Soft Computing
https://doi.org/10.1007/s00500-021-05749-7

METHODOLOGIES AND APPLICATION

A clique-based discrete bat algorithm for influence maximization in identifying top-k influential nodes of social networks

Lihong Han[1,5] · Kuan-Ching Li[2] · Arcangelo Castiglione[3] · Jianxin Tang[4] · Hengjun Huang[5] · Qingguo Zhou[1]

Accepted: 12 March 2021

Abstract
The problem of identifying the top-k influential node is still an open and deeply felt issue. The development of a stable and efficient algorithm to deal with such identification is still a challenging research hot spot. Although conventional centrality-based and greedy-based methods show high performance, they are not very efficient when dealing with large-scale social networks. Recently, algorithms based on swarm intelligence are applied to solve the problems mentioned above, and the existing researches show that such algorithms can obtain the optimal global solution. In particular, the discrete bat algorithm (DBA) has been proved to have excellent performance, but the evolution mechanism based on a random selection strategy leads to the optimal solution's instability. To solve this problem, in this paper, we propose a clique-DBA algorithm. The proposed algorithm is based on the clique partition of a network and enhances the initial DBA algorithm's stability. The experimental results show that the proposed clique-DBA algorithm converges to a determined local influence estimation (LIE) value in each run, eliminating the phenomenon of large fluctuation of LIE fitness value generated by the original DBA algorithm. Finally, the simulated results achieved under the independent cascade model show that the clique-DBA algorithm has a comparable performance of influence spreading compared with the algorithms proposed in the state of the art.

Keywords Large-scale networks · Top-k influential nodes · Discrete bat algorithm · Clique partition · Diffusion of influence

1 Introduction

With the maturity of 5G technology, the application of Internet of Things, blockchain and other technologies will be very extensive. Social networks, as an abstraction of the interaction between individuals in production and life, will become larger in scale, which will bring challenges to data storage (Wei et al. 2020), calculation (Ogiela and Marek 2020), analysis (Ogiela 2020) and representation (Yan et al. 2020). Network of graphs can represent many relationships between things in the real world. Based on a specific topic in a given network, identifying the top-k influential nodes of such a network is a crucial issue, which relates to widespread applications like information spreading (Zhu et al. 2020), viral marketing (David 2003), network security (Bi et al. 2020; Wei et al. 2019), key node detection of IoT (Wei Liang 2020; Yang et al. 2018; Zhang et al. 2020), recommender systems (Hsieh et al. 2018), cognitive management (Marek and Ogiela 2017; Ogiela and Takizawa 2017), and so on. To some extent, top-k influence node identification is equivalent to an influence maximization (IM) problem. The influence of maximization is an NP-hard problem (David et al. 2003). Therefore, the identification of seed nodes as source spreaders to

✉ Qingguo Zhou
zhouqg@lzu.edu.cn

[1] School of Information Science and Engineering, Lanzhou University, Lanzhou, China

[2] Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

[3] Department of Computer Science, University of Salerno, Fisciano, SA, Italy

[4] School of Computer and Communication, Lanzhou University of Technology, Lanzhou, China

[5] School of Statistics, Lanzhou University of Finance and Economics, Lanzhou, China

Published online: 04 April 2021

produce the largest spreading is the challenge optimization problem.

Up to now, there have been many works on solving the influence maximization problem. In general, the primary methods for top-k influential nodes identification include the centrality-based heuristic method, such as betweenness (Freeman 1977), and eigenvector (Bonacich and Lloyd 2001) centrality, as well as metaheuristic algorithms, such as DPSO (Gong et al. 2016), SAEDV (Jiang et al. 2011), and greedy algorithm (David et al. 2003). In terms of overall effectiveness and efficiency, the centrality-based algorithm is simple but highly demanding in terms of computational complexity. Therefore, this algorithm is not suitable for large-scale social networks. As a matter of comparison, the metaheuristic algorithm has higher efficiency, especially in large-scale networks. However, there is room for further improvement in the effectiveness and stability of some heuristic algorithms.

Recently, algorithms based on swarm intelligence are applied to solve the problems mentioned above, and the existing researches show that such algorithms can obtain the optimal global solution. Jiang et al. (2011) applied the Simulated Annealing (SA) algorithm to obtain the optimal solution of influence maximization. Aybike et al. (2018) employed the Grey Wolf Optimizer (GWO) and Whale Optimization Algorithm (WOA) to solve the influence maximization problem, in which experimental results show that the swarm intelligence approach is effective and efficient. Gong et al. (2016) discretized and reconstructed the PSO algorithm based on the network structure and proposed the DPSO algorithm to identify the top-k most influential social network nodes. Sankar et al. (2016) adopted the bee algorithm to explore the bee colony's waggle dance behavior for identifying influential nodes. Tang et al. (2020) proposed a discrete shuffled frog-leaping algorithm to select top-k influential nodes in a social network. In particular, the Discrete Bat Algorithm (DBA) (Tang and Zhang 2018) has been proved to have excellent performance, but the evolution mechanism based on a random selection strategy leads to the optimal solution's instability.

To solve this problem, in this paper, we propose a clique-DBA algorithm. The proposed algorithm is based on the clique partition of a network and enhances the initial DBA algorithm's stability. The experimental results show that the proposed clique-DBA algorithm converges to a determined Local Influence Estimation (LIE) value in each run, eliminating the phenomenon of large fluctuation of LIE fitness value generated by the original DBA algorithm. Again, the simulated results achieved under the Independent Cascade (IC) model show that the clique-DBA algorithm has a comparable performance of influence spreading compared with the algorithms proposed in state of the art.

The remaining of this article is structured as follows. The problem definition and the diffusion model are given in Sect. 2. Section 3 briefly introduces the basic BA algorithm and the evolution rules of DBA algorithm. Section 4 details the proposed clique-DBA algorithm, including its framework and performance evaluation results. Section 5 shows the clique-DBA algorithm's performance in the six experimental networks and the comparison results with other benchmark algorithms, and finally, the concluding remarks and future work are presented in Sect. 6.

2 Problem definition

Let $G = (V, E)$ denote a social network graph, where $V = [v_1, v_2,\ldots,v_n]$ denotes a set of nodes and $E = [e_1, e_2,\ldots, e_m]$ denotes a set of edges. A node represents an individual actor in a social network graph, and an edge represents the relationship between individuals, such as collaboration, friendship, or certain social relationships. This relationships have a typical feature, that is, individuals are mutual and equal, which can be represented by an undirected social network. The different social relationships can build different network graph. A sample of undirected social graph is depicted in Fig. 1a. In order to study the influence spread, a node's state in the network models is defined as *active* and *inactive*. More precisely, the activated node denotes that the corresponding individual can spread information or influence. In contrast, the inactive node denotes that active nodes can influence the corresponding individual.

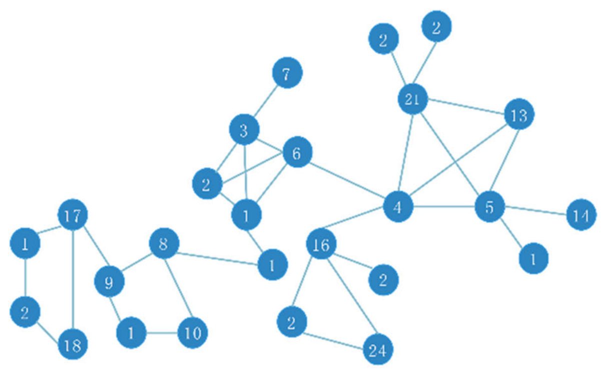
Definition 1 (**Top-k Influential nodes**). The top-k influential nodes are defined as a specified number of nodes within a social network that has significant influence under a specific topic. It aims at identifying a set of nodes of size k based on a specific topic or a model whose influence in the social network is greater than that of any node outside the set.

Formally, the influence of node v under a specific model or topic is denoted as $f(v)$. The influence S' and the relationship of the top-k influential nodes can be expressed by Eq. (1) as follows:

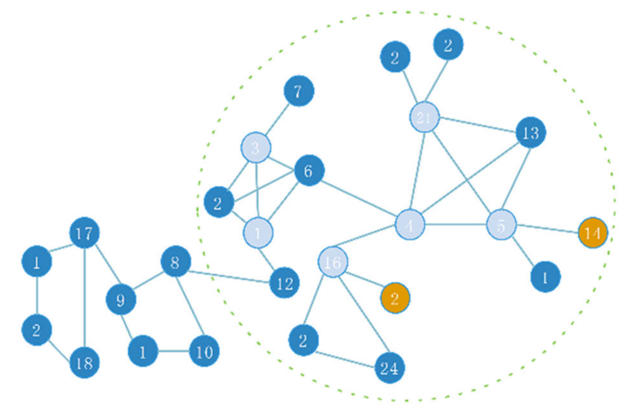
$$S' = \sum_{v=1}^{k} f(v)[|S'| = k, f(1) \geq f(2) \geq \cdots f(v) \cdots \geq f(k)] \tag{1}$$

where k is the user-specified number of nodes.

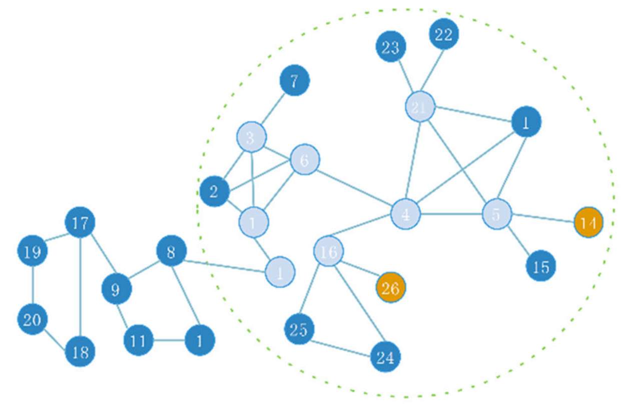
Definition 2 (**Influence Maximization Problem**). It is an optimization problem that requires identifying a set of



a A sample graph of social network



b The diagram of candidate seed nodes when the size of candidate seed node pool is 3 times the number of seed nodes (yellow node represents the node currently active, and the gray node represents the nodes of candidate seed node pool).



c The diagram of candidate seed nodes when the size of candidate seed node pool is 4 times the number of seed nodes (yellow node represents the node currently active, and the gray node represents the nodes of candidate seed node pool).

Fig. 1 A sample social graph and the nodes of candidate seed node pool

nodes S as seed nodes that could maximize the influence spread. The influence maximization of S, denoted as S^{*}, is measured by the expected number of active nodes at the end, under the specific influence diffusion model, denoted as $d(S)$. Their relationship can be expressed by Eq. (2) as follows:

$$S^{*} = argmax[d(S)]\{S \in V, |S| = k\} \tag{2}$$

where k is the size of the seed node set.

Definition 3 (Independent Cascade Diffusion model). The Independent Cascade (IC) is an information dissemination model, which is an abstract model of the information dissemination process. The basic assumption of this model is that the success of the attempt of node u to activate its neighboring node v is an event with probability $p_{u,v}$. The probability that a node in an *inactive* state is activated by a newly *active* neighbor node is independent of the neighbor's activity that has tried to activate the node before. In addition, the model also makes the assumption that any node u in the network has only one chance to try to activate its neighbor node v. No matter whether it succeeds or not, node u remains *active*, but it no longer has influence in the later time. This kind of node is called an active node without influence. The basic propagation process of the IC model is as follows:

1. The initial set of active nodes S.
2. At time t, the newly activated node u has an influence on its adjacent node v, and the probability of success is $p_{u,v}$.
3. If node v is successfully activated, then at time $t + 1$, node v becomes *active*, which will influence its neighboring inactive nodes; otherwise, the state of node v does not change at time $t + 1$.
4. The above process is repeated continuously until there is no active node with influence in the network and the propagation process ends.

Definition 4 (Clique). A clique is essentially a complete subgraph that usually includes overlapping and nonoverlapping clique. An overlapping clique means that each node of a clique can belong to more than one clique, while a nonoverlapping clique denotes that the nodes in a clique belong only to their clique. In this paper, clique, unless otherwise specified, is the nonoverlapping clique. According to the clique's definition, a clique containing N nodes must have $\frac{N(N-1)}{2}$ edges in an undirected graph.

Definition 5 (Local Influence Estimation, LIE). The LIE function is expressed by Eq. (3) as follows:

$$\begin{aligned}\text{LIE} &= \sigma_0(S) + \sigma_1^*(S) + \sigma_{\tilde{2}}(S) \\ &= \text{k} + \sigma_1^*(S) + \frac{\sigma_1^*(S)}{\left|N_S^{(1)} \backslash S\right|} \sum_{u \in N_S^{(2)} \backslash S} p_u^* d_u^* \\ &= \text{k} + \left(\frac{1}{\left|N_S^{(1)} \backslash S\right|} \sum_{u \in N_S^{(2)} \backslash S} p_u^* d_u^* \right) \sum_{i \in N_S^{(1)} \backslash S} \left(1 - \prod_{(i,j) \in E, j \in S} (1 - p_{i,j}) \right)\end{aligned} \tag{3}$$

where $N_S^{(1)}$ and $N_S^{(2)}$ are the sums of the degree values within the range from one-hop to two-hop of each node in the node set S, respectively; p_u^* is the constant probability of successfully activating its neighbor node, and d_u^* is the number of edges for node u within the range of one-hop and two-hop. Some studies have shown that the LIE evaluation function has a good optimization effect (Gong et al. 2016; Tang et al. 2020). This objective function is the suite for the influence spread approximation of large-scale social networks in the IC model.

3 Basic bat and discrete bat algorithm

Based on bat foraging behavior simulation, Yang et al. (2012) proposed the bat algorithm (BA). The BA is a novel metaheuristic algorithm, whose performance has been proven better than genetic algorithms (GA) and particle swarm optimization (PSO) algorithms. Again, the bat algorithm has been further optimized to obtain a better global solution and expanded its application scope. For example, Tang et al. (2018) redesigned the BA in the discrete space to propose Discrete Bat Algorithm (DBA), which is used to solve the problem of influence maximization in the discrete network space.

3.1 Basic BA and its evolution rules

In the basic BA, to simulate bat to detect prey and avoid obstacles, three approximate or idealized rules should be assumed. First, use echolocation to perceive distance and distinguish between prey and obstacles. Second, the transmitting pulse wavelength (or frequency) and the pulse emissivity are automatically adjusted according to their proximity to the target. Third, it is always assumed that bat loudness value varies from a maximum to a fixed minimum. Therefore, in the basic BA, the evolution process and the adjustment of auxiliary parameters are carried out according to the following rules.

3.1.1 Rules of bat evolution

In the tth generation of the basic bat algorithm, the information in bat i $(i = 1, 2, \ldots, N)$ can be expressed by a quintuple T:

$$T = \langle \underset{i}{\overset{\rightarrow}{\mathrm{x}}}(t), \underset{i}{\overset{\rightarrow}{\mathrm{v}}}(t), f_i(t), A_i(t), r_i(t) \rangle \tag{4}$$

where N is the number of virtual bats within the definition domain. The position vector $x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iM}(t))$ represents the position information of bat i in the tth generation, and it is a solution in the search space. The velocity vector $v_i(t) = (v_{i1}(t), v_{i2}(t), \ldots, xv_{iM}(t))$ represents the speed and direction of the bat i during the evolution of foraging behavior in generation t.

Based on the above definitions, the mathematical expression of the velocity vector for $t + 1$ generation is expressed by Eq. (5):

$$v_{ik}(t+1) = v_{ik}(t) + \left(x_{ik}(t) - p_k^*(t)\right) \cdot f_i(t) \tag{5}$$

where p_k^* represents the historically optimal position of bat population discovered in the previous t generation, and $\left(x_{ik}(t) - p_k^*(t)\right) \cdot f_i(t)$ represents the influence of the deviation between $x_i(t)$ and $p_k^*(t)$ on the velocity of the next generation. Again, based on the evolution of the velocity vector in Eq. (5), a random selection strategy was used to determine the position vector's evolution for the bat i. The mathematical expression of evolution is given by Eqs. (6–7):

$$x'_{ik}(t+1) = \begin{cases} x_{ik}(t) + v_{ik}(t+1) \; if \; \text{rand}_1 < r_i(t) & (6) \\ p_k(t) + \varepsilon_{ik} \cdot \overline{A}(t) \; Otherwise & (7) \end{cases}$$

where $rand_1$is a random number between 0 and 1, subject to a uniform distribution. ε_{ik} is a random number between (− 1,1), satisfying the uniform distribution. $\overline{A}(t)$ is the average loudness of the bat at time t.

3.1.2 Auxiliary parameters of basic BA

The frequency $f_i(t)$, loudness $A_i(t)$ and pulse emission rate $r_i(t)$ are three important auxiliary parameters of the bat i in the basic BA algorithm.

In detail, the frequency $f_i(t)$ is randomly generated according to Eq. (8):

$$f_i(t) = f_{\min} + (f_{\max} - f_{\min}) \cdot \text{rand}_2 \tag{8}$$

where rand_2 is a random number between 0 and 1, satisfying a uniform distribution, while the two parameters $f_{\max}$ and $f_{\min}$ are the preset upper and lower frequency limits, respectively.

The loudness parameter $A_i(t)$ determines the bat's local searching ability. The setting and evolution processes of loudness parameter are expressed by Eq. (9):

$$A_i(t+1) = \alpha A_i(t) \tag{9}$$

The updated formula of pulse emission rate $r_i(t+1)$ is defined by Eq. (10):

$$r_i(t+1) = r_i(0) \cdot (1 - e^{-\gamma t}) \tag{10}$$

where $\alpha > 0$ and $r > 0$ are preset parameters, $A_i(0)$ is the initial value of loudness, and $r_i(0)$ is the initial value of pulse emission rate.

3.2 DBA and its framework

The DBA algorithm proposed by Tang et al. (2018) is suitable for top-k influence nodes identification in the discrete network space and can achieve more comparable effectiveness in some real networks' experiments.

3.2.1 Evolution rules of DBA

For the DBA algorithm, each bat can be represented by k nodes of a network, i.e., k nodes form a set of vectors to represent individual bats' location. Each node in the position evolves according to bat swarm intelligent foraging rules. Therefore, to solve the influence maximization in the social network solution space, the encoding rules of position and velocity should be redesigned. In this way, k candidate nodes of a social network represent each bat in the bat population. For example, the position vector of the bat i can be expressed by a k-dimensional vector $x_i(x_{i1}, x_{i2}, \ldots, x_{ik})$, where $i = (1, 2, \ldots, N)$. Again, for velocity encoding, different from position vector encoding, each bat's velocity is encoded by 0 or 1.

The specific meaning of encoding 0 and 1 is as follows. If $v_{ij} = 0$, the corresponding node in the temporary optimal vector in current evolution remains unchanged. On the other hand, if $v_{ij} = 1$, it means that the corresponding node x_{ij} needs to adopt a new node in the candidate node set for replacement and evolution. According to the encoding rules mentioned above, the form of DBA evolutionary rules can be expressed by Eqs. (11–12):

$$V_i^{t+1} \leftarrow H\left[V_i^t + \left(X_* \cap x_i^t\right) \cdot f_i\right] \tag{11}$$

$$X_i^{t+1} \leftarrow X_i^t V_i^{t+1} \tag{12}$$

where X_i is coded by the integer number of the node ID and denotes the position vector of the bat i; $V_i = (0, 1)$ is the velocity vector, where 0 indicates the node corresponding to x_i in the position vector that does not need to be adjusted, and 1 represents the node corresponding to x_i in the position vector that needs to be adjusted; parameter f_i denotes the latter part's inertia weight, which is used to dynamically adjust the rate towards the optimal position of the bat i; operator "$H(\cdot)$" is a decision operator, and operator "$\cap$" is a logical similar intersection operation. Equations (11) and (12) are the discretized form of evolution rules given by Eqs. (5–7) in the basic BA. For more details of the DBA algorithm, the readers can refer to (Tang and Zhang 2018).

3.2.2 The framework of DBA

Based on the encoding and evolutionary rules mentioned above, the position and velocity vectors are iterated and evolved to explore the optimal global solution. To avoid the blindness of bat population searching in the solution space, Tang et al. (2018) proposed a searching mechanism that combines a greedy strategy and a random walk strategy.

Moreover, based on the degree and closeness centrality, the author constructs a search pool of candidate nodes to improve the optimal global solution's exploring ability. The framework pseudo-code of the basic DBA algorithm for identifying top-k influence nodes is listed in Algorithm 1.

Algorithm 1. The Framework of DBA.

Require: Network $G = (V, E)$, maximum generation g_{max}, bat population size N, frequency bounds f_{min}, and f_{max}.

```
01: Initialize evolutionary generation variable $g = 0$
02: Initialize pulse rate $r$ with a random number between 0 and 1
03: Initialize position vector $X$ based on degree centrality
04: Initialize velocity vector $V$ with 0 or 1
05: Select out initial best position vector $X^*$
06: Repeat:
07:   For each bat $x_i \in X$ do:
08:     update the position vector $X_i$ according to Eq. (9)
09:     update the velocity vector $V_i$ according to Eq. (8)
10:     if $rand_1 > r_i$ then
11:       $x_i \leftarrow$ LocalSearch($x_i$,$G$)
12:     end if
13:     $x_{i\_new} \leftarrow$ RandomWalk($x_i$, CandidatePool,k)
14:     if LIE($x_{i\_new}$) > LIE($x_i$) and rand2 < $A_i$ :
15:       Accept $x_{i\_new}$
16:     end if
17:     update $A_i \leftarrow \alpha A_i$
18:     update $r_i \leftarrow r_{i\_0}[1 - e^{-rg}]$
19:   End for
20:   select out the best global position vector $X^*$ of the current generation
21:   $g \leftarrow g+1$
22: Until $g == g_{max}$
Return: The global best solution $X^*$
```

4 The proposed algorithm

4.1 Motivation

From the evolution process of the above basic BA algorithm, it can be seen that the convergence of BA is mainly based on iterations of three random numbers that obey the uniform distribution. The first scenario that relies on a random number is as Eq. (6), in which the generation of a new position vector must meet the condition $rand_1 < r_i(t)$. The second scenario that relies on a random number is as Eq. (7), in which the generation of the new position vector depends on the random number ε, whose value is between − 1 and 1 and meets uniform distribution. The third scenario that relies on a random number is Eq. (8), where the generation of frequency $f_i(t)$ depends on the random number rand_2.

In the DBA algorithm, in order to avoid the blindness of exploiting the optimal solution a random walk strategy from the candidate pool is introduced to generate a new node of position vector for the bat *i*. In detail, the construction of the candidate seed node pool is based on the degree and closeness centrality of nodes in the network; again, their weights each account for 50% to calculate each node's contribution. The candidate seed node pool constructed based on the above strategy can avoid the blindness of bat exploration to a certain extent, and can improve the algorithm's convergence speed. However, due to node contribution repetitiveness, the candidate seed node pool constructed based on the above strategy cannot wholly avoid the DBA algorithm's instability. In order to verify the optimization effect of the DBA algorithm in solving top-k influence nodes identification, we run the DBA algorithm ten times in different experimental networks, and the boxplot graph of the LIE fitness value is shown in Figs. 5 and 6. We can observe that the LIE value calculated in each run is different, and the difference is quite large. Through careful analysis, we find that the function of the candidate seed node pool in the DBA algorithm is not only to avoid the blindness of the algorithm in the evolution of new position vectors of bats but also the diversity of the nodes in the candidate seed node pool is related to the global search ability of the algorithm. The strategy based on degree and closeness centrality does not diversify the nodes of the candidate seed node pool.

Take Fig. 1a as an example, assuming that the initial active nodes are node 24 and node 25. As shown in Fig. 2b, the gray nodes are the candidate seed node pool nodes constructed by the strategy of the DBA algorithm. If the size of the candidate seed node pool is expanded to 4•K, the candidate seed nodes are still concentrated near the fully connected nodes of the network, as shown in Fig. 2c. Moreover, experiments have proved that when the size of the candidate seed node pool is between 3•K and 4•K, the DBA algorithm has the best performance. It is worth mentioning that when the candidate seed node pool is too large, although the diversity of the nodes in the candidate seed node pool is enhanced, the convergence and stability of the algorithm will decrease.

From the above analysis, we can see that improving the diversity of node distribution without expanding the size of the candidate seed node pool is an effective way to improve the global exploration ability of the algorithm. Therefore, in this paper, we proposed the Clique_DBA algorithm based on the Clique partition of the network structure.

4.2 The framework of Clique_DBA

Based on the analysis mentioned above in Sect. 4.1, the flowchart of the proposed Clique_DBA algorithm is shown in Fig. 3. In the framework of Clique_DBA, we can see that it mainly includes six steps.

Step 1: Initialize some of the parameters involved in the algorithm in this step.

Step 2: Clique division is performed on the network in this step, and the candidate seed node pool is constructed on the basis of clique partition. It is worth mentioning that when selecting candidate seed nodes for each clique, we choose the node with the largest degree value to construct the candidate seed node pool instead of random selection, which is more conducive to the convergence of the algorithm.

Step 3: Initialize the position vector *X* of the bat population, and the velocity vector *V*. Select the initial optimal solution position vector *X** from the initialized bat population.

Step 4: The evolution of bat population. In the each generation of evolution, N individuals of bat population are successively evolved according to Eq. (6) or Eq. (7). At the same time, under the control of uniformly distributed random numbers, local search and global exploration are performed. Among them, function LocalSearch() implements local search, and function RandomExplor() implements global exploration.

Step 5: After each generation of evolution process is over, the current global optimal solution position vector *X** is selected.

Step 6: Check termination criterion. Steps 4–5 are repeated until $g_{\max}$ is reached. After the iteration, the current optimal solution *X** is the global optimal solution.

It is worth mentioning that in the iterative process, the loudness A_i of the bat *i* will decrease in the process of approaching the global optimal solution, while the rate of pulse emission r_i is increasing. Their values can be expressed by Eqs. (13) and (14):

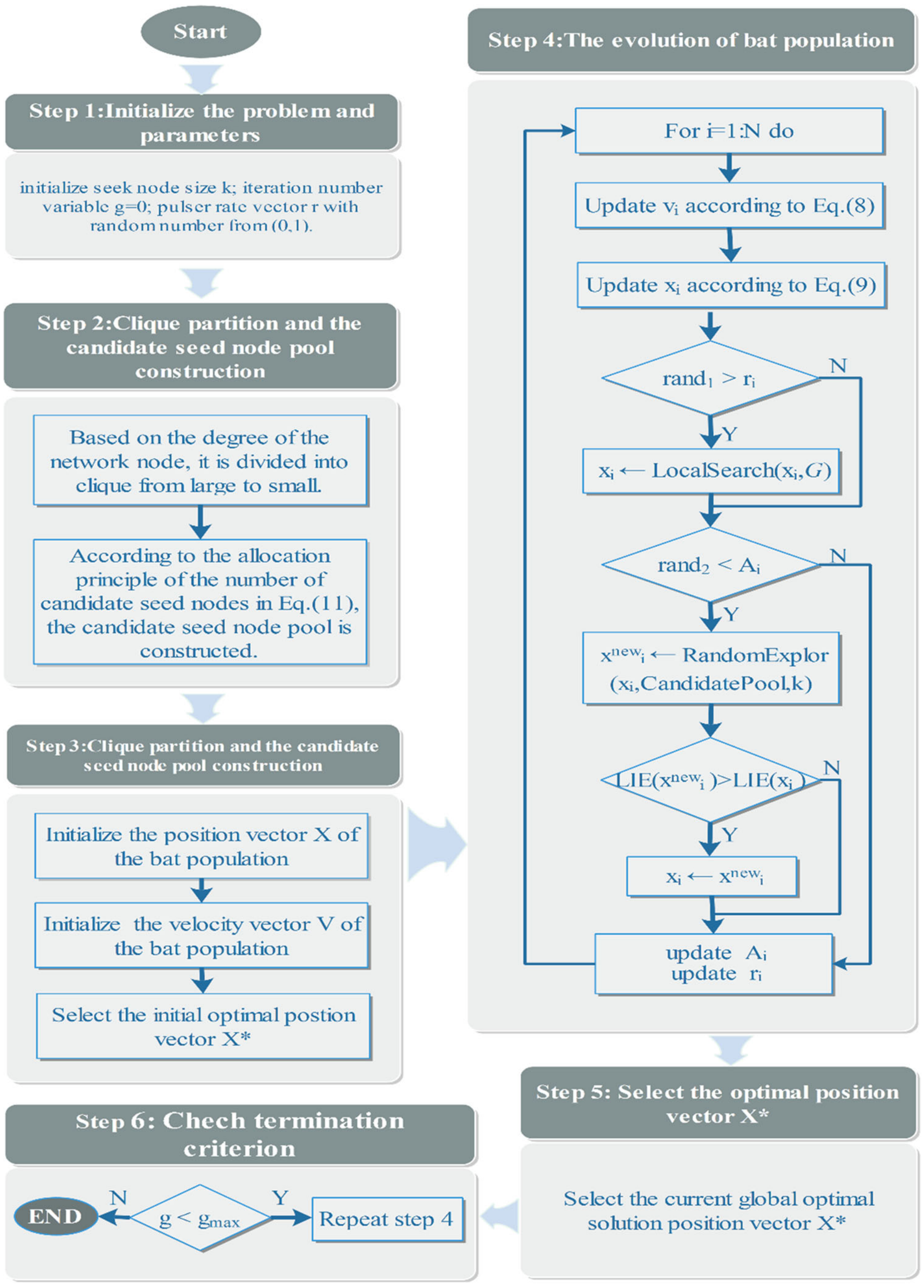

Start
Step 1:Initialize the problem and parameters
initialize seek node size k; iteration number variable g=0; pulser rate vector r with random number from (0,1).
Step 2:Clique partition and the candidate seed node pool construction
Based on the degree of the network node, it is divided into clique from large to small.
According to the allocation principle of the number of candidate seed nodes in Eq.(11), the candidate seed node pool is constructed.
Step 3:Clique partition and the candidate seed node pool construction
Initialize the position vector X of the bat population
Initialize the velocity vector V of the bat population
Select the initial optimal postion vector X*
Step 4:The evolution of bat population
For i=1:N do
Update v_i according to Eq.(8)
Update x_i according to Eq.(9)
$rand_1 > r_i$
N
Y
$x_i \leftarrow$ LocalSearch(x_i,G)
$rand_2 < A_i$
N
Y
$x^{new}_i \leftarrow$ RandomExplor (x_i,CandidatePool,k)
LIE(x^{new}_i)>LIE(x_i)
N
Y
$x_i \leftarrow x^{new}_i$
update A_i
update r_i
Step 5: Select the optimal position vector X*
Select the current global optimal solution position vector X*
Step 6: Chech termination criterion
END
N
$g < g_{max}$
Y
Repeat step 4

Fig. 2 Flowchart of Clique_DBA for performing optimization in solve IM problem of social networks

$$A_i^t \rightarrow 0 (t \rightarrow \infty) \tag{13}$$

$$r_i^t \rightarrow r_i^0 (t \rightarrow \infty) \tag{14}$$

4.3 Cliques partition

Based on the definition of clique in Definition 4, we use the clique division method based on the maximum degree value to divide the network's clique structure. Taking the network in Fig. 1a as an example, the process of the clique partition method based on the maximum degree value is shown in Fig. 3. Starting from the node with the maximum degree value, we find the maximum clique in the network until there is no clique structure in the network. The pseudo-code of the clique partition algorithm is given in Algorithm 2.

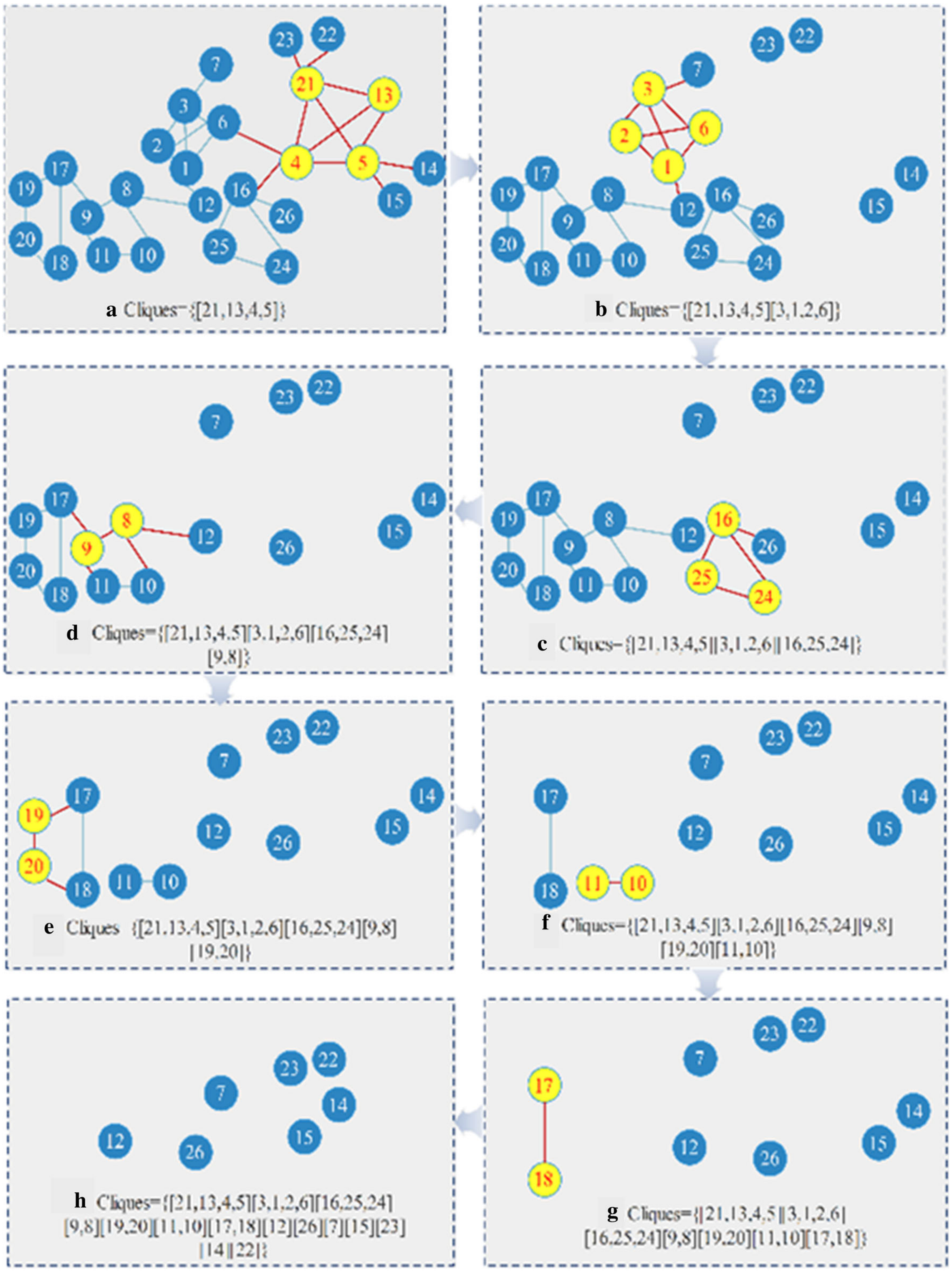

a Cliques={[21,13,4,5]}
b Cliques={[21,13,4,5][3,1,2,6]}
c Cliques={[21,13,4,5][3,1,2,6][16,25,24]}
d Cliques={[21,13,4,5][3,1,2,6][16,25,24][9,8]}
e Cliques={[21,13,4,5][3,1,2,6][16,25,24][9,8][19,20]}
f Cliques={[21,13,4,5][3,1,2,6][16,25,24][9,8][19,20][11,10]}
g Cliques={[21,13,4,5][3,1,2,6][16,25,24][9,8][19,20][11,10][17,18]}
h Cliques={[21,13,4,5][3,1,2,6][16,25,24][9,8][19,20][11,10][17,18][12][26][7][15][23][14][22]}

Fig. 3 Schematic diagram of the Clique structure partition process in the network

Algorithm 2. Cliques partition based on the degree of nodes

```
Require: Network G = (V, E)
01. Initialize Clique_set as null
02. Initialize cliques number variable Clique_num as 0.
03. Edges ← ⟨E⟩
04. Nodes ← |V|
05. While Edges is not null:
06.     clique ← FindMaxClique(G)
07.     clique_set ← Clique
08.     deleteNode(Nodes, Clique)
09.     deleteEdge(Edges, Clique)
10.     clique_num ← Clique_num+1
11.  End While
12.  Clique_set ← Nodes
13.  Return: Clique_set, Clique_num
```

In Algorithm 2, the function FindMaxClique() finds the largest clique in the social network. This function traverses the nodes in turn, according to the degree value, from large to small. At the end of the loop, cliques are sorted in the clique set variable Clique_set according to the size of the clique. Again, the function DeleteNode() removes the nodes that make up the clique from the nodes set variable Nodes. The function DeleteEdge() removes all the edges connected to the nodes in the clique from the edge set variable Edges. Finally, the statement on line 12 stores all the isolated nodes in the clique set.

4.4 Candidate seed nodes allocation

The size of the candidate seed nodes pool should be related to the number of seed nodes of the algorithm. In particular, this number should not be too small to affect the optimal global solution's effectiveness or too large to affect the efficiency of the searching process for the optimal global solution. By setting candidate seed node pools of different sizes, experiments in the six experimental networks found that when the candidate seed node pool size is usually set to 3 K or 4 K, its optimization performance is the best and the optimization efficiency is the highest. Therefore, to maximize the candidate seed node pool's effectiveness in different network structures, we define the candidate node pool's size in a range that can be dynamically adjusted according to the network structure in Algorithm Clique-DBA. Formally, Eq. (15) calculates the detailed partition of the number of candidate nodes:

$$\mathrm{Clique}_{\mathrm{number\_seed}} = \begin{cases} 3\cdot k & \textit{If } C_{\mathrm{num}} \le 3\cdot k \\ C_{\mathrm{num}} & \textit{If } (3\cdot k) < C_{\mathrm{num}} < 4\cdot k \\ 4\cdot k & \textit{If } C_{\mathrm{num}} \ge 4\cdot k \end{cases} \tag{15}$$

According to the above definition of the size of the candidate seed nodes pool, the number of candidate seed nodes corresponding to each clique is allocated based on the following rules:

- If $C_{\mathrm{num}} \le 3\cdot k$, the number of candidate seed nodes allocated in each clique is calculated by Eq. (16) as follows:

$$\mathrm{Clique(i)} = \left\lceil \frac{\mathrm{Clique}(i)_{\mathrm{size}}}{|\mathrm{Nodes}|} \cdot 3\cdot k \right\rceil \tag{16}$$

  where $\mathrm{Clique}(i)_{\mathrm{size}}$ is the number of nodes in the clique(i), and |Nodes| is the size of the network.
- If $(3\cdot k) < C_{\mathrm{num}} < 4\cdot k$, a random node is selected from each clique to form the candidate seed node pool.
- If $C_{\mathrm{num}} \ge 4\cdot k$, the first $3\cdot k$ candidate nodes are selected from the top $3\cdot k$ largest cliques, where one node is randomly selected from each clique. For the remaining K nodes, select out K cliques randomly from the remaining $(C_{\mathrm{num}} - 3\cdot k)$ cliques, and then randomly select one node from each selected clique.

According to the above allocation rules, the selected candidate seed nodes constitute a node pool of candidate nodes.

4.5 Computational complexity analysis

From the above framework of the clique-DBA algorithm, the cliques' detection needs $O(k\cdot N)$ operations, and the ordering and replacement operations require $O(k\cdot \log k)$ and $O(k\cdot \overline{D})$ operations, respectively. The initialization and updating velocity vector operations require $O(k\cdot N)$ and $O(k\cdot \log k\cdot N)$ operation units, respectively. The random exploration in the candidate node pool requires $O(k)$ operation units, and the evaluation of LIE value requires $O(k\cdot \overline{D})$ operations.

Therefore, the total computation upper limit of the clique-DBA algorithm is $O(N\cdot K\cdot((\log k + \overline{D})\cdot g\max + 1))$ operating units. Compared to the basic DBA, clique-DBA's computational complexity is different from that of the clique detection since the clique-DBA's time complexity includes the cliques' detection computation and the bat population evolution.

5 Experimental results

This section shows the results achieved in the experiments carried out to verify the clique-DBA algorithm's effectiveness in six real-world social networks. Then, we compare clique-DBA with other state-of-the-art algorithms' performance in the same real-world networks. The simulation program is coded by the C++ language. The experiment is conducted on a Server platform equipped with an Intel Xeon E5 CPU, 16 GB RAM, with Windows Server 2003 (Service Pack 2) OS installed. The experimental simulation is based on the IC model, and the maximum seed node set is set to 50.

5.1 Experimental networks and baseline algorithms

5.1.1 Experimental networks

To make verification and comparison more objective, we conduct extensive experiments on six real-world undirected social networks. The statistical characteristics of the six experimental networks are shown in Table 1.

SynRand is a randomly generated synthetic network consisting of 14,991 nodes and 56,152 edges. Its node degree obeys the Gaussian distribution. The Pretty Good Privacy encryption algorithm represents the basis of the PGP social network. It consists of 10,680 nodes that represent the people who share confidential information. All experimental social networks get from SNAP1. The node degree distributions of each experimental network are shown in Fig. 4.

Table 1 Statistical characteristics of the six real-world networks. $|V|$ is the number of nodes, $|E|$ is the number of edges. $<K>$ represents the average degree of network, $<d>$ represents the average path distance, $<C>$ is the average clustering coefficient of the network, D is the density of the network

Networks	$\|V\|$	$\|E\|$	$<K>$	$<d>$	$<C>$	D
NetScience	379	914	4.824	6.042	0.798	0.013
Email	1133	5452	9.624	3.606	0.254	0.009
HepTH	9877	25998	5.26	5.945	0.601	0.001
PGP	10680	24316	4.554	7.486	0.44	0.001
CondMat	23133	186936	16.156	5.353	0.706	0.001
SynRand	14991	56152	7.492	5.006	0.001	0.001

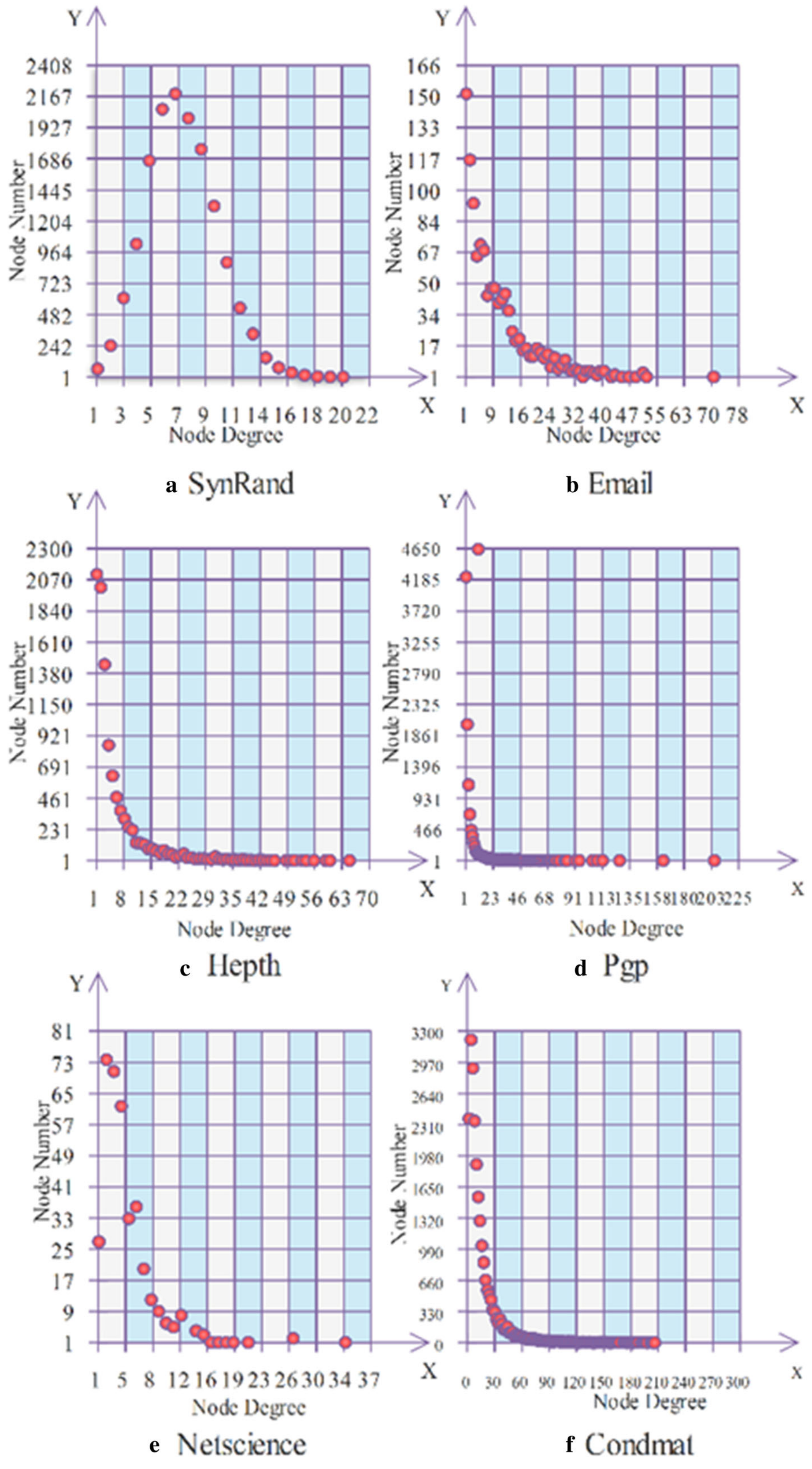
*NetScience (Newman 2006), Email (Guimerà et al. 2004), HepTH (Lu et al. 2017), PGP (Gregory 2009), CondMat (Leskovec et al. 2007a)

5.1.2 Baseline algorithms

Five state-of-the-art algorithms are used as baseline algorithms, and the influence spread performance is simulated under the IC diffusion model for six algorithms, respectively. In detail, we consider the following baseline algorithms for comparison:

- DBA (Discrete Bat Algorithm) (Tang and Zhang 2018) is a metaheuristic algorithm used to optimize the network space solution after discretizing the basic Bat Algorithm. The basic Bat Algorithm is a swarm intelligence algorithm based on the foraging behavior of the bat population. It is a swarm intelligent optimization algorithm combining the evolutionary rules of basic bat population with the greedy search strategy based on probability.
- DPSO (Discrete Particle Swarm Optimization) (Gong et al. 2016) is a discretized form of the basic Particle Swarm Optimization (PSO) algorithm. The basic PSO algorithm solves the optimization problem in reality by mimicking the foraging behavior of flocks. By discretizing and reconstructing the evolutionary behavior of birds foraging according to the network structure, DPSO is applied to the optimization problem of node identification in the network space. Through experiments in real networks, it is found that it has comparable performance and efficiency in solving the problem of maximizing the influence of nodes in the large-scale networks.
- ELDPSO (Enhanced Local DPSO) (Tang et al. 2018) integrates the ascending orders of node degree centrality into the local search operation of DPSO. It uses the ascending sequence of candidate node degree to enhance the local search ability, to prevent the algorithm from falling into a premature solution.
- The greedy algorithm (David et al. 2003) calculates each node's influence and selects the top-k nodes maximizing the marginal value. More precisely, a greedy strategy is adopted to select the node u with the maximum marginal return and add such a node to the seed set S in each iteration. However, due to the many iterations necessary to calculate the influence spread, the efficiency of the algorithm is very low.
- SD (SingleDiscount) (Chen et al. 2009) is a simple degree discount heuristic algorithm. In detail, it is a modified degree centrality algorithm, where each neighbor of a newly selected node discounts its degree by one.
- CELF (Cost-Effective Lazy Forward) (Leskovec et al. 2007b) is a greedy algorithm based on the lazy forward-selection strategy, which effectively reduces the computation complexity. Indeed, this algorithm is 700 times more efficient than the standard hill-climbing greedy algorithm.

Fig. 4 Degree distribution characteristic chart of six experimental social networks



5.2 Comparison of LIE evaluation

To validate the clique-DBA algorithm's performance, we carried out an experiment for evaluating the LIE values on the six aforementioned social networks.

In detail, the experiments were carried out under two scenarios of propagation probability, that is, $p = 0.01$ and $p = 0.05$, respectively. In the experiment, the clique-DBA algorithm's parameter setting adopts the parameters that have been evidenced to achieve the DBA algorithm's optimal performance. Besides, to verify the clique structure's improvement effect on the DBA algorithm's stability, we also conducted experiments on the improved DBA algorithm based on the community structure similar to the network's clique structure and then compared the results. Therefore, another LIE evaluation result of improved algorithm based on the community structure, called Community-DBA, was experimented under the same experimental environment.

The community-DBA is similar to clique-DBA. Indeed, it uses similar allocation rules of candidate seed node and construction methods of candidate seed node pool as the clique-DBA algorithm. The only difference is to change the clique structure of the algorithm to the community structure. In the community-DBA algorithm, the Louvain algorithm (Blondel et al. 2008) is used to detect the community structure. When the propagation probability is $p = 0.01$, the evolutionary processes of three algorithms is shown in Fig. 5, where the performance of the clique-DBA algorithm is the best in terms of stability. Compared with the original DBA algorithm, the community-DBA algorithm's stability is also much improved, but it is still inferior to the clique-DBA algorithm. Again, both community-DBA and clique-DBA can achieve a LIE value comparable to the original DBA in all the experimented networks shown in Fig. 5. However, we remark that the clique-DBA algorithm's performance is the best in LIE value, especially in the SynRand network. When $p = 0.05$, the clique-DBA algorithm's performance is similar to that when $p = 0.01$ and its evolutionary processes are shown in Fig. 6.

5.3 Comparison of typical algorithms

To show the proposed clique-DBA algorithm's performance on influence spread, we choose the six related state-of-the-art algorithms mentioned in Sect 5.1.1 as baseline algorithms. Then, MC simulation is used to evaluate the spreading performance of the algorithms under the IC model. The simulated evolutionary performances are shown in Figs. 7 and 8.

From the simulated curves, we can observe that clique-DBA achieves comparable performance, more effective than the original DBA algorithm. In particular, in the scenarios depicted in Fig. 7b, c, the performance of influence diffusion of the clique-DBA algorithm is even better than that of the greedy algorithm. The clique-DBA is more robust and effective than the original DBA algorithm, especially when dealing with large-scale social networks.

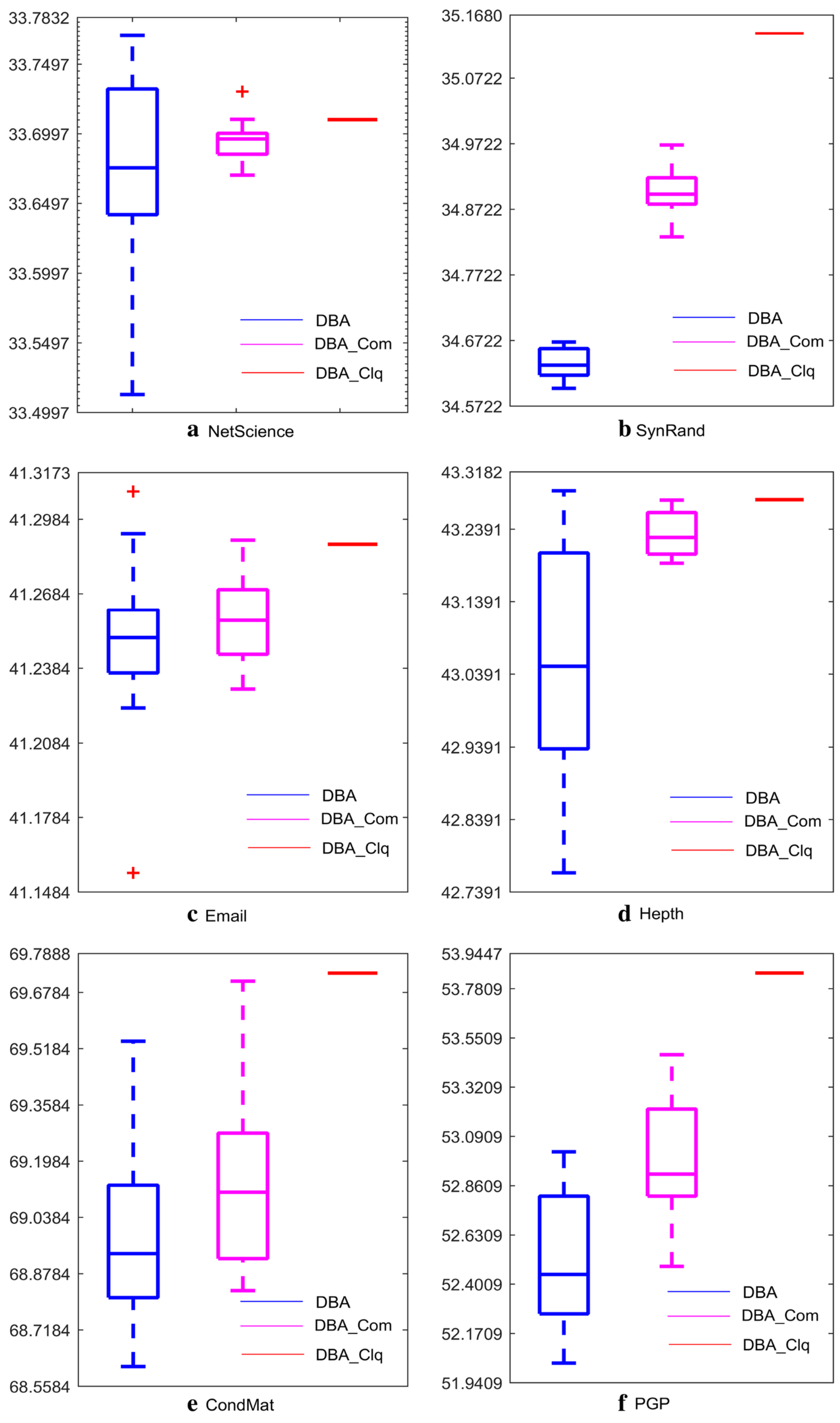
Moreover, as shown in Fig. 8a, f, from the influence propagation scale obtained by simulating the seed node set, we can observe that the greedy algorithm has the most outstanding performance when $p = 0.05$. However, the computational complexity is high, especially in a large-scale social network. Again, the DBA algorithm is not stable enough because of its random selection strategy. This aspect is evident in Fig. 8f. The CELF algorithm has good propagation performance in the six social networks. However, the CELF algorithm is not suitable for large social networks due to its high computational complexity. In summary, the clique-DBA algorithm is more effective and stable in large-scale social networks as the seed node set increases. This algorithm is also suitable for identifying the most top-k influential node set in large-scale networks when a larger size of seed node set is required.

5.4 Comparisons of the running time

The comparison of running time (the case of 30 seed nodes) under the propagation probability $p = 0.01$ is shown in Fig. 9.

In this figure, we can observe that the greedy algorithm is the most time-consuming. This algorithm takes 39 k seconds, 13 k seconds, 19 k seconds, 29 k seconds, 2 k seconds, and 32.5 k seconds in the six experimental networks, respectively. Furthermore, as shown in Fig. 9, the computing time of clique-DBA is slightly higher than that of the initial DBA algorithm in the experimental networks. This aspect is caused by the division of the clique structure of clique-DBA before the evolutionary processing. However, we remark that the running time of the clique-DBA algorithm has a significant advantage over the greedy strategy algorithms, such as the greedy algorithm and the CELF algorithm. For what concerns the running time, the SD algorithm has the best performance. However, the SD algorithm cannot provide the global optimal seed nodes since it does not eliminate the overlapping problem of nodes' influence in the seed node set.

Fig. 5 Evolutionary boxplot graph of LIE values in the case of propagation probability p = 0.01 and k = 30 for six experimental social networks



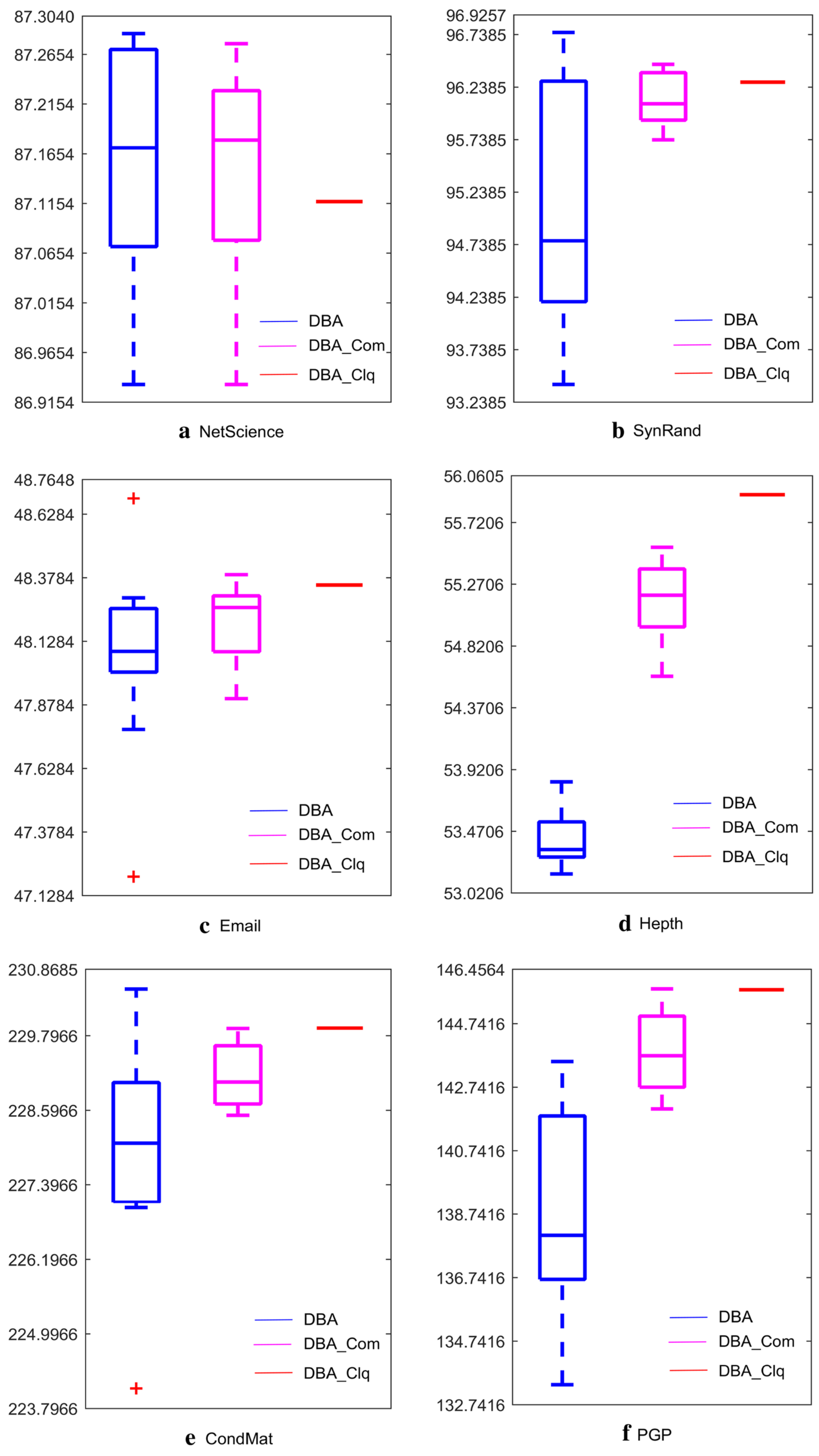


Fig. 6 Evolutionary boxplot graph of LIE values in the case of propagation probability p = 0.05 and k = 30 for six experimental social networks

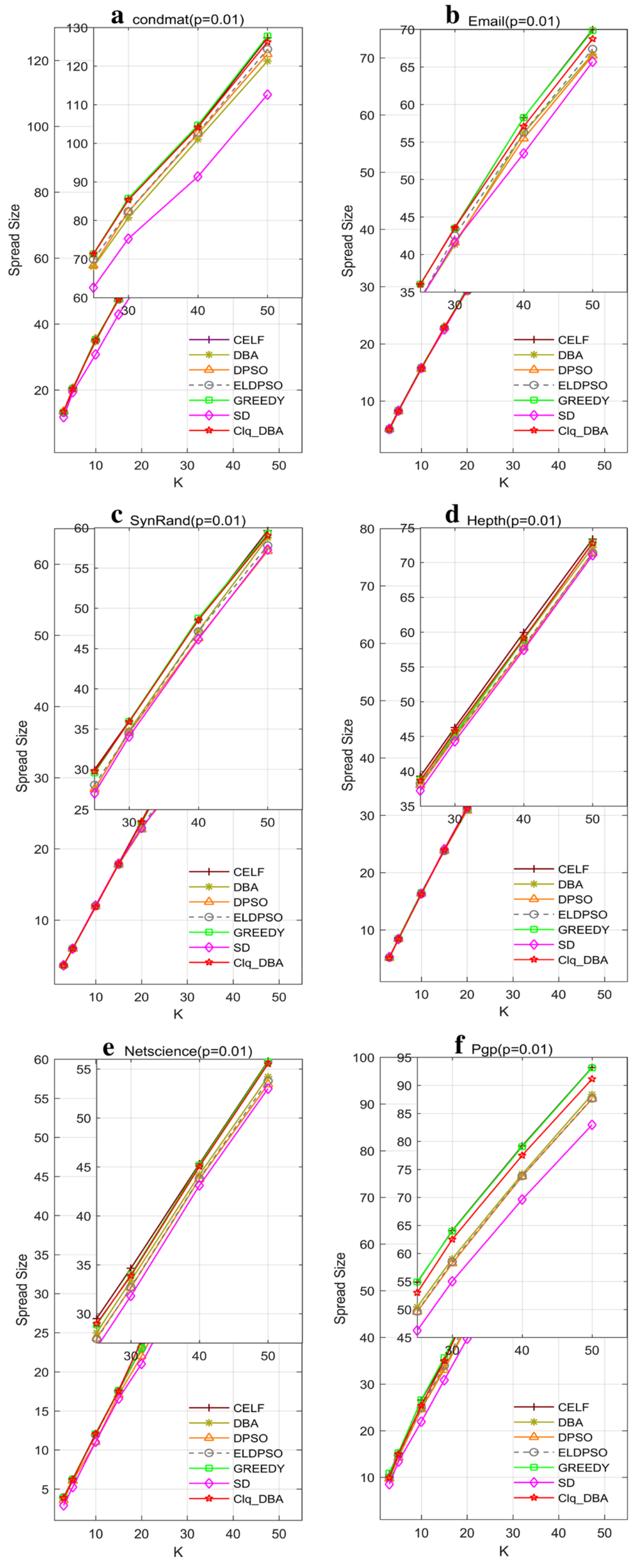


Fig. 7 Line graph of influence diffusion scale in six experimental social networks under the scenario of propagation probability $p = 0.01$ for different sizes of seed node sets

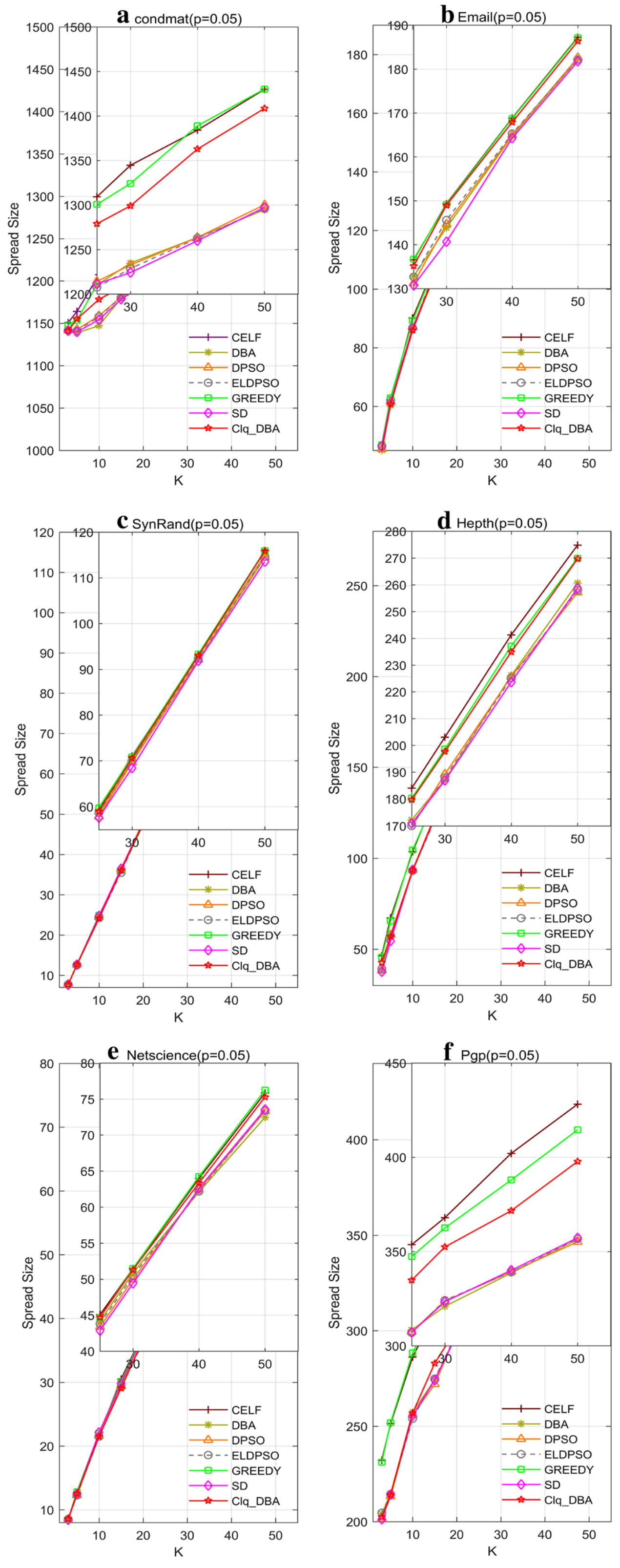


Fig. 8 Line graph of influence diffusion scale in six experimental social networks under the scenario of propagation probability $p = 0.05$ for different sizes of seed node sets

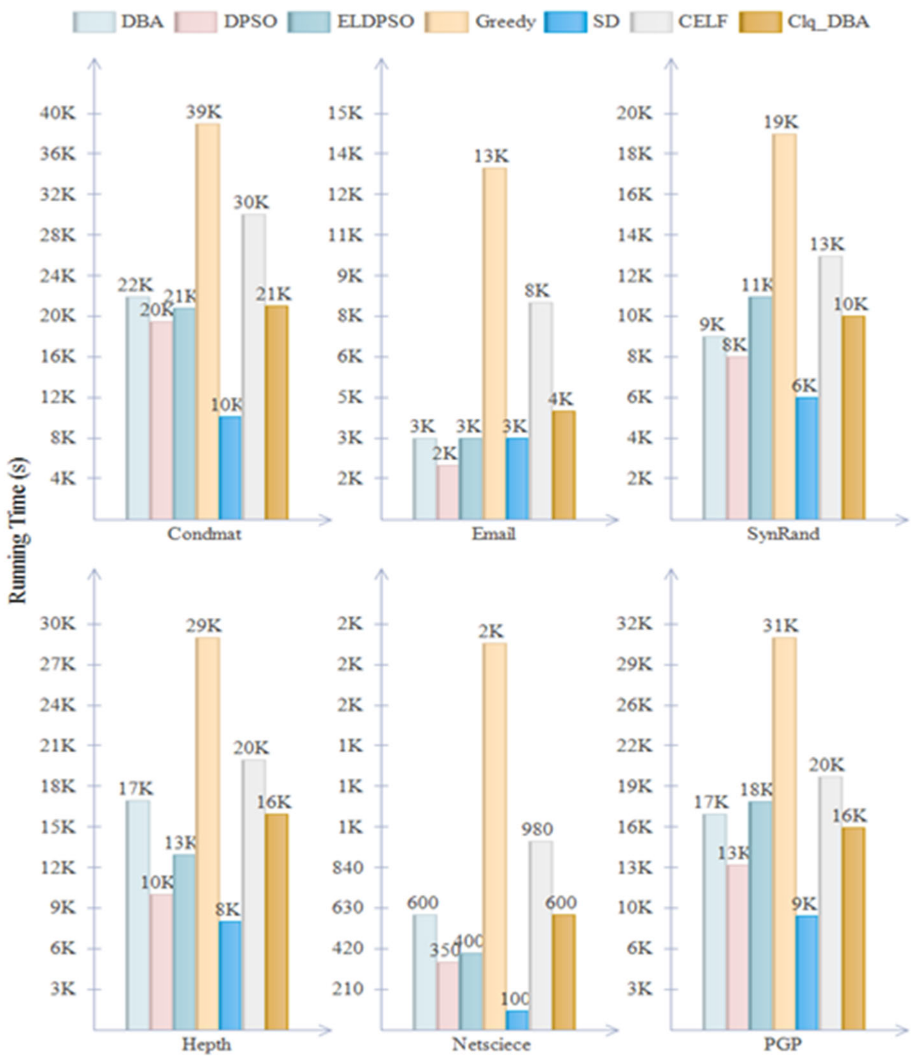


Fig. 9 Running time comparison of the seven algorithms on the six experimental social network, when the seed node set size k = 30 and the propagation probability p = 0.01

6 Conclusions and future work

With the continuous expansion of network scale, it is becoming increasingly necessary to develop more effective and robust algorithms. The experiments show that the DBA algorithm has good performance in identifying the top-k influential individuals in large-scale networks. However, this algorithm has poor stability in solving the optimal global solution.

In this paper, based on the analysis of the instability factors in the original DBA algorithm, we propose a clique-DBA algorithm. The proposed algorithm is based on the clique partition of a network to enhance the stability of the basic DBA algorithm. The experimental results show that the proposed algorithm runs several times in the two scenarios of propagation probability p = 0.01 and p = 0.05, respectively, and its LIE fitness value converges to the unique determined value. In this way, the clique-DBA algorithm eliminates the phenomenon that the LIE value of the original DBA algorithm has a large fluctuation. The simulated results under the IC model show that the clique-DBA algorithm has a performance of influence spreading scale comparable with the similar state-of-the-art algorithms.

There are several directions for future investigations. First, the application of swarm intelligence algorithms to the top-k influence node identification is minimal. Therefore, we intend to reconstruct and redesign other swarm intelligence algorithms in large-scale social networks to solve the problem of identifying top-k influential nodes. There are many swarm intelligence algorithms, and each one has its unique characteristics and advantages. Different algorithms may be more appropriate for the networks of different social relationships, such as a following, a friendship, or a partnership. Furthermore, there are more realistic and complex influence propagation models than the IC model. Therefore, it could be useful and realistic to apply a discrete swarm intelligence algorithm to top-k influence node identification under other information propagation models.

Acknowledgements This work was partially supported by the National Key R&D Program of China under Grant No. 2020YFC0832500, Ministry of Education—China Mobile Research Foundation under Grant No. MCM20170206, The Fundamental Research Funds for the Central Universities under Grant No. lzujbky-2019-kb51 and lzujbky-2018-k12, National Natural Science Foundation of China under Grant No. 61402210, Major National Project of High-Resolution Earth Observation System under Grant No. 30-Y20A34-9010-15/17, State Grid Corporation of China Science and Technology Project under Grant No. SGGSKY00WYJS2000062, Program for New Century Excellent Talents in University under Grant No. NCET-12-0250, Strategic Priority Research Program of the Chinese Academy of Sciences with Grant No. XDA03030100, Google Research Awards, and Google Faculty Award, Science and Technology Plan of Qinghai Province under Grant No. 2020-GX-164, National Social Science Fund Project under Grant No.20XTJ005, National Social Science Fund Project under Grant No.18BTJ001 and Zhejiang Provincial Natural Science Foundation under Grant No. LQ20F020011. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Jetson TX1 used for this research.

Declarations

Conflict of interest All authors declare that they have no conflicts of interest regarding the publication of this manuscript.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

AybikeŞİMŞEK KR (2018) Using swarm intelligence algorithms to detect influential individuals for influence maximization in social networks. Expert Syst Appl 114:224–236

Bi K, Han D, Zhang G, Li K-C, Castiglione A (2020) K maximum probability attack paths generation algorithm for target nodes in networked systems. Int J Inf Secur. https://doi.org/10.1007/s10207-020-00517-4

Blondel V, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of community in large networks. J Stat Mech 2008:P10008

Bonacich P, Lloyd P (2001) Eigenvector-like measures of centrality for asymmetric relations. Social Netw 23:191–201

Chakkingal PS, Kumar SKN (2016) Learning from bees: an approach for influence maximization on viral campaigns. PLoS ONE 11:e0168125

Chen W, Wang Y, Yang S Efficient influence maximization in social networks. In: Acm Sigkdd international conference on knowledge discovery and data mining, 2009. ACM, pp 199–208. https://doi.org/10.1145/1557019.1557047.

David K, Jon K, Éva T (2003) Maximizing the Spread of Influence through a Social Network. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 137–146. https://doi.org/10.1145/956750.956769

Freeman L (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41. https://doi.org/10.2307/3033543

Gong M, Yan J, Shen B, Lijia M, Cai Q (2016) Influence maximization in social networks based on discrete particle swarm optimization Information Ences An. Int J 367(368):600–614

Gregory S (2009) Finding overlapping communities using disjoint community detection algorithms. Complex networks. Springer, Berlin

Guimerà R, Danon L, Diaz-Guilera A, Giralt F (2004) Self-similar community structure in a network of human interactions. Phys Rev E Stat Nonliner Soft Matter Phys 68:065103

Hsieh M-Y, Weng T-H, Li K-C (2018) A keyword-aware recommender system using implicit feedback on Hadoop. J Parallel Distrib Comput 116:63–73. https://doi.org/10.1016/j.jpdc.2017.12.008

Jiang Q, Song G, Gao C, Yu W, Xie K (2011) Simulated annealing based influence maximization in social networks. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence., San Francisco, California, USA. AAAI, pp 7–11

Leskovec J, Kleinberg J, Faloutsos C (2007a) Graph evolution: densification and shrinking diameters. ACM Trans Knowl Discovery Data 1:2

Leskovec J, Krause A, Guestrin C, Faloutsos C (2007b) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining., San Jose, California, USA, August 12–15.

Lu F, Zhang W, Shao L, Jiang X, Xu P, Jin H (2017) Scalable influence maximization under independent cascade model. J Netw Comput Appl 86:15–23

Marek RO, Ogiela L (2017) Cognitive keys in personalized cryptography. Paper presented at the IEEE international conference on advanced information networking and applications (AINA 2017), Taipei, Taiwan

Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74:036104. https://doi.org/10.1103/PhysRevE.74.036104

Ogiela L (2020) Transformative computing in advanced data analysis processes in the cloud. Inf Process Manag 57(5):102260

Ogiela L, Marek RO (2020) Cognitive security paradigm for cloud computing applications. Concurr Comput Pract Exp 32:e5316

Ogiela L, Takizawa M (2017) Personalized cryptography in cognitive management. Soft Comput 21:2451–2464

Tang J, Zhang R (2018) Maximizing the spread of influence via the collective intelligence of discrete bat algorithm. Knowl-Based Syst 160:88–103

Tang J, Zhang R, Wang P, Zhao Z, Fan L, Liu X (2020) A discrete shuffled frog-leaping algorithm to identify influential nodes for influence maximization in social networks. Knowl Based Syst 187:104833.104831-104833.104812

Tang J, Zhang R, Yao Y, Fan Y, Zhao Z, Hu R, Yuan Y (2018) Identification of top-k influential nodes based on enhanced discrete particle swarm optimization for influence maximization. Phys A Stat Mech Appl 513:477–496

Wei L, Kuan-Ching L, Jing L, Xiaoyan K, Zomaya AY (2019) An industrial network intrusion detection algorithm based on multifeature data clustering optimization model. IEEE Trans Ind Informatics 16:2063–2071

Wei L, Yongkai F, Kuan-Ching L, Dafang Z, Jean-Luc G (2020) Secure data storage and recovery in industrial blockchain network environments. IEEE Trans Industr Inf 99:1–1

Liang W, Huang W, Long J, Li K-C, Zhang D (2020) Deep reinforcement learning for resource protection and real-time detection in IoT environment. IEEE Internet Things J 7(7):6392–6401

Yan J, Wei L, Jintian T, Kuan-Ching L (2020) A novel data representation framework based on nonnegative manifold regularisation. Connect Sci. https://doi.org/10.1080/09540091.2020.1772722

Yang X, Zhou Q, Wang J, Zhou R, Li KC (2018) An energy-efficient dynamic decision model for wireless multi-sensor network. J Supercomput 76:1585–1603

Yang XS (2012) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3:267–274

Zhang W, Han D, Li KC et al (2020) Wireless sensor network intrusion detection system based on MK-ELM[J]. Soft Comput 24(16):12361–12374. https://doi.org/10.1007/s00500-020-04678-1

Zhu G, Pan Z, Wang Q, Zhang S, Li KC (2020) Building multi-subtopic Bi-level network for micro-blog hot topic based on feature Co-Occurrence and semantic community division. J Netw Comput Appl 170:102815