 555–584.
[46] W. Chen, T. Lin, C. Yang, Efficient topic-aware influence maximization using preprocessing, 2014, CoRR, abs/1403.0057.
[47] Q. Liqing, G. Chunmei, Z. Shuang, T. Xiangbo, Z. Mingjv, TSIM: A two-stage selection algorithm for influence maximization in social networks, IEEE Access 8 (2020) 12084–12095.
[48] J. Ding, W. Sun, J. Wu, Y. Guo, Influence maximization based on the realistic independent cascade model, Knowl.-Based Syst. 191 (2020) 105265.
[49] Y. Tang, X. Xiao, Y. Shi, Influence maximization: Near-optimal time complexity meets practical efficiency, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 75–86.
[50] Y. Tang, Y. Shi, X. Xiao, Influence maximization in near-linear time: A martingale approach, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1539–1554.

[51] B. Liu, G. Cong, D. Xu, Y. Zeng, Time constrained influence maximization in social networks, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 439–448.
[52] M. Kitsak, et al., Identification of influential spreaders in complex networks, Nat. Phys. 6 (11) (2010) 888.
[53] D.-L. Nguyen, T.-H. Nguyen, T.-H. Do, M. Yoo, Probability-based multi-hop diffusion method for influence maximization in social networks, Wirel. Pers. Commun. 93 (4) (2017) 903–916.
[54] Q. Yu, H. Li, Y. Liao, S. Cui, Fast budgeted influence maximization over multi-action event logs, IEEE Access 6 (2018) 14367–14378.
[55] H. Ahmadi Beni, A. Bouyer, Identifying influential nodes using a shell-based ranking and filtering method in social networks, Big Data (2021).
[56] W. Li, K. Zhong, J. Wang, D. Chen, A dynamic algorithm based on cohesive entropy for influence maximization in social networks, Expert Syst. Appl. 169 (2021) 114207.
[57] Z. Wang, C. Sun, J. Xi, X. Li, Influence maximization in social graphs based on community structure and node coverage gain, Future Gener. Comput. Syst. 118 (2021) 327–338.
[58] L. Ma, Y. Liu, Maximizing three-hop influence spread in social networks using discrete comprehensive learning artificial bee colony optimizer, Appl. Soft Comput. 83 (2019) 105606.
[59] A. Zareie, A. Sheikhahmadi, M. Jalili, Identification of influential users in social network using gray wolf optimization algorithm, Expert Syst. Appl. 142 (2020) 112971.
[60] S. Wang, J. Liu, Y. Jin, Finding influential nodes in multiplex networks using a memetic algorithm, IEEE Trans. Cybern. (2019).
[61] S. Wang, J. Liu, Community robustness and its enhancement in interdependent networks, Appl. Soft Comput. 77 (2019) 665–677.
[62] A. Bozorgi, H. Haghighi, M. Sadegh 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.
[63] P.G. Lind, M.C. Gonzalez, H.J. Herrmann, Cycles and clustering in bipartite networks, Phys. Rev. E 72 (5) (2005) 056127.
[64] J. Kunegis, Konect: the koblenz network collection, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 1343–1350.
[65] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (5) (2004) 056122.
[66] B. Rozemberczki, R. Davies, R. Sarkar, C. Sutton, Gemsec: Graph embedding with self clustering, in: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2019, pp. 65–72.
[67] R. Zafarani, H. Liu, Social computing data repository at ASU, 2009, 2009, URL http://socialcomputing.asu.edu.
[68] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.