Research Article

Influence maximization algorithm based on reducing search space in the social networks

Zahra Aghaee[1] · Sahar Kianian[1]

Received: 24 April 2020 / Accepted: 31 October 2020 / Published online: 24 November 2020

Abstract
The influence maximization problem in social networks is an optimization problem in viral marketing. This problem is concerned with identifying a certain number of people with the most influence in the social network level. Considering the NP-hard of this problem, finding an optimal solution with acceptable accuracy and the low running time is of the high importance. To this purpose, GIN (Group of Influential Nodes) algorithm is presented in this article which creates different groups of graph nodes with more connections than other groups. Then, it selects specific nodes from each group to reduce the search space to find the most influential nodes. Following the greedy method, it selects the seed nodes with the highest expected diffusion value. Experimental results show that the GIN algorithm has provided high influence spread along with low running time in comparison algorithms on all seven real-world datasets.

Keywords Social network · Influence maximization problem · Viral marketing · Information diffusion

1 Introduction

The social networks like LinkedIn, Facebook, and Twitter provide the grounds for people to interact with no time and space limitation [1]. Consequently, information diffusion occurs across a wide range of social networks at high speed. That's why business companies are using the speed of information diffusion on social networks to achieve widespread advertising and business optimization. Given the limited advertising resources, the main challenge in this problem is to select a specific set of influential persons that most effectively on other people in a short time. To solve this challenge, the influence maximization problem in social networks has been provided.

The influence maximization problem identifies the active nodes as seed nodes. Algorithms in this problem take the graph G and a number k as input and generate seed set, with the intention that the expected number of nodes influenced by the seed set by the stochastic process of the diffusion model, One major problem in the influence maximization problem to maximize the expected size of the final active set, given some constraints on the seed set. The diffusion process of the seed nodes is performed to maximize the influence spread. The influence of seed nodes is determined by the number of nodes activated. The influence maximization problem under both IC and LT models are NP-hard. In other words, the goal of the influence maximization problem is to find initial active people who have the most influence on other people in a short time under a diffusion model [2, 3].

Viral marketing [1, 4–9], terrorist attack prevention [4], and network control [10, 11] constitute the applications of the influence maximization problem that viral marketing has attracted researchers more than other applications. In viral marketing, unlike traditional marketing where information diffusion was through newspaper advertising and the mass media, the focus is on peoples effective through family members, relatives, acquaintances, and friends by

✉ Zahra Aghaee, z.aghayii@sru.ac.ir; Sahar Kianian, Sahar.kianian@sru.ac.ir | [1]Faculty of Computer Engineering, Shahid Rajaee Teacher Training University (SRTTU), Tehran, Iran.

word of mouth [12]. In this way, diffusion is information transmission from person to other person, such as virus transmission [1]. Domingos and Richardson were the first who addressed the influence maximization problem [6]. Kempe et al. formulated and proved that this problem is an NP-hard [2], who then proposed an approximation algorithm called General Greedy algorithm to solve the influence maximization problem [2]. This algorithm has very low speed and high running time [2]. Some researchers proposed some algorithms to overcome drawbacks in the General Greedy algorithm. The existing influence maximization algorithms have several major drawbacks:

1. They have no effective methods to reduce the search space for influence spread calculation and do not optimally select the candidate nodes.

2. They are not suitable for large-scale networks due to high computational time.

Handling the NP-Hardness and to optimize the seed selection by reducing search space while keeping the running time faster are important to influence maximization problem. So, to address these issues, we propose a new algorithm named GIN heuristic algorithm. In this algorithm, by the identified nodes with a more common number of connections and neighbors than other nodes of the graph, different groups of nodes are generated. By selecting certain nodes from each group, the search space where the final effective nodes are to be found is reduced. Its goal is to increase the speed of the algorithm. During this process, the nodes with the highest expected diffusion value are selected as the seed nodes.

The contributions of our work are as follows:

- Use a grouping method for graph nodes.
- We use candidate nodes optimally to reduce the search space for influence spread calculation.
- The experimental results on seven real-world networks show that the proposed algorithm GIN performs better than the other algorithms in terms of influence spread and running time.

The structure of this article is organized as follows. In Sect. 2, the related works for influence maximization problem is introduced. In Sect. 3, problem definition is expressed. Our proposed algorithm is presented in Sect. 4. Experiments results and evaluations are described in Sect. 5 and we conclude this article in Sect. 6.

2 Related works

Identifying seed nodes is one of the key issues for influence maximization problem in social networks. This is one of the NP-hard problems [2] that has many challenges like high influence spread and low running time, especially on large-scale graphs. Therefore, researchers have presented different algorithms to solve the influence maximization problem. For the first time, Kempe et al. presented an approximate algorithm called General Greedy algorithm with high influence spread that has an optimal approximation [2]. However, this algorithm has a high running time, especially on large-scale graphs [13]. So they proposed the High Degree algorithm by selecting nodes with the highest degree in graph [2]. The High Degree algorithm is very fast, even on large-scale graphs, but has a lower influence spread than the General Greedy algorithm.

Leskovec et al. proposed the CELF algorithm [14]. This algorithm uses lazy evaluation in functions that have the submodularity, significantly reduces influence spread evaluation. Although the CELF algorithm is faster than the General Greedy algorithm, it does not have acceptable speed. Then, Chen et al. proposed the NewGreedyIC algorithm to improve the CELF algorithm [15]. This algorithm computes the influence spread by creating sample graphs from the original graph. However, the NewGreedyIC algorithm improves the running time of the CELF algorithm, but this algorithm still needs to improve the running time, especially on large-scale graphs.

Chen et al. presented the DegreeDiscount algorithm to improve the running time of the CELF algorithm [15]. This algorithm uses a degree discount method for influence spread computations. The DegreeDiscount algorithm has a better running time than the CELF algorithm but does not guarantee optimal approximation. Also, Jiang et al. presented an evolutionary algorithm SA to improve the DegreeDiscount algorithm [16]. This algorithm has gained more influence spread than the DegreeDiscount algorithm, with a high number of iterations and increased running time. Then presented the SAEDV algorithm to reduce the running time of the SA algorithm [16]. In this algorithm, the seed nodes are selected with the maximum expected diffusion value based on the SA algorithm. The SAEDV algorithm has a lower running time than the SA algorithm.

Goyal et al. introduced the SIMPATH algorithm to improve the running time of the CELF algorithm [17]. This algorithm computes the influence spread without using Monte Carlo simulation and using simple paths count. The SIMPATH algorithm has a low memory overhead but does not guarantee optimal approximation. Another algorithm proposed to solve the influence maximization problem is the MIA algorithm [18]. This algorithm was proposed by Wang et al. to improve the computation of the influence spread of the DegreeDiscount algorithm. In the MIA algorithm, using the Dijkstra algorithm, the shortest maximum influence path for each node of the graph is calculated. Although the MIA algorithm has an acceptable running time by eliminating the number of Monte Carlo simulation,

the disadvantage of this algorithm is the high memory overhead.

Jung et al. proposed the IRIE algorithm to improve the MIA algorithm [19]. In this algorithm, the influence spread is calculated recursively based on the influence spread of the neighbors of each node in the graph. In the IRIE algorithm, the memory overhead is very low, but the influence spread is low compared to the General Greedy algorithm. In the following, He et al. presented the CLDAG algorithm to improve the General Greedy algorithm [20]. The focus of this algorithm is on blocking the influence maximization problem. Thus, the most important advantage of this algorithm is not dependent on the number of seed nodes, and its main disadvantage is the high memory overhead.

Tang et al. proposed the TIM algorithm to improve the SIMPATH algorithm [21]. In this algorithm, the first set of RR-Set includes nodes that have paths to each other, and then the seed nodes are selected using graph nodes covering. Benefits of TIM algorithm are low running time and low memory overhead. Then, Lu et al. developed the RR-CIM algorithm to improve the High Degree algorithm [22]. This algorithm solves the influence maximization problem by using complementary products. The parameters of this algorithm close the influence maximization problem to the real-world, but the algorithm has a high running time.

In the following, He et al. presented a new algorithm based on community detection to improve the High Degree algorithm [23]. In this algorithm, each community is considered as one node. Then influential nodes are selected based on the red-black tree. This algorithm avoids the rich-club phenomenon but has a high running time. Also, Morone et al. Presented the CI algorithm based on localization of influence spread computation [24]. In this algorithm, the influence spread computation in a circle is limited to the radius L. The main advantage of the CI algorithm is to provide acceptable influence spread into high-density social networks, however, that the calculation of influence spread and running time of the CI algorithm depends on the parameter L and the number of seed nodes.

Then the topology-based algorithm LIR is presented by Liu et al. to improve DegreeDiscount and NewGreedyIC algorithms [25]. First, in this algorithm is calculated the LI of each node, which is indicated the local index of each node that avoids the Rich-club Effect but does not guarantee optimal approximation. The CoFIM algorithm is also based on community detection proposed by Shang et al. to improve High Degree algorithm [26]. In this algorithm, first, the influence spread is calculated based on diffusion between communities and then diffusion inside communities. The advantage of this algorithm is low running time, but the overlapping nodes are not considered. Then Wu et al. developed the LAIM algorithm to improve CoFIM algorithm [27]. By this algorithm, the influence is estimated locally. The main advantage of the LAIM algorithm is its very low running time, and its main disadvantage is that it does not guarantee optimal approximation.

Cui et al. presented an evolutionary algorithm called DDSE to improve the running time of the CELF algorithm [28]. In this algorithm, using the genetic algorithm and local and global search, the seed nodes are selected with the highest expected diffusion value. The DDSE algorithm has influence spread very close to the CELF algorithm on some small datasets. Xie et al. developed the IRR algorithm based on the MBIC model [29]. This algorithm uses the same idea as the DegreeDiscount algorithm. The main advantage of the IRR algorithm is the use of different states based on real-world criteria for nodes, but this algorithm is dependent on the number of nodes in the graph.

The C2IM algorithm is presented by Singh et al. to improve the time-efficiency of the TIM algorithm for the influence maximization problem [10]. This algorithm uses the user's interests and community framework to find effective seed nodes. Also, Ahmadi Beni et al. presented an algorithm called TI-SC based on community detection to improve the influence spread of the DegreeDiscount algorithm [30]. This algorithm selects seed nodes by examining the relationships between the core nodes. The TI-SC algorithm is useful to reduce the overlap in selecting the seed nodes, but this algorithm has not an efficient performance in large-scale networks.

3 Problem definition

The social networks' graph is mapped as $G(V, E)$, where V and E are described as follows:

- $V = \{v_1, v_2, \ldots, v_n\}$ is a set of graph nodes, where v_i is the individuals, and n is the number of nodes in the graph.
- $E = \{e_{1,2}, e_{1,3}, \ldots, e_{1,n}, e_{2,1}, e_{2,3}, \ldots, e_{n,n-1}\}$ is a set of graph edges, where, $e_{i,j}$ is the relationship between two individuals or an edge between each pair of nodes ($i \neq j$).

Considering $G(V, E)$ to find k seed nodes for influence maximization problem, the subset S select from $V = \{v_1, v_2, \ldots, v_n\}$ due to the diffusion model. The process of seed nodes selection is done until $|S| = k$ so that the influence spread of S, $\sigma(S)$ is maximized by Eq. (1) under the given diffusion model [31]:

$$S^* = argmax_{S \subset V, |S|=k} \sigma(S) \tag{1}$$

The influence spread is calculated due to the diffusion model for influence maximization problem. One of the most widely used information diffusion models is the

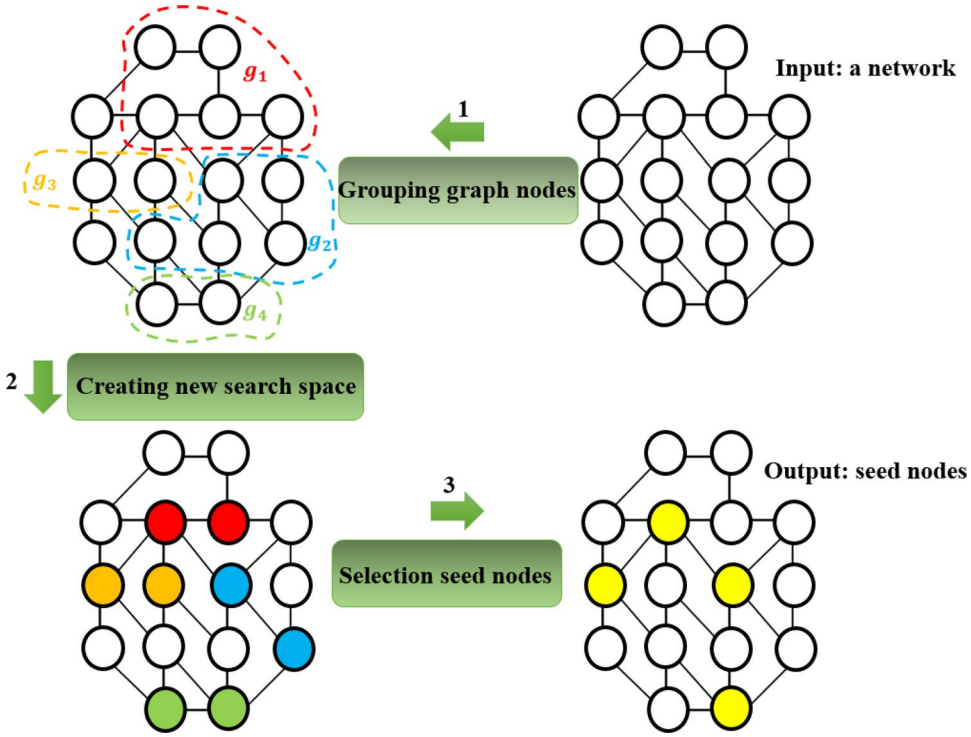


Fig. 1 General levels of the GIN algorithm

independent cascade model proposed by Kempe et al. The independent cascade model has two important properties in the influence spread function which these properties are submodularity and monotonicity. A set function $f: 2^V \rightarrow R$ is submodular if, for any $S \subseteq T \subseteq V$ subsets and any$v \in V \backslash T$ element , the marginal contribution of v element, when added to a T set can not exceed the marginal contribution of v element when added to a $S \subseteq T$ subset due to Eq. (2) [31]:

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T) \tag{2}$$

The objective function $f: 2^V \rightarrow R$ is monotone if we have $f(S) \leq f(T)$ for each$S \subseteq T \subseteq V$ subset [31].

4 Proposed algorithm

In GIN algorithm, a heuristic method is proposed to identify the influential nodes for the influence maximization problem. In this algorithm, different graph nodes are created to identify nodes with more communication and more common neighbors than other graph nodes. Then, certain nodes are selected from each group. This process is done to create new search space and reduce the number of influence spread calculations. At last, the final influential nodes are selected among the nodes in the new search space which maximize the expected diffusion value as the seed nodes. Figure 1 shows an overview of the general levels of GIN algorithm.

According to Fig. 1, the GIN algorithm consists of three levels: (1) grouping graph nodes, (2) creating new search space, and (3) selecting seed nodes.

4.1 Grouping graph nodes

The first level of algorithm GIN is due to AntCBO [32] algorithm by modification. This level of GIN algorithm includes the process of creating different groups of graph nodes with more communication and common neighbors than other nodes. This level consists of six basic steps:

Step 1 For every node of graph, an *available* variable is defined which determines the available or non- available of each node. The value of this variable is either *True* or *False*. At first, the value of this variable is considered *True* for all graph nodes. If the node has an *available* = *True* value, it will be available; so, it can be selected for calculations, otherwise non-available; thus, it cannot be selected for calculations. Figure 2 shows a graph consists of 16 nodes with *available* = *True* variable for all graph nodes.

Step 2 The graph nodes are sorted from highest to lowest degree in descending order in an undirected graph, or out-degree in a directed graph and placed in list L. Thus, in the first place of list, the node is with the highest degree, and in the last place, the node is with the lowest degree from graph, according to Fig. 3.

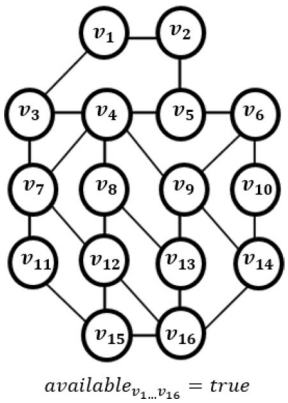


Fig. 2 Example of the graph containing 16 nodes with the *available* = *True* variable

Step 3 The group $g_i = \{\}$ is created ($i = \{1, 2, 3, \ldots, I\}$). Then, the first node from list *L* which has an *available* = *True* value, is selected and added to g_i group ($g_i \cup \{v_i\}$). Then, the availability value of this node changes into *available* = *False*. In the following, v_j node with the highest degree among v_i node 's neighbors which has the *available* = *True* value, is selected and added to g_i group ($g_i \cup \{v_j\}$). The availability value of v_j *node* is changed into *available* = *False* (Fig. 4).

Step 4 If v_i and v_j nodes do not have common neighbors, the computation continues from the previous step for the next node from *L* list with the value of *avalable* = *True*.

Step 5 If v_i and v_j nodes have common neighbors, *ComNei* set will be created including common neighbors of v_i and v_j nodes ($ComNei = \{v_1, v_2, \ldots, v_m\}$) which $\{v_1, v_2, \ldots, v_m\}$ will be always belonging to *ComNei* for each v_i and v_j. The v_m node is selected from *ComNei* set with *available* = *True* value randomly and is added to g_i group ($g_i \cup \{v_m\}$). Then, the availability value of this node is changed to *available* = *False*. If the neighbors of v_m node do not have any common with nodes within *ComNei* set , the availability of next node from *ComNei* set will be checked which is added to g_i group ; otherwise, *Z* set including the common nodes of v_m's neighbors and *ComNei* set is created ($Z = \text{neighbor}(v_m) \cap ComNei$). Then, *ComNei* set is replaced with *Z* set . This process is repeated until *ComNei* = {}, or the nodes in this set have *available* = *false* value (Fig. 5).

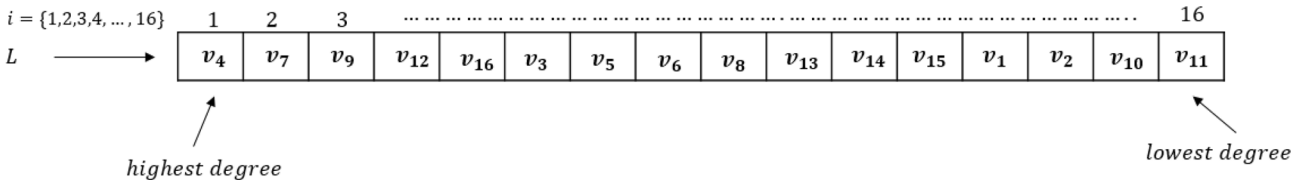


Fig. 3 Example of placing nodes in list *L*

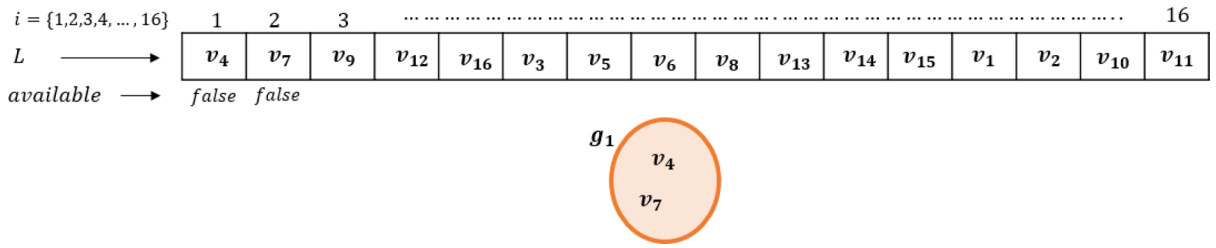


Fig. 4 An example of placing graph nodes in the first group

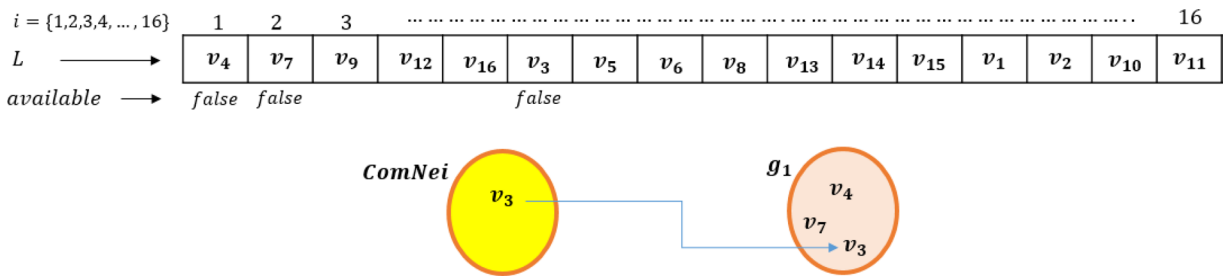


Fig. 5 An example of adding other nodes to the first group

Step 6 The previous described steps are repeated until the created groups reach a specified number ($Itrg_i = I$). I is the maximum number of groups created; also, that is constant.

4.2 Creating new search space

In this level, the two nodes v_i and v_j which include the first and second added nodes are selected from each g_i group. The first node selected from each g_i group is the node v_i with the highest degree in that group, and the second node selected is the v_j node with the highest degree among the neighbors v_i. Each group g_i contains at least one v_i node. If there exists no node v_j in a g_i, that group has only one node, thus the only v_i node is selected. Also, if there are multiple nodes v_i that have the same maximum degree value in each g_i group, and similarly, if there are such multiple nodes v_j, these are selected randomly. In this way, by identifying and selecting these nodes from each group, the search space to find influential nodes is reduced. This level is repeated until the selected nodes reach a specified number ($Ncandidate = m$). m is the maximum number of candidate nodes selected.

4.3 Selection of seed nodes

In this level, the selection of seed nodes from a new search space is done by the greedy method. Thus, according to Eq. (3), the Expected Diffusion Value (EDV) of these nodes is calculated. This equation has also been used in other studies [16, 28]. The first time, Jiang et al. proposed the expected diffusion value to estimate the actual influence spread under the independent Cascade model and employed it in their algorithm to replace Monte Carlo simulation [16]. The EDV approximates the spread by computing how many direct neighbors of the nodes in the candidate node set S are expected to be activated. According to the Eq. (3), k is the number of seed node, p is the activation probability the non-seed nodes, $N_S^I \setminus S$ is the neighbors of the nodes S except the nodes of set S, and $\tau(i)$ is the number of neighbors of i node in set S.

$$EDV(S) = k + \sum_{i \epsilon N_S^I \setminus S} 1 - (1-p)^{\tau(i)} \tag{3}$$

Consequently, by applying the greedy selection method, a node from the new search space which causes maximization of the marginal gain function ($EDV(S \cup \{w\}) - EDV(S)$) as the seed node added to the set S. This process is repeated until $|S| = k$. Then, the influence spread of seed nodes is calculated using the independent cascade model. Algorithm1 shows the pseudo-code of the GIN algorithm.

Table 1 Specifications of seven real-world social networks

Dataset	Statistic characteristics		
	# Node	# Edge	Type
Email	1133	5451	Undirected
NetScience	1461	2742	Undirected
Power	4941	6594	Undirected
NetHEPT	15,233	58,891	Undirected
NetPHY	37,149	231,507	Undirected
Epinions1	75,879	508,837	Directed
Slashdot0902	82,168	948,464	Directed

Algorithm1 GIN Algorithm

Input: Graph $G = (V, E)$, the size of seed set k
Sort all vertices by their degree in descending order L
$\forall\, v \in V: v.availabe = true$, CN = {}
for each $v_i \in L$ **do**
  $g = \{\}$ //g is a group
  if $v_i.availale$ **then**
    find v_j which has the largest degree in the neighbor of v_i *and* $v_j.availabe$ is $True$
    $v_j.availabe = false$
    put v_i *and* v_j *into* g
    compute common neighbors $comNei$ between v_i *and* v_j //$(ComNei = \{v_1, v_2, \ldots, v_m\}$
    while $ComNei$ is not empty **do**
      for each vertex v_m in $ComNei$
        $g = g \cup v_m$
        $v_m.availabe = false$
        $Z = neighbor(v_m) \cap ComNei$ and replace $ComNei$ with Z
      end for
    end while
  end if
  $CN(i).g = g$
end for
$A = \{\}$
for $i = 1$ to l **do** //l is Maximum number of groups created
  if $|A| == m$ **then** //m is the maximum number of candidate nodes selected
    break
  end if
  $A = A \cup CN(i).g(1)$
  $A = A \cup CN(i).g(2)$
end for
$S = \{\}$
for $i = 1$ to k **do**
  Select $w \longleftarrow argmax_{w \in A \setminus S}(EDV(S \cup \{w\}) - EDV(S))$
  $S = S \cup \{w\}$
end for
output $\sigma(S)$

5 Experiments and evaluation of the proposed algorithm

The GIN algorithm has been evaluated on seven real datasets. Then the results of influence spread and running time of algorithms evaluated under the independent cascade model in a similar system. Influence spread means the number of final nodes activated by the seed nodes in the diffusion process. The running time means the time has passed to find the seed nodes. All the codes are written in C#, and all algorithms are performed on a Windows 10 operating system with 2.20 GHz CPU (Intel® Core (TN) i5-5200U) and 8 GB of RAM.

5.1 Description of the datasets

The GIN algorithm is implemented on seven datasets of different sizes. The characteristics of these datasets are shown in Table 1.

Email [33]**:** This dataset is a network of emails at URV University that includes professors, administrators, technicians, researchers, and graduates. In this dataset, each node represents the email address, and the link between the emails is an edge.

NetScience [34]**:** This dataset contains a collaboration network of scientists working on network theory. Newman compiles this dataset in the year 2006. The nodes in this dataset represent the authors of the articles, and the edges indicate the collaboration between the authors in a common article.

Table 2 The influence spread of GIN algorithm with different numbers of *l* on Email dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	8.2679	8.2595	**8.295**	8.2717
10	15.5823	**15.6284**	15.5704	15.6157
15	22.7254	**22.7354**	22.7309	22.7035
20	29.5863	**29.6679**	29.5751	29.5901
25	**36.3615**	36.2817	36.2673	36.2868
30	42.7783	**42.8733**	42.7996	42.7962
35	49.2754	**49.3008**	49.2544	49.2541
40	55.4953	55.509	55.4815	**55.5318**
45	61.7827	**61.8008**	61.6706	61.6959
50	67.7524	**67.753**	67.7057	67.7358

Table 3 The influence spread of GIN algorithm with different numbers of *l* on NetScience dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=180$	$l=200$
5	6.3036	**6.3608**	6.35	6.3358
10	12.303	**12.3133**	12.3096	12.2741
15	18.0436	**18.0881**	18.0548	18.0213
20	23.6001	**23.6807**	23.6243	23.6576
25	29.1569	29.1871	**29.1906**	29.1561
30	34.6915	**34.7668**	34.6975	34.6549
35	40.1395	**40.1623**	40.1432	40.1577
40	45.5781	**45.6042**	45.5842	45.5854
45	51.02	**51.0576**	51.0198	50.9824
50	56.304	**56.6203**	56.2735	56.3542

Power [35]**:** This dataset contains information about the power grid of the Western States of the United States of America that a node is either a generator, a transformator, or a substation. Also, an edge represents a power supply line.

NetHEPT [15]**:** This dataset contains arXiv articles in the "High Energy Physics Theory" section, published in the years 1991 to 2003. The nodes in this dataset represent the authors and, if they work together in an article, an edge is created between them.

NetPHY [15]**:** This dataset contains arXiv articles in the "Physics" section. In this network, each node is an author, and the edges between two nodes represent the collaboration of two authors in one article.

Epinions1[1]**:** This dataset is an online social network trust-based, collected from www.epinions.com. Each node in this dataset represents customers who are users of the site, and each edge represents a secure connection.

Slashdot0902[2]**:** This dataset is a version of the popular site www.slashdot.org about technology news where users can tag other users as friends or foes. The network was collected in February 2009.

5.2 The compared algorithms

The GIN algorithm is compared with various algorithms, including General Greedy, High Degree, SAEDV, LIR, and DDSE algorithms based on two criteria of influence spread and running time. The algorithm is implemented with parameter $l=150$, Monte Carlo simulation $R=10000$, and the activation probability $p=0.01$.

Table 4 The influence spread of GIN algorithm with different numbers of *l* on Power dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	5.8198	**5.8205**	5.8055	5.8106
10	11.4745	11.4658	11.4727	**11.4791**
15	17.0807	17.0843	**17.0934**	17.0735
20	22.6347	**22.6825**	22.6381	22.6704
25	28.1598	28.1442	**28.1626**	28.1487
30	33.6792	**33.6978**	33.6612	33.6942
35	39.1785	**39.195**	39.1695	39.1915
40	44.6525	**44.6766**	44.6573	44.6417
45	50.1167	**50.1749**	50.1231	50.1651
50	55.5309	**55.5962**	55.5603	55.5219

Table 5 The influence spread of GIN algorithm with different numbers of *l* on NetHEPT dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	**26.1731**	26.0955	26.1535	26.0446
10	41.2602	41.3175	**41.3776**	41.3197
15	**54.9883**	54.8802	54.9348	54.8401
20	67.8898	**67.8937**	67.6899	67.7233
25	79.8648	**79.9642**	79.9536	80.0444
30	**90.6858**	90.6806	90.6263	90.6541
35	102.075	**102.3536**	102.2648	102.1021
40	112.525	**112.7438**	112.5794	112.6274
45	122.8239	**123.1066**	122.7608	122.9495
50	133.0002	**133.369**	133.2169	133.0452

[1] https://snap.stanford.edu/data/soc-Epinions1.html

[2] https://snap.stanford.edu/data/soc-Slashdot0902.html

Table 6 The influence spread of GIN algorithm with different numbers of *l* on NetPHY dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	79.2958	79.6139	**80.169**	80.0705
10	133.6122	**133.9336**	133.8765	133.8068
15	**163.5793**	163.4865	162.8304	162.949
20	189.8483	**191.1813**	190.3365	190.7536
25	**213.0976**	212.5837	213.0566	212.7283
30	235.4313	**236.9035**	236.1157	235.8644
35	**257.4762**	256.9975	257.1422	257.4434
40	274.1901	**274.2439**	274.079	273.9996
45	297.3691	**297.4009**	296.6467	296.7144
50	312.5917	**312.6715**	311.8995	311.0109

Table 7 The influence spread of GIN algorithm with different numbers of *l* on Epinions1 dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	297.5951	296.5954	**298.7346**	297.5165
10	386.8471	386.6091	386.9116	**386.9241**
15	443.5619	**443.6522**	443.1479	443.5492
20	495.6272	**496.7142**	496.3577	496.4521
25	**531.0407**	530.365	529.8079	530.0372
30	**561.9917**	561.391	561.741	562.222
35	596.5624	**596.6392**	596.4791	595.4423
40	617.033	**617.4826**	615.7071	616.8711
45	**644.4913**	643.5229	643.6808	643.9246
50	668.1688	**668.1874**	668.4707	666.7429

Table 8 The influence spread of GIN algorithm with different numbers of *l* on Slashdot0902 dataset

Seed nodes	Number of groups			
	$l=100$	$l=150$	$l=200$	$l=250$
5	797.9882	798.2574	798.0206	**800.7689**
10	904.4977	902.3836	900.3276	**904.5713**
15	967.3272	967.9448	967.1864	**969.4299**
20	1031.35	**1033.427**	1030.27	1032.101
25	1078.159	**1078.335**	1078.108	1075.829
30	1115.339	**1115.513**	1114.212	1114.561
35	1153.326	**1155.722**	1155.516	1155.113
40	1190.819	1191.583	**1194.431**	1194.35
45	1226.905	**1228.293**	1228.11	1228.263
50	1255.015	**1255.91**	1255.753	1255.092

General Greedy According to 2], this algorithm is evaluated by Monte Carlo simulation $R=10{,}000$ and the activation probability $p=0.01$ under independent cascade model.

High Degree According to [2], in this algorithm, the influence spread of k seed nodes with the highest degree of the graph is calculated under the independent cascade model.

SAEDV According to [16], in this algorithm, choose to initialize a node set randomly and optimize it with iterations. The parameters for SAEDV algorithm are set as follows: the initial temperature $T_0=500000$, the final temperature $T_f=100000$, the number of inner loop $q=10$ and the temperature drop after each inner loop $\Delta T=2000$.

LIR According to [25], in this algorithm, nodes with 0-LI are selected for the NetScience, NetHEPT, NetPHY, Epinions1, and Slashdot0902 datasets, while in the Email dataset, nodes with 0-LI are low, then nodes are selected with 1-LI. Influence spread calculations are performed by Monte Carlo simulation $R=10{,}000$ and the activation probability $p=0.01$ under independent cascade model.

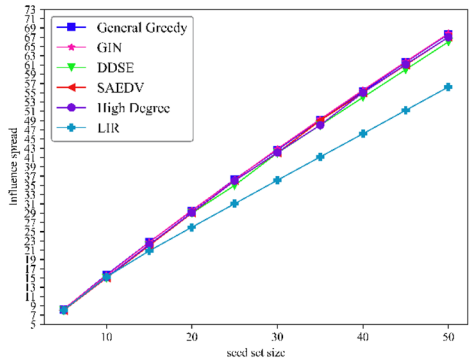
DDSE According to [28], the results of this algorithm are obtained with the size of the population $Npop=10$, mutation probability $OP_{Mutation}=0.1$, Crossover probability $OP_{Crossover}=0.4$, Number of generation the evolution lasts $MaxIt=200$ and the diversity of the population $Diversity=0.6$.

5.3 Diffusion model

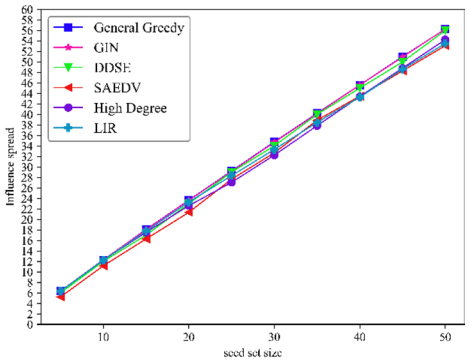
Researchers have proposed various types of information diffusion models [2, 36–40]. One of the most widely used diffusion models is the independent cascade model, which has been used in most studies [2, 5, 37, 41, 42]. Our algorithm focuses on the influence maximization problem under the independent cascade model too. This model was introduced by Goldenberg et al. [5]. In this model, the nodes are either in an active or inactive state. Each edge between two nodes *v* and *u* has the activation probability P_{vu} that means inactive node *u* is active by node *v*. Each active node *v* at time *t* can only activate its neighbor's inactive node once. And then, nodes that are activated at time $t-1$ can activate their neighbor inactive nodes. Thus, if every active node *v* at time *t* with probability P_{vu} activates its neighbor inactive nodes, then at time $t+1$ considered as an active node. The diffusion process will stop when no more new nodes are activated.

5.4 Experimental results

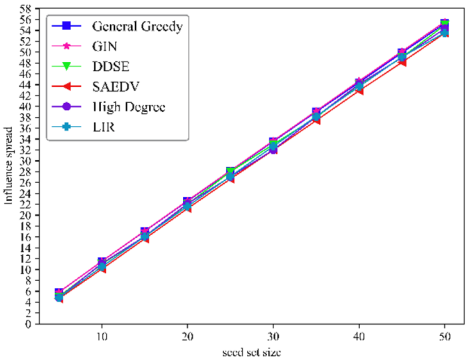
According to the description of the proposed algorithm, first, *l* groups are created. Here, the GIN algorithm is tested


General Greedy
GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

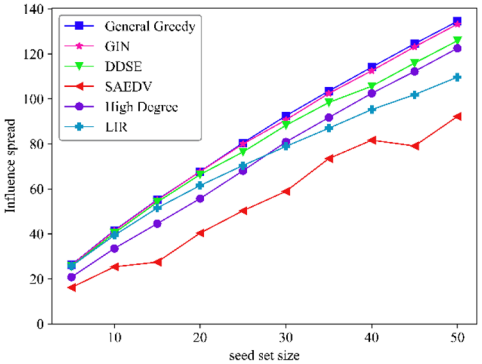
(a)


General Greedy
GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

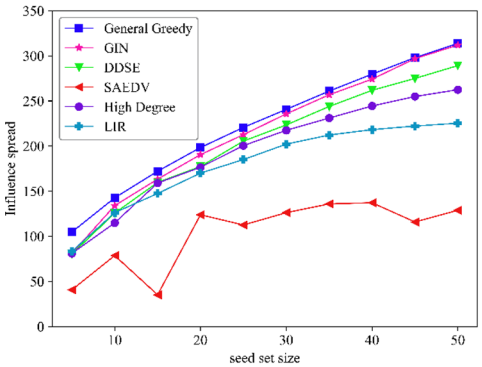
(b)


General Greedy
GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

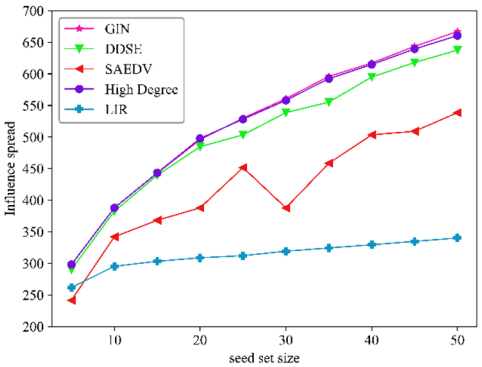
(c)


General Greedy
GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

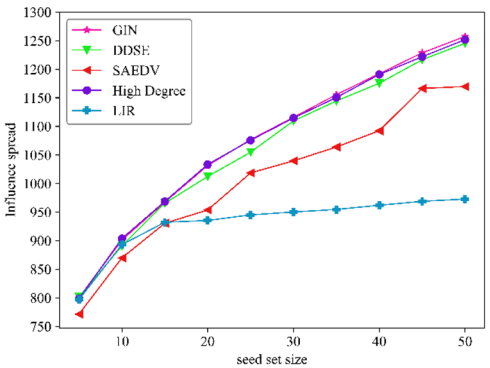
(d)


General Greedy
GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

(e)


GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

(f)


GIN
DDSE
SAEDV
High Degree
LIR
Influence spread
seed set size

(g)

◂**Fig. 6** Diagram of the influence spread on (**a**) Email dataset, (**b**) NetScience dataset, (**c**) Power dataset, (**d**) NetHEPT dataset, (**e**) NetPHY dataset, (**f**) Epinions1 dataset and (**g**) Slashdot0902 dataset

with different *l* to selecting the best value of parameter *l* with the highest influence spread on each seven real-world datasets. The influence spread results of this algorithm with different *l* are shown in Tables 2, 3, 4, 5, 6, 7, and 8 on Email, NetScience, Power, NetHEPT, NetPHY, Epinions1, and Slashdot0902 datasets respectively. According to these results, obviously, the influence spread of the different *l* is very close to each other algorithms, but the influence spread of the GIN with $l = 150$ is higher than other values of the parameter *l* in most of the *k* nodes. Thus, to compare the influence spread of the proposed algorithm with other algorithms is used parameter *l*=150 for the GIN algorithm on all of seven datasets.

The results of the evaluated algorithms in terms of influence spread and running time are shown in Fig. 6. These algorithms were implemented with fixed activation probability ($p = 0.01$) and varying sizes of the seed set with the range from $k = 5$ to $k = 50$ based on the independent cascade model. The x-axis indicates the seed nodes, and the y-axis indicates the influence spread.

The influence spread of the evaluated algorithms on the Email dataset is shown in Fig. 6a. In this figure, the results of the algorithms General Greedy and GIN are close to each other. The influence spread in the SAEDV algorithm is higher than the High Degree and DDSE algorithms. The influence spread in the DDSE algorithm from $k = 5$ to $k = 20$ is very close to that of the High Degree algorithm, and from $k = 25$ to $k = 50$, it follows a gradual reduction. The LIR algorithm has the lowest influence spread than the other algorithms in the Email dataset.

In Fig. 6b, the influence spread of the evaluated algorithms on the NetScience dataset is shown. In this figure, the influence spread of General Greedy, GIN, and DDSE algorithms are higher than other algorithms and are close to each other. The influence spread in LIR and SAEDV algorithms is less than the High Degree algorithm. Also, the influence spread of the comparable algorithms on the Power dataset is shown in Fig. 6c. According to Fig. 6c, General Greedy and GIN have higher influence spread than other algorithms. The highest influence spread belongs to DDSE after than the General Greedy and GIN algorithms. Then, the influence spread from high to low belongs to the High Degree, LIR, and SAEDV in most *k* nodes, respectively.

The influence spread of the evaluated algorithms on the NetHEPT dataset is shown in Fig. 6d, where the highest influence spread is for General Greedy and then GIN algorithm. The influence spread in the GIN algorithm, for $k = 5$ to $k = 25$, is very close to the General Greedy algorithm. The DDSE algorithm has a higher influence spread than the High Degree, LIR, and SAEDV algorithms. Of course, the results of the DDSE algorithm from $k = 5$ to $k = 20$ are close to the GIN algorithm. After the DDSE algorithm, the highest influence spread is for the High Degree algorithm and then the LIR algorithm. The influence spread of the SAEDV algorithm is very low on the NetHEPT dataset.

The influence spread of the evaluated algorithms on the NetPHY dataset is shown in Fig. 6e. In this figure, the influence spread of General Greedy and then GIN are higher than other algorithms. Also, the influence spread from most to less is for the DDSE algorithm and then the High Degree, LIR, and SAEDV algorithms. Also, the influence spread of the evaluated algorithms on the Epinions1 dataset is shown in Fig. 6f. Epinions1 is a large-scale dataset, and the General Greedy algorithm cannot run on it because this algorithm has a high running time. Therefore, the run of the General Greedy algorithm on the Epinions1 dataset is ignored. As observed in Fig. 6f, the GIN and High Degree algorithms have higher influence spread than other algorithms. However, GIN and High Degree algorithms have very close results in some *k* nodes, but the GIN algorithm in most *k* nodes has a higher influence spread than the High Degree. Then the influence spread of the DDSE algorithm is more than SAEDV and LIR algorithms.

In Fig. 6g, the influence spread of the algorithms on the Slashdot0902 dataset is shown. This dataset, like the Epinions1 dataset, has a large-scale, and the General Greedy algorithm cannot run on it because this algorithm has a high running time. Therefore, the run of the General Greedy algorithm on the Slashdot0902 dataset is ignored. According to Fig. 6g, the influence spread of the GIN algorithm is higher than other algorithms. Of course, the High Degree and DDSE algorithms in some *k* nodes have results close to the GIN algorithm. But in general, the GIN algorithm in the Slashdot0902 dataset has shown more influence spread than the compared algorithms. The SAEDV algorithm and then the LIR algorithm in this dataset has the lowest influence spread.

The speedup % (in terms of influence spread) of the GIN algorithm over other algorithms is shown in Table 9. In this table, the General Greedy algorithm called GG and the High Degree algorithm called HD. Due to the General Greedy lack of scalability, its results are not reported for Epinions1 and Slashdot0902 datasets. Table 9 shows the speedup % (in terms of influence spread) of our proposed algorithm GIN over other algorithms. The speedup is computed using Eq. (4). We can see that the GIN algorithm has a positive speedup over other algorithms in this table.

$$Speedup = ((\alpha - \beta)/\alpha) \times 100 \quad (4)$$

Table 9 Speedup % (in terms of influence spread) on different datasets

Datasets	Seed set size	Algorithms									
		GIN	GG	GIN	DDSE	GIN	SAEDV	GIN	LIR	GIN	HD
Email	10	−0.92	0.93	−0.61	0.614	−0.4	0.45	2.13	−2.09	−0.46	0.46
	20	0.35	−0.35	1.37	−1.35	0.34	−0.34	14	−12.3	0.40	−0.40
	30	0.48	−0.48	1.62	−1.60	0.93	−0.92	18.7	−15.7	0.74	−0.73
	40	0.51	−0.50	1.98	−1.94	0.32	−0.32	20.3	−16.9	0.84	−0.83
	50	0.17	−0.17	2.69	−2.62	0.38	−0.38	20.5	−17	0.63	−0.63
NetScience	10	0.28	−0.28	0.53	−0.53	10.1	−9.24	0.67	−0.67	0.38	−0.38
	20	−0.17	0.17	0.21	−0.21	10.6	−9.62	1.36	−1.34	4.42	−4.23
	30	−0.25	0.25	0.51	−0.51	6.08	−5.73	4.06	−3.90	7.58	−7.04
	40	−0.04	0.04	0.53	−0.53	4.86	−4.63	5.35	−5.07	5.05	−4.81
	50	0.04	−0.04	0.13	−0.13	6.02	−5.67	5.18	−4.92	3.75	−3.61
Power	10	0.03	−0.03	4.48	−4.29	14.1	−12.3	9.68	−8.83	3.54	−3.42
	20	0.33	−0.33	3.03	−2.91	6.82	−6.39	4.78	−4.56	2.53	−2.47
	30	0.30	−0.30	2.02	−1.98	5.27	−5.01	3.06	−2.97	5.21	−4.95
	40	0.60	−0.60	1.53	−1.51	4.18	−4.01	2.38	−2.32	1.30	−1.28
	50	0.56	−0.56	1.01	−1.01	3.85	−3.70	3.75	−3.61	2.28	−2.23
NetHEPT	10	−0.55	0.56	2.208	−2.16	62.7	−38.5	4.88	−4.65	23.152	−18.80
	20	−0.02	0.02	1.973	−1.93	67.7	−40.3	10	−9.13	21.541	−17.72
	30	−1.97	2.01	2.721	−2.64	53.8	−35	14.9	−12.9	12.124	−10.81
	40	−1.39	1.41	6.519	−6.12	37.8	−27.4	18.1	−15.3	9.799	−8.92
	50	−0.93	0.94	5.835	−5.51	44.4	−30.7	21.4	−17.6	8.747	−8.043
NetPHY	10	−6.06	6.45	−7.30	−6.197	70.3	−41.2	6.15	−5.79	16.676	−14.29
	20	−3.95	4.12	7.411	−6.90	53.7	−34.9	12	−10.7	7.931	−7.348
	30	−2.07	2.11	5.480	−5.195	86.7	−46.4	16.5	−14.2	8.366	−7.720
	40	−1.96	2	4.741	−4.527	99.8	−49.9	25.6	−20.4	12.171	−10.85
	50	−0.59	0.59	7.882	−7.306	142	−58.7	38.4	−27.7	18.858	−15.86
Epinions1	10	–	–	1.416	−1.396	13.3	−11.8	31.3	−23.8	−0.009	0.009
	20	–	–	2.296	−2.245	27.7	−21.7	60.5	−37.7	−0.388	0.390
	30	–	–	4.101	−3.940	44.5	−30.8	75.6	−43	0.484	−0.482
	40	–	–	3.738	−3.603	22.5	−18.4	87.3	−46.6	0.395	−0.393
	50	–	–	4.610	−4.407	23.8	−19.2	96.1	−49	0.987	−0.977
Slashdot0902	10	–	–	1.161	−1.148	3.59	−3.47	0.90	−0.89	−0.283	0.284
	20	–	–	1.907	−1.871	8.17	−7.55	10.2	−9.32	−0.178	0.178
	30	–	–	0.563	−0.559	7.32	−6.82	17.4	−14.8	0.106	−0.106
	40	–	–	1.418	−1.399	9.13	−8.37	23.9	−19.3	0.109	−0.109
	50	–	–	0.954	−0.945	7.49	−6.96	29.2	−22.6	0.452	−0.450

According to the Eq. (4), α is the influence spread of the GIN algorithm, and β is the influence spread of the other algorithms for different k seed nodes.

The results of the running time algorithms are shown in Fig. 7 on the seven datasets Email, NetScience, Power, NetHEPT, NetPHY, Epinions1, and Slashdot0902. These results are for $k = 50$ and the number of the Monte Carlo simulation $R = 10000$ under the independent cascade model per second. According to Fig. 7, the High Degree and LIR algorithms have the lowest running time in all seven datasets. Of course, the High Degree and LIR algorithms have low the influence spread on more comparable datasets. Then, the running time from lowest to highest belongs to the GIN, DDSE, SAEDV algorithms, and then to the General Greedy algorithm.

Given the results of influence spread and running time of the compared algorithms in Figs. 6 and 7, it is quite obvious that the GIN algorithm has high influence spread along with low running time in all seven datasets.

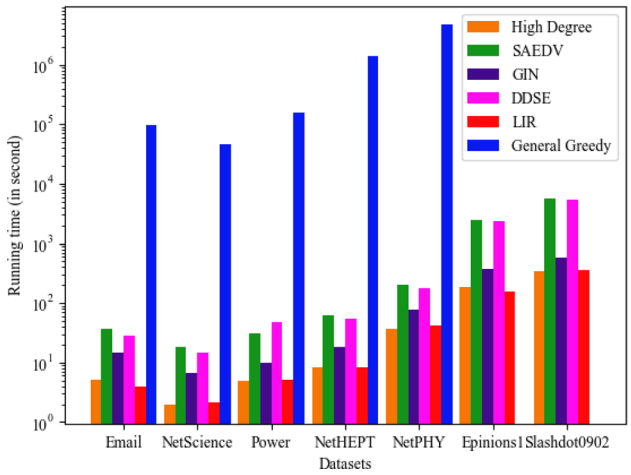


Fig. 7 Diagram of running time of different algorithms for $k = 50$

6 Conclusion

Influence maximization is a problem that has been addressed in social networks analysis and viral marketing. The purpose of this problem is to find the k individual that, by activating them, the highest number of people in a social network can be activated at an acceptable time. For this purpose, the GIN algorithm is presented in this article. In this algorithm, the process of creating different groups, including graph nodes with more number of connections and more common neighbors than other groups, is performed. Hence, the search space to find the final influential nodes is reduced. This process reduces the number of computations and increases the speed of the proposed algorithm. Next, nodes from the new search space that maximize the marginal gain function of the expected diffusion value are selected as the seed node. Experimental results show that the GIN algorithm, in comparison algorithms on all seven real-world datasets, has provided high influence spread along with low running time.

Compliance with ethical standards

Conflict of interest The authors have no conflicts of interest.

References

1. Rui X et al (2019) A reversed node ranking approach for influence maximization in social networks. Appl Intell 49(7):2684–2698
2. 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
3. Banerjee S, Jenamani M, Pratihar DK (2019) ComBIM: a community-based solution approach for the budgeted influence maximization problem. Expert Syst Appl 125:1–13
4. Huang H, Shen H, Meng Z (2019) Community-based influence maximization in attributed networks. Appl Intell 1–11
5. Goldenberg J, Libai B, Muller E (2001) Talk of the network: a complex systems look at the underlying process of word-of-mouth. Mark Lett 12(3):211–223
6. 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
7. Richardson M, Domingos P (2002) Mining knowledge-sharing sites for viral marketing. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM
8. Pei S et al (2017) Efficient collective influence maximization in cascading processes with first-order transitions. Sci Rep 7:45240
9. Chen Y et al (2020) Semantics-aware influence maximization in social networks. Inf Sci 513:442–464
10. Singh SS et al (2019) C2IM: community based context-aware influence maximization in social networks. Phys A Stat Mechan Appl 514:796–818
11. Leskovec J, Adamic LA, Huberman BA (2007) The dynamics of viral marketing. ACM Trans Web (TWEB) 1(1):5
12. Liqing Q et al (2020) TSIM: a two-stage selection algorithm for influence maximization in social networks. IEEE Access 8:12084–12095
13. Srinivasan S, Babu LD (2019) Interest aware influential information disseminators in social networks. SN Appl Sci 1(11):1456
14. Leskovec J, et al (2007) Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM
15. 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
16. Jiang Q, et al (2011) Simulated annealing based influence maximization in social networks. In Twenty-fifth AAAI conference on artificial intelligence
17. Goyal A, Lu W, Lakshmanan LV (2011) Simpath: an efficient algorithm for influence maximization under the linear threshold model. In 2011 IEEE 11th international conference on data mining. IEEE
18. Wang C, Chen W, Wang Y (2012) Scalable influence maximization for independent cascade model in large-scale social networks. Data Min Knowl Disc 25(3):545–576
19. 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
20. He X, et al (2012) Influence blocking maximization in social networks under the competitive linear threshold model. In Proceedings of the 2012 siam international conference on data mining. SIAM
21. Tang Y, Xiao X, Shi Y (2014) Influence maximization: Near-optimal time complexity meets practical efficiency. In Proceedings of the 2014 ACM SIGMOD international conference on management of data
22. Lu W, Chen W, Lakshmanan LV (2015) From competition to complementarity: comparative influence diffusion and maximization. Proc VLDB Endow 9(2):60–71
23. He J-L, Fu Y, Chen D-B (2015) A novel top-k strategy for influence maximization in complex networks with community structure. PLoS One 10(12):e0145283
24. Morone F et al (2016) Collective influence algorithm to find influencers via optimal percolation in massively large social media. Sci Rep 6:30062

25. Liu D et al (2017) A fast and efficient algorithm for mining top-k nodes in complex networks. Sci Rep 7:43330
26. Shang J et al (2017) CoFIM: a community-based framework for influence maximization on large-scale networks. Knowl-Based Syst 117:88–100
27. Wu H et al (2018) LAIM: a linear time iterative approach for efficient influence maximization in large-scale networks. IEEE Access 6:44221–44234
28. Cui L et al (2018) DDSE: a novel evolutionary algorithm based on degree-descending search strategy for influence maximization in social networks. J Netw Comput Appl 103:119–130
29. Xie G et al (2019) MBIC: a novel influence propagation model for membership-based influence maximization in social networks. IEEE Access 7:75696–75707
30. Beni HA, Bouyer A (2020) TI-SC: top-k influential nodes selection based on community detection and scoring criteria in social networks. J Ambient Intell Humaniz Comput 1–20
31. Chen W, Lakshmanan LV, Castillo C (2013) Information and influence propagation in social networks. Synth Lect Data Manag 5(4):1–177
32. Zhou X et al (2015) An ant colony based algorithm for overlapping community detection in complex networks. Phys A Stat Mech Appl 427:289–301
33. Guimera R et al (2003) Self-similar community structure in a network of human interactions. Phys Rev E 68(6):065103
34. Newman ME (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74(3):036104
35. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world'networks. Nature 393(6684):440–442
36. Peng S et al (2018) Influence analysis in social networks: A survey. J Netw Comput Appl 106:17–32
37. Wang W, Street WN (2018) Modeling and maximizing influence diffusion in social networks for viral marketing. Appl Netw Sci 3(1):6
38. Singh SS et al (2019) MIM2: multiple influence maximization across multiple social networks. Phys A Stat Mech Appl 526:120902
39. Tejaswi V, Bindu P, Thilagam PS (2016) Diffusion models and approaches for influence maximization in social networks. In 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE
40. Banerjee S, Jenamani M, Pratihar DK (2018) A survey on influence maximization in a social network. arXiv preprint arXiv:1808.05502
41. Zhong Z, Yang H (2019) Community Structure-based re-ranking information influence maximization algorithm for typhoon disasters. Ekoloji Dergisi 2019(107)
42. Zhu R, et al (2018) QuickIM: efficient, accurate and robust influence maximization algorithm on billion-scale networks. arXiv preprint arXiv:1805.12320