Applied Intelligence (2021) 51:7749–7765
https://doi.org/10.1007/s10489-021-02283-9

Identification of top-*k* influential nodes based on discrete crow search algorithm optimization for influence maximization

Huan Li[1] · **Ruisheng Zhang**[1] · **Zhili Zhao**[1] · **Xin Liu**[1] · **Yongna Yuan**[1]

Accepted: 18 February 2021 / Published online: 16 March 2021

Abstract

Influence maximization refers to selecting a small number of influential nodes in a given network to maximize the influence affected by the subset. In social network analysis and viral marketing, influence maximization is greatly significant. The greedy-based algorithm is time-consuming in estimating the expected influence diffusion of a given node set, which is unsuitable for large-scale network. The traditional heuristics often have the problem of low accuracy. In this study, in order to solve the influence maximization problem more effectively, a meta-heuristic discrete crow search algorithm (DCSA) using the intelligence of crow population is proposed. In DCSA, a new coding mechanism and discrete evolution rules are constructed. The degree-based initialization method and the random walk strategy are adopted to enhance the search ability. Moreover, according to the network topology, influential nodes *Candidates* are generated to avoid blindness in the process of crow search. Extensive experiments are conducted on six real-world social networks under independent cascade (IC) model, the results show that DCSA outperforms other state-of-the-art algorithms and obtains comparable influence diffusion results to CELF but with lower time complexity.

Keywords Social networks · Influence maximization · Crow search algorithm · Swarm intelligence

1 Introduction

Social networks are structures that connect individuals or organizations. For example, email network, Facebook, etc. Due to large number of users and the rapid spread of information, social network has become a powerful information dissemination platform [1, 2]. A typical application driven by social networks is viral marketing [3], which uses the "word of mouth" in interpersonal relationships to influence the attitude and behavior of consumer and is proved to be a more effective way than conventional marketing tools such as mass marketing, newspapers, etc. The purpose of viral marketing is to choose a few influential users, and ultimately they can influence most people in the network.

The above problem is formulated as influence maximization (IM), and its goal is to select a small number of the most influential nodes that can trigger maximum scale of cascading. The solution to this problem has many applications, such as product marketing [4] and detection of opinion outbreaks [5].

The influence maximization problem was firstly studied from the perspective of algorithm by Domings and Richardson [6]. Kempe et al. [7] considered influence maximization in several of the most widely used models and introduced a greedy hill-climbing algorithm based on submodular function to solve the problem. However, in order to evaluate as accurately as possible the influence diffusion of a given seed node set, thousands of Monte-Carlo simulations are required, this is very time consuming. And the time complexity of the traditional greedy hill-climbing algorithm is proved to be $O(kNMR)$ [8], which severely limits it to deal with large-scale social networks.

Following the pioneering work, novel influential node selection methods have emerged to more effectively solve the influence maximization problem. Chen et al. [9] improved the traditional greedy algorithm by removing all the non propagating edges in the network. Cheng et al. [10] proposed a static greedy algorithm that strictly guarantees the submodularity of the diffusion function. In addition, a dynamic update strategy based on static snapshots was

✉ Ruisheng Zhang
zhangrs@lzu.edu.cn

[1] School of Information Science and Engineering, Lanzhou University, Lanzhou, Gansu 730000, People's Republic of China

introduced to improve the static greedy algorithm. Assuming that the influence paths are independent of each other and taking the independent influence paths as the influence evaluation unit, an algorithm based on the influence paths was proposed by Kim et al. [11]. Furthermore, a parallel path influence strategy is adopted to identify seed nodes more quickly. Ju et al. [12] introduced an algorithm using independent paths to measure the activation probability between nodes, thus avoiding the use of simulation methods to calculate influence propagation. Song et al. [13] considered the influence maximization problem on mobile social network based on community detection and proposed a divide-and-conquer method. Other methods based on community were also adopted in [14, 15]. Compared with the traditional greedy algorithm, the scalability and accuracy of community-based strategy for discovering influential nodes need to be improved. Borgs et al. [16] introduced a novel algorithm on account of reverse influence sampling (RIS) method to identify influential nodes. Although theoretical analysis shows that this method can achieve near-linear time, it suffers from huge memory consumption. Other methods intuitively select the most central nodes as most influential users [17]. The commonly used centrality methods are closeness centrality [18], betweenness centrality [19] and degree centrality [20], etc. However, the influence maximization algorithms based on the structure centrality measure often cannot provide performance guarantee [21, 22]. Mohammed Alshahrani et al. [23] introduced a new means to measure the influence of nodes via combining the local characteristics of degree centrality and the global characteristics of Katz centrality. With the continuous expansion of the social network scale, the existing algorithms often have the problems of large computation time, low accuracy, or high memory cost. Therefore, the development of efficient and effective influence maximization methods for large-scale network is still a problem worthy of further exploration in social network analysis. Meta-heuristic algorithms are widely used to deal with complex optimization problems and their effectiveness has also been proven [24, 25]. Therefore, we design a discrete crow search algorithm (DCSA) to deal with the influence maximization of social network. Our main contributions are listed below:

- The evolution rule of original meta-heuristic crow search algorithm for continuous optimization problem is modified, and a discrete crow search algorithm considering the characteristics of network structure is designed for the problem of influence maximization.
- On the basis of the contribution of each node's $k-$shell value and structural hole constraint to the network topology, the *Candidates* is generated to save the potential influence nodes for the random walk strategy designed in the discrete crow search algorithm.
- Experiments on six real-world networks illustrate the proposed DCSA is superior to other algorithms and comparable to CELF on influence spread.
- As far as we know, the discrete crow search algorithm is firstly used for influence maximization in social network.

The rest of the paper is arranged as below: Section 2 reviews and discusses related work. The preliminary information including influence maximization problem definition, typical influence independent cascade model, and influence estimate function are presented in Section 3. The proposed discrete crow search algorithm is introduced in Section 4. The simulation results and analysis are provided in Sections 5 and 6 in the conclusion.

2 Related work

Since Domings and Richardson [6] first considered the influence maximization as an algorithm problem, a number of methods have been proposed to select seed nodes for maximizing their influence spread. The existing influence maximization algorithms are mainly classified into four groups, which are discussed as followings.

2.1 Greedy based algorithms

Kempe et al. [7] considered influence maximization in several of the most widely used models, namely leaner threshold (LT) model, independent cascade (IC) model and weighted cascade (WC) model, and proved that the optimization problem of choosing the most influential nodes is NP-hard under these models. They proposed a hill-climbing greedy algorithm based on submodular function to approximate the optimal solution within a factor of $(1 - 1/e - \epsilon)$, where e is the natural logarithm and ϵ is the error caused by Monte-Carlo simulation. However, in order to accurately assess the influence propagation of a given set of nodes, thousands of Monte-Carlo simulations are required, which is proved to be ♯P-hard problem. Each time when the next seed node is selected, the greedy algorithm evaluates all nodes one by one in the entire network space, resulting in time-consumption. These problems severely limit greedy algorithm to deal with large-scale social networks. Therefore, in order to optimize the running time of the original greedy algorithm, some researches have been carried out. By exploiting the submodularity property, Leskovex et al. [8] used a priority queue to store the nodes with the largest marginal gain and proposed a Cost-Effective

Lazy-Forward algorithm (called CELF in short). Experimental results show that CELF algorithm can achieve the solution accuracy close to the traditional greedy algorithm, and can effectively reduce the Monte-Carlo simulation times when calculating the marginal gain of nodes. CELF++ strategy [26] was proposed to reduce the time complexity using similar idea. Experimental results show that it is 35%-55% faster than CELF. In order to improve the efficiency of greedy algorithm, Heidari et al. [27] designed a state machine greedy algorithm, which counts the traversing nodes during the estimation propagation process, and constructs the Monte-Carlo graph in the simulation. Lu et al. [28] designed a recursive equation to estimate the influence spread recursively by using the reachability probability between nodes. And based on the greedy strategy, the node with the largest estimate is selected recursively as the seed node.

Compared with the traditional greedy algorithm, these algorithms have some improvement in the running time, but they still have the time-consuming shortcoming in dealing with large-scale social networks and even sacrifice accuracy.

2.2 Heuristic algorithms

In order to improve the time efficiency of the optimization process of influence maximization, a series of heuristic methods have been proposed. A simple method is to select the first k nodes as the most influential individuals according to a given centrality index, for example, degree centrality [20]. However, previous studies [7, 9] show that simply selecting the top k nodes with the highest index as seeds may produce inaccurate results, because these nodes are often clustered together, leading to overlapping influence diffusion. The SingleDiscount algorithm [9] considered the influence of the selected node on the current candidate. Once a node is selected as seed node, the degree values of its neighbors are all reduced by 1. Chen et al. [9] considered the IC model with a small propagation probability p and designed a new method called DegreeDiscount. They gave a more accurate discount value for the neighbors of nodes which are chosen as seeds. Both SingleDiscount and DegreeDiscount simply assumed that nodes with more neighbors tend to have higher influence and will therefore be selected as seed nodes. Inspired by Pagerank [29] heuristic scheme, Zhang et al. [30] proposed PRDiscount algorithm. They take into account neighbors' influential power when measuring one node's influential power and discount the influence power of those nodes which have relationships with chosen seeds to avoid influence overlapping problem. Like DegreeDicount method, Wang et al. [31] carried out a strategy to punish the influence of the neighbors within two-hop of nodes that have already been selected as spreaders to avoid the overlapping phenomenon in influence diffusion.

2.3 Reverse influence sampling based algorithms

Borgs et al. [16] designed a reverse influence sampling (RIS) method for influence maximization, which can obtain the near-optimal approximation factor of $(1-1/e-\epsilon)$ with probability of 3/5 in time $O((m+n)\epsilon^{-3}\log n)$.The basic idea of RIS is that if a node often appears in the generated subgraph, it is likely to be the most influential node. On the basis of RIS method, researchers have done many improvements. By including more adjustable parameters to link the theory and practice of influence maximization, TIM and TIM+ algorithm were developed by Tang et al. [32]. Compared with RIS algorithm, TIM has lower time complexity. Then Tang et al. [33] futher put forward IMM algorithm based on martingales. IMM keeps the superior approximate guarantee and lower time complexity of TIM and TIM+, and overcomes their shortcomings. Nguyen et al. [34] proposed two new algorithmic frameworks SSA and D-SSA based on RIS. Their computational efficiency is 1200 times faster than IMM, and they can achieve the same approximation guarantee of $(1-1/e-\epsilon)$. A greedy Sketch-based algorithm, SKIM, was developed by Cohen et al. [35] through building combined reachability sketches. SKIM can be extended to networks with billions of edges and achieve same approximation guarantee compared with TIM+. But SKIM needs huge amounts of memory to generate subgraphs. Other sketch based RIS frameworks, BKRIS [36] and RSRS [37], were developed to deal with the limitations of TIM+ and SKIM. Compared with basic greedy algorithm, the reverse influence sampling based algorithms have lower time complexity. However, with the increase of seed set size, these methods often get suboptimal solutions. At the same time, in order to save the generated subgraphs, they suffer from a lot of memory overload.

2.4 Meta-heuristic algorithms

Influence maximization is discrete combination optimization problem and evolutionary optimization algorithms can be applied to solve it by defining a fitness function. Jiang et al. [38] put forward Expected Diffusion Value(EDV), as shown in (1), where $N_S{}^{(1)}$ represents the one-hop area of candidate set S, $t(v)$ represents the number of edges of node v in $N_S{}^{(1)}$.

$$EDV(S) = k + \sum_{v \in N_S{}^{(1)} \setminus S} (1-(1-p)^{r(v)}) \tag{1}$$

The simulated annealing algorithm was adopted to optimize and find seed set S with the aim of maximizing EDV. Experiments exhibit the algorithm can maintain high accuracy, and it is 2$\sim$3 orders of magnitude faster than greedy algorithm. Gong et al. [39] introduced a fast influence estimation function to measure the influence of seed set on two-hop

neighbors, and used Discrete Particle Swarm Optimization (DPSO) to solve the influence maximization problem. In addition, Gong et al. [40] also introduced a memetic algorithm to optimize the 2-hop influence spread to find the most influential nodes. A degree-descending search evolution (DDSE) algorithm was proposed by Cui et al. [41]. The algorithm uses function *EDV* to avoid repeated simulation, thus overcoming the efficiency problem of greedy algorithm. Experimental results show that DDSE is more efficiency compared with the state-of-the-art greedy algorithm. Zareie et al. [42] applied the concept of entropy to define a fitness function and then solved the influence maximization by using gray wolf optimization algorithm. In addition, Tang et al. [43] introduced discrete bat optimization algorithm. The advantage of meta-heuristic evolutionary algorithm is that it can avoid using Monte-Carlo simulation to evaluate a given node set and reduce the time complexity. But the reasonable discrete evolution mechanism is very important to the solution quality and efficient of the algorithm. Therefore, the development of efficient and effective influence maximization algorithm is still a problem worthy of further exploration.

3 Preliminary information

Let $G(V, E)$ be a graph, where $V(|V| = n)$ and $E(|E| = m)$ denote the node set and the edge set in G respectively. The node is the individual in the network, and the edge denotes the relationship between two individuals.

3.1 Influence maximization

Definition 1 For a given social network G and a positive integer $k\,(k \leq n)$, the influence maximization problem refers to choose a group of most influential nodes with the size of k from network as seed set S. Under a given influence propagation model, the influence spread of the seed set S, denoted as $\sigma(S)$, reaches the maximum at the end of propagation process.

$$S^* = \underset{S \subseteq V,\, |S|=k}{\arg\max}\ \sigma(S) \tag{2}$$

As an optimization problem, influence maximization has been proven to be NP-hard by Kempe et al. [7]. It is ♯P-hard to accurately estimate the influence of a given seed set S [44].

3.2 Diffusion model

There are three commonly used models to simulate the process of influence propagation, which are the independent cascade (IC) model, the linear threshold (LT) model and the weighted cascade (WC) model [7]. In this paper, the IC model is chosen as the simulation mechanism. In IC model, nodes are defined as active or inactive. The active node is its corresponding individual who has accepted new products or new ideas. Otherwise, it is an inactive node. Nodes can only switch from inactive state to active state, but not vice versa. Each edge is assigned an active probability p that characterizes the possibility that an inactive node can be activated by its active neighbors.

The specific process of IC model can be described as follows: (1) Given an initial set of active nodes S, at time t, the active node u attempts to activate the inactive neighbor v with probability p. No matter whether it succeeds or not, the active node u no longer has the ability to activate other nodes after time t; (2) If node v has multiple active neighbors, then these nodes can try to activate node v in any order. If node u successfully activates node v, then node v will be converted into active node at time $t + 1$; (3) At time $t+1$, the active node v tries to activate the inactive neighbors with a certain probability. The process is repeated until no active nodes are generated at time T.

3.3 Influence estimate function

The calculation of the influence diffusion $\sigma(S)$ of a given node set has been proven to be ♯P-hard. The Monte-Carlo simulation method usually runs at least tens of thousands of times in order to get more accurate estimation of influence spread, which is very time-consuming. Therefore, it is important to construct an effective mechanism to accurately estimate the expected influence diffusion of a given node set. As pointed in [45], in social networks, there is an intrinsic-decay in the spread of personal opinions, and one's influence tends to be limited to a small number of friends in a local area. Furthermore, Pei et al. [46] pointed that the sum of nearest neighbors is a reliable local proxy for user influence. When there is no complete global network structure, the expected influence diffusion of nodes in the network depends on the influence of the two-hop area. Gong et al. [39] developed a local influence estimation (*LIE*) as shown in (3) to estimate the influence within the two-hop area of a given node set.

$$\begin{aligned} LIE(S) &= \sigma_0(S) + \sigma_1^*(S) + \tilde{\sigma}_2(S) \\ &= k + \sigma_1^*(S) + \frac{\sigma_1^*(S)}{|N_S^{(1)} \backslash S|} \sum_{u \in N_S^{(2)} \backslash S} p_u^* d_u^* \\ &= k + \left(1 + \frac{1}{|N_S^{(1)} \backslash S|} \sum_{u \in N_S^{(2)} \backslash S} p_u^* d_u^*\right) \\ &\quad \sum_{i \in N_S^{(1)} \backslash S} \left(1 - \prod_{(i,j) \in E,\, j \in S} (1 - p_{i,j})\right) \end{aligned} \tag{3}$$

where $\sigma_0(S)$ represents the initial seed set S, $\sigma_1^*(S)$ and $\tilde{\sigma}_2(S)$ represent the expected influence spread of one-hop area and two-hop area for a node set S, respectively. $N_S^{(1)}$ and $N_S^{(2)}$ are the one-hop and two-hop area of seed set S. p_u^* is the constant active probability of a propogation model. d_u^* is the number of edges of node u within $N_S^{(1)}$ and $N_S^{(2)}$. By maximizing the fitness value of (3), the optimal seed set can be selected. The problem of influence maximization is changed into the problem of optimizing the *LIE* function.

3.4 Crow search algorithm

Crows are widely distributed and regarded as one of the most intelligent animals in the world. By idealizing the intelligent behavior of crows, Alireza Askarzadeh [47] proposed a population-based technique, crow search algorithm (CSA), which works on the principle that crows store excess food in a hidden place and retrieve it when needed. The ideology of CSA can be described as follows.

The number of crows (flock size) is n and the location of crow i at time (iteration) $iter$ in the d-dimensional search space is represented by a vector $x^{i,iter}(i = 1, 2, 3, ..., n; iter = 1, 2, ..., iter_{max})$ where $iter_{max}$ is the maximum number of iterations. Every crow has a memory, and the location of its hiding place (food source) is remembered. At iteration $iter$, the location of core i's hiding place is $m^{i,iter}$. This is also the best location that corw i has got so far. Crows move in space, looking for better food sources. Suppose in iterative $iter$, crow j wants to access its hidden position $m^{j,iter}$. Crow i decides to follow crow j and approach crow j's hiding place. If crow j doesn't know that crow i is tracking it. Crow i will be close to crow j's hiding place. In this case, the new location of crow i is updated as follows:

$$x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - x^{i,iter}) \quad (4)$$

Where r_i is a random number uniformly distributed between 0 and 1. $fl^{i,iter}$ represents the flight length of crow i at iteration $iter$. $m^{j,iter}$ is the memory location of crow j. If crow j knows that crow i is tracking it. In order to protect its food source from being stolen, crow j will fool crow i to another position in the search space. In this case, the position of crow i will be updated randomly.

Totally, these two cases can be expressed as follows:

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - x^{i,iter}), & r_j \geq AP^{j,iter} \\ a\ random\ position, & otherwise \end{cases} \quad (5)$$

Where r_j is a random number uniformly distributed between 0 and 1. $AP^{j,iter}$ is the awareness probability of crow j at iteration $iter$. The specific process of crow search algorithm can be expressed as pseudo code displayed in the Algorithm 1 [47].

Algorithm 1 Corw search algorithm.

1: Randomly initialize the position of a flock of n crows in the search space
2: Evaluate the position of the crows
3: Initialize the memory of each crow
4: **while** $iter < iter_{max}$ **do**
5: **for** $i = 1 : N$ **do**
6: Randomly choose one of the crows to follow (for example j)
7: Define an awareness probability
8: **if** $r_j \geq AP^{j,iter}$ **then**
9: $x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - x^{i,iter})$
10: **else**
11: $x^{i,iter+1} =$ a random position of search space
12: **end if**
13: **end for**
14: Check the feasibility of new positions
15: Evaluate the new position of the crows
16: Update the memory of crows
17: **end while**

The pseudo code of original CSA is presented in Algorithm 1. Firstly, the position and momery position of crows are initialized randomly in the search space (lines 1-3). During the iteration process, the new position of each crow is updated by two strategies (line 9 and line 11). Then, the memory position of crows are updated (line 16). These iteration processes will continue until the maximum number of iterations is reached. Since CSA was proposed, it has been applied in optimization [48–52], prediction [53, 54], feature selection [55], classification [56, 57] and so on [58–61].

4 Proposed method

The purpose of influence maximization is to find a group of most influential nodes, which is an optimization problem. As mentioned above, according to the local influence estimation function, the expected influence diffusion of a given set of nodes can be estimated. The goal of influence maximization is transformed into selecting top-k influential nodes by optimizing LIE function. Algorithms based on swarm intelligence have an efficient evolution mechanism. Therefore, we conducted an in-depth analysis of the swarm intelligence algorithm and designed a discrete crow search algorithm (DCSA) for influence maximization. The solution space of influence maximization problem is discrete, so

the original crow search algorithm for continuous problems cannot be directly applied. Therefore, we redefined the solution vector as discrete form. In addition, we propose a new evolutionary rules for discrete crow population. The degree-based initilization method and the random walk strategy are adopted to enhance the search ability of crows. In the following section, we give the discrete encoding mechanism, discrete evolutionary rules and the overall framework and details of DCSA for influence maximization.

4.1 Discrete crow search algorithm

4.1.1 Discrete encoding mechanism

The original CSA was developed to solve the continuous combinational optimization problem [47]. However, the solution space of the influence maximization problem is discrete. According to the characteristics of network topology, we redesigned a coding mechanism for crow individuals. Each individual in crow population is expressed by k potential candidate seed nodes, i.e., each crow is a seed set of k-size. The position of crow i can be encoded as a k-dimensional integer vector $X_i = (x_{i1}, x_{i2}, ..., x_{ik})$ (where $i = 1, 2, ..., n$). Each element $x_{ij} (j = 1, 2, ..., k)$ of X_i is a node in the network G. For example, given the size of seed set $k = 5$, a crow individual $X_i = (3, 15, 6, 12, 28)$ means that 5 nodes, 3, 15, 6, 12, 28, are selected from the network as the most influential candidate nodes. On the basis of the update mechanism of the discrete crow search algorithm, each individual continues being updated before interation is terminated, and then the optimal crow is regarded as the target seed set. Therefore, the discrete crow search algorithm can solve the influence maximization problem.

4.1.2 Update rules

In order to complete the evolution of crows in the discrete network space and find the optimal location. The update rules are redesigned in a discrete form for discrete crow search algorithm. The newly evolutionary rules for DCSA are defined as:

$$x^{i,iter+1} \leftarrow x^{i,iter} \oplus H(r_i \times fl^{i,iter} \times (x^{i,iter} \cap m^{j,iter})) \quad (6)$$

Where r_i is a random number uniformly distributed between 0 and 1. $x^{i,iter}$ and $x^{i,iter+1}$ are the position of crow x_i in interation $iter$ and $iter + 1$ respectively. $fl^{i,iter}$ is the flight lenth of crow i. $m^{j,iter}$ is the memory location of crow j. The operator "$\cap$" in the formula is a logical operator defined as a similar intersection operation, which is used to determine whether there is a common element between $x^{i,iter}$ and $m^{j,iter}$. If in $m^{j,iter}$ there is a node that is same as an element in $x^{i,iter}$, the element at the corresponding position of the result is set to 0, otherwise it is set to 1. For example, given the number of initial set $k = 5$, if $x^{i,iter} = (3, 15, 6, 12, 28)$ and $m^{j,iter} = (4, 10, 3, 6, 28)$ represent the corw i's position vector and corw j's hiding place in interation $iter$, respectively, then the corresponding result of $x^{i,iter} \cap m^{j,iter}$ can be computed as $(0, 1, 0, 1, 0)$.

$H(\cdot)$ in (6) is a decesion function. Given that X_i as the argument, then the function $H(X_i)$ is expressed as $H(X_i) = (h_1(x_{i1}), h_2(x_{i2}), ..., h_k(x_{ik}))$, where $h_j(x_{ij})(j = 1, 2, ..., k)$ is a threshold factor:

$$h_j(x_{ij}) = \begin{cases} 0, & if\ x_{ij} < 1 \\ 1, & if\ x_{ij} \geq 1 \end{cases} \quad (7)$$

The operator "$\oplus$" in (6) is used to determine whether element x_{ij} in X_i should be retained or replaced. And the update rules of the element in $x^{i,iter+1}$ can be formulated in (8)

$$x_{ij,iter+1} = \begin{cases} x_{ij,iter}, & if\ h_j(x_{ij}) = 0 \\ Replace(x_{ij,iter}, V), & if\ h_j(x_{ij}) = 1 \end{cases} \quad (8)$$

$H(\cdot)$ in the right of operator "$\oplus$" returns a vector of 0 and 1. If $h_j(x_{ij}) = 0$, x_{ij} at the corresponding position is retained. Otherwise, the function $Replace(\cdot)$ will be executed. $Replace(\cdot)$ in (8) is a function to replace the element $x_{ij,iter}$ with a random node from set V of network, and it can ensure that there are no duplicate nodes in x_i after the replacement is completed.

4.2 Framework of DCSA

Influence maximization is a typical optimization problem. An effective and efficient evaluation model is essential to identify influential nodes. Therefore, based on the *LIE* function, we propose DCSA algorithm to find the most influential k nodes in the discrete network search space. The degree-based method is applied to initialize. And random walk strategy is devised to improve search ability. The local search strategy is used to speed up the convergence. Algorithm 2 shows the framework of the proposed DCSA.

In the framework, the position vectors of corw population are initialized by a high degree centrality based algorithm (lines 1 and 2 in Algorithm 2), which are detailed in Section 4.2.1. And the $LIE(\cdot)$ function (3) is for calculating the expected local influence spread of seed set. The best position vector is determined by the LIE function (line 4). For every crow x_i in population x, the new position of x_i is updated by a randomly selected crow j (lines 7-11). If a random number r is larger than the awareness probability of j, the new position of x_i will be updated according to

(6) (lines 7-10). Otherwise, the random walk strategy will be applied (lines 12-14). Furthermore, in the random walk strategy, we generate a set of *Candidates* seed nodes for the discrete crow algorithm (line 12) to improve the accuracy and convergence speed. And the details are presented in Section 4.2.2. Finally, to further improve the convergence speed, a local search strategy is adopted (lines 20-21), and the detailed information are discussed in Section 4.2.3.

Algorithm 2 Framework of DCSA.

Input: Graph $G = (V, E)$, corw population size n, seed set size k, maximum number of iterations g_{max}, flight lenth fl and awareness propability AP.

1: Initialize position vector $x \leftarrow$ Initialization(G, n, k)
2: Initialize memory vector $m \leftarrow$ Initialization(G, n, k)
3: Initialize vector r with number drawn randomly from (0,1)
4: Choose the best location vector x_* according to the LIE of each x_i
5: Initialize iterator $iter = 0$
6: **while** $iter < g_{max}$ **do**
7: **for** each $x_i \in x$ **do**
8: $j \leftarrow Random(n, index)$
9: **if** $r_j \geq AP^j$ **then**
10: Update the vector x_i according to (6)
11: **else**
12: $x_i' \leftarrow RandomWalk(x_i, Candidates, k)$
13: **if** $LIE(x_i') > LIE(x_i)$ **then**
14: $x_i \leftarrow x_i'$
15: **end if**
16: **end if**
17: **end for**
18: Update the memory of crows
19: Select out the best position vector $x_\star$ in the current iteration
20: $x_*' \leftarrow Localsearch(x_*, G)$
21: $x_* \leftarrow Max(x_*', x_*)$
22: $iter \leftarrow iter + 1$
23: **end while**

Output: The best position vector x_*

4.2.1 Initialization

We adopt a high degree based heuristic method to initialize the crows' position vectors, which is similar to that of DPSO [39]. Each crow in the population is initialized by a highest degree metric, which selects the k nodes in network G. At the same time, we use a turbulence mechanism to randomly increase the diversity of the initial crow population. The details of initialization process is given in Algorithm 3.

Algorithm 3 $Initialization(G, n, k)$.

Input: Graph $G = (V, E)$, corw population size n and seed set size k.

1: **for** each $i \leq n$ **do**
2: $X_i \leftarrow Degree(G, k)$
3: **for** each $x_{ij} \in X_i$ **do**
4: **if** $rand() > 0.5$ **then**
5: $x_{ij} \leftarrow Replace(x_{ij}, V)$
6: **end if**
7: **end for**
8: **end for**

Output: $The\ initial\ position\ vector\ X$.

The nodes with higher degree in graph G are selected to initialize the algorithm (the function in line 2). The function $Replace(x_{ij}, V)$ in line 5 replaces x_{ij} with nodes randomly selected from V of graph G and ensures that there are no duplicate nodes in the initial position vector X_i after replacement.

4.2.2 Random walk strategy

In original CSA, once crow j knows that crow i is tracking it, crow j will search for a random location in the space to fool crow i. It is recommended to use a random walk strategy to generate a new location for the crow. However, randomly generating new positions for crows in the search space will slow down convergence speed. In order to avoid the crow exploiting bindly, we introduce a random walk strategy to create new positions for crows. In addition, a group of candidate is produced by the proposed node contribution index. The node contribution index can be expressed as follows:

$$NC_i = \alpha \frac{KS_i}{\sum_{i=i}^{N} KS_i} + (1-\alpha)\frac{1/SH_i}{\sum_{i=1}^{N}(1/SH_i)} \tag{9}$$

where NC_i is the node contribution value of i, N is the number of nodes in the graph, KS_i represents the k-shell [62] value of node i, and SH_i represents the value of structural hole [63] of node i, we use the constraint on all nodes in network. The k-shell value reflects the location of node. The index of network constraint is used to quantify the constraints of nodes when they form structural holes, which can be used to reflect the pivotal role of a node. The parameter α is a constant coefficient. We arrange the nodes in descending order according to the contribution index value, and select the top $\beta \cdot k$ nodes to form the *Candidates*. The pseudocode of this strategy is expressed in Algorithm 4.

Algorithm 4 $RandomWalk(x_i, Candidates, k)$.

Input: *Candidates*, seed set size k and constant coefficient α.

$j \leftarrow 1, x'_i \leftarrow \emptyset$
while $j \leq k$ **do**
  $temp \leftarrow$ a random node select from *Candidates*
  if $temp \notin x'_i$ **then**
    $x'_{ij} \leftarrow temp$
  end if
  $j \leftarrow j + 1$
end while

Output: $The\ new\ position\ vector\ x'_i$

4.2.3 Local search strategy

In order to accelerate the convergence speed of DCSA, we employ a greedy-based local search strategy. The goal is to seek a better one near the best solution at present. The pseudocode is described in Algorithm 5.

Algorithm 5 $Localsearch(x_i, G)$.

Input: Graph $G = (V, E)$, crow position vector x_i.

1: $d_{x_i} \leftarrow Degree(x_i)$
2: $x'_i \leftarrow Order(x_i, d_{x_i})$
3: **for** each element $x'_{ij} \in x'_i$ **do**
4: **for** each node $Neighbors$ in $Neighbor^{(1)}_{x'_{ij}}$ **do**
5: $x'_i \leftarrow Replace(x'_{ij}, Neighbors)$
6: **if** $LIE(x'_i) > LIE(x_i)$ **then**
7: $x_i \leftarrow x'_i$
8: **end if**
9: **end for**
10: **end for**

Output: $The\ new\ position\ vector\ x_i$

Firstly, k nodes in the selected crow position vector are arranged in ascending order by degree centrality (lines 1 and 2 in Algorithm 5), so that the nodes with less degree are updated to their neighbors with greater influence. Secondly, the local search strategy is applied to the crow position, and the current node is replaced by the node with more influence in the one-hop set of neighbors to get the new local optimal crow position (line 8 in Algorithm 5), which is used for the next generation evolution until the termination condition is met.

4.3 Computational complexity of DCSA

The computational complexity of DCSA is analyzed according the aforementioned process. We use k, n, $\bar{D}$ and g_{max} to denote the size of node set, population size, average node degree and the number of iterations, respectively. The complexity of high degree centrality based initialization is $O(n \cdot k)$. The complexity of updating the position vector is $O(k \cdot \log k \cdot n)$. The random walk strategy needs $O(k)$ and local search strategy needs $O(k(\log k + \bar{D}))$. In addition, calculating the fitness function LIE needs $O(k \cdot \bar{D})$. In the worst case, the computational complexity of DCSA is $O(n \cdot k \cdot (\log k + \bar{D}) \cdot g_{max})$.

5 Experimental results

In this section, in order to validate the effectiveness and efficiency of DCSA, experiments are conducted on social networks. Firstly, the experimental networks and the benchmark algorithms of comparison are introduced. After that, experiments are conducted to determine the optimal parameters of DCSA algorithm. And DCSA is compared with the variant version without *Candiates* in Random walk strategy called DCSA-nc to verify the effectiveness of the strategy. Finally, the experimental results of all algorithms on the propagation model are compared and discussed.

5.1 Experimental networks and baseline algorithms

Six real networks are used to test DCSA in dealing with the problem of influence maximization. The statistical characteristics of these networks are shown in Table 1.

Among the six networks, the four networks of HepTh, NetHEHT, AstroPh and CaAstroPh are cooperative networks of scientific authors from the Astrophysics section of arXiv. The node in graph represents the author and the edge indicates that the two authors have a co-authored publication. CondMat is the collaborative network of authors of scientific papers on Condensed Matter. These five networks are downloaded from arXiv.[1] Brightkite is a network containing the friendship between users and users on Brightkite. The node and edge represent the user and their friendship respectively. This network is downloaded from KONECT[2]. In order to better understand the structure of the graphs, we present the node degree distribution of the six networks, as shown in Fig. 1.

Seven state-of-the-art influence maximization algorithms are compared with our algorithm DCSA on the IC model with a propagation probability of $p = 0.01$.

SSA (Stop-and-Stare Algorithm) [34] is a influence maximization algorithm based on reverse influence sampling.

[1] http://www.arxiv.org/
[2] http://konect.uni-koblenz.de/networks/

Table 1 Statistical characteristics of six real-world networks

Networks	$\|N\|$	$\|E\|$	k_{max}	$\langle k \rangle$	$\bar{d}$	C
HepTh	8361	15751	50	3.767	7.025	0.486
NetHEHT	15229	31376	64	4.121	5.840	0.499
AstroPh	16706	121251	360	14.516	4.798	0.665
CaAstroPh	18772	198050	504	21.101	4.194	0.631
CondMat	40421	175692	278	8.693	5.499	0.650
Brightkite	58229	214078	1134	7.353	4.917	0.172

$|N|$ and $|E|$ are the number of nodes and edges, respectively. k_{max} represents the maximum degree, $\langle k \rangle$ represents the average degree, $\bar{d}$ represents the average shortest path length, C is the average clustering coefficient

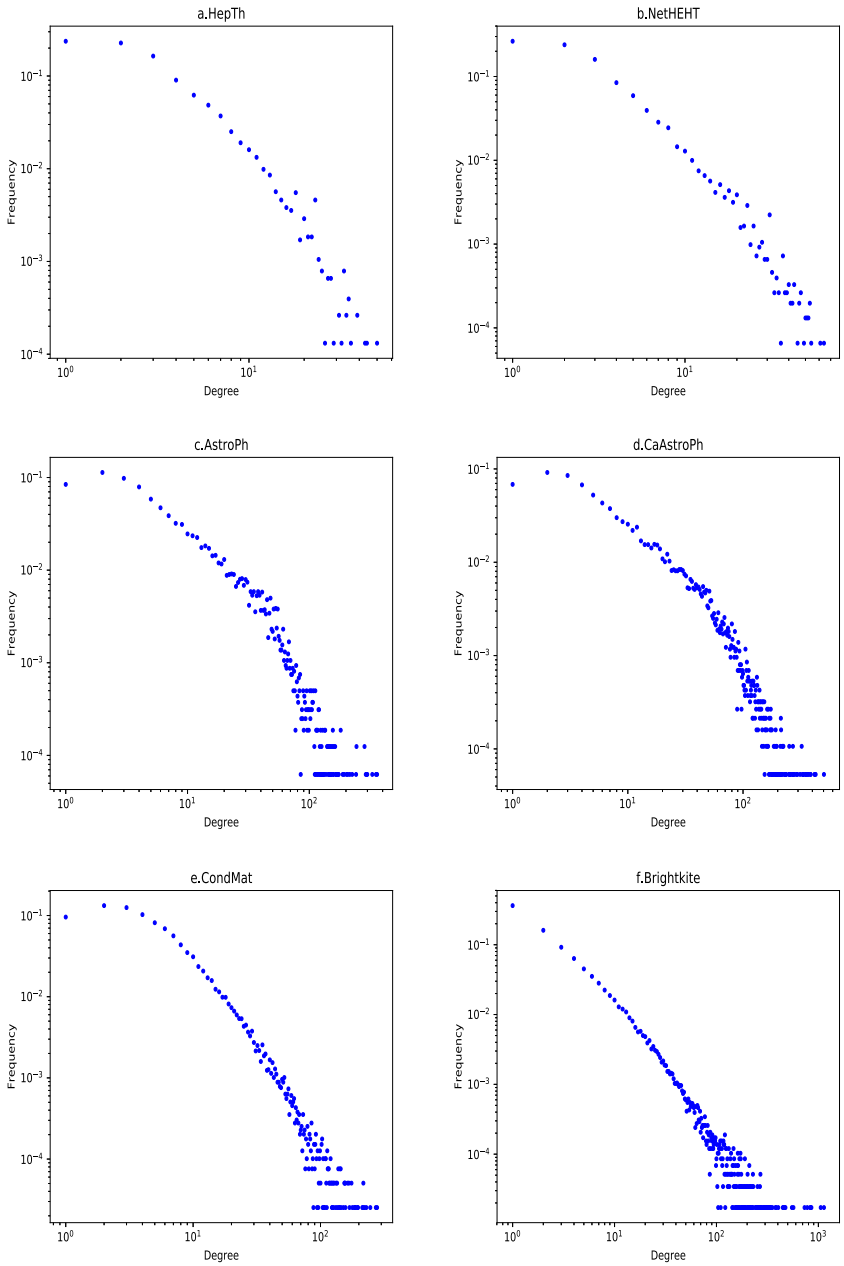


Fig. 1 Node degree distribution of the six real-world networks

The algorithm selects the k nodes which appear most frequently in the generated subgraph as seed nodes. As suggested in SSA, the parameter ϵ and δ are set to 0.1 and 0.01 respectively.

DBA (discrete bat algorithm) [43] is a meta-heuristic algorithm based on discrete bat algorithm. The local influence function is used as the funtion to optimize. The maximal iterative generation is set to 100, the parameters α and γ in the algorithm are all set to 0.3 as suggested in DBA.

LAIM (Linear Time Iterative Approach for Influence Maximization) [64] is an iterative algorithm with a recursive equation that mines the top k nodes with the greatest local influence spread as seeds.

DDSE (Degree-Descending Search Evolution) [41] algorithm uses the EDV to estimate the expected influence of nodes. A discrete differential evolution algorithm is applied to optimize the EDV to find seed nodes. The maximal iterative generation is set to 100.

IMPC (Influence Maximization based on multi-neighbor Potential in Community networks) [14] divides the influence diffusion process into two phases. The algorithm first expands the seed node to its potential neighbors. Then, node influence spreads in its inter-community.

DPSO (Discrete particle swarm optimization) [39] algorithm is a meta-heuristic algorithm. The discrete particle swarm optimization algorithm is used to optimize the LIE for selecting seeds. The maximal iterative generation is set to 100, the learning factors c_1 and c_2 are set to 2 and the inertia wight w is set to 0.8.

CELF (Cost-Effective Lazy Forward) [8] is a classical greedy strategy based algorithm, which can achieve the same accuracy as the traditional greedy algorithm. In order to obtain accurate influence estimation, the Monte-Carlo simulation times are set to 10000.

5.2 Experiments for the parameters of DCSA

First, we carry out experiments for the parameters of DCSA algorithm and detemine the best value of these parameters. In the proposed DCSA algorithm, the size of node set is denoted as k, the size of population is n, the maximal iterative generations is g_{max}, the awareness probability is AP and the flight length is fl. We assign the seed set size k the value of 50. The parameters α and β are set to 0.5 and 3 respectively of the random stratsgy in DCSA.

5.2.1 Awareness probability and flight length

In DCSA, intensification and diversification are mainly controlled by awareness probability AP. By reducing the value of awareness probability, DCSA tends to search in the local area. Besides, as the value of awareness probability increases, DCSA tries to explore the search space globally. The awareness probability is important for the convergence of evolution. LIE optimization experiments are carried out to select the appropriate AP value. The other parameters are seted as follws: $n = 30$, $g_{max} = 100$, $fl = 2$. Figure 2a presents results of distinct values of AP. In Fig. 2a, when AP changes from 0.1 to 1 with interval 0.1, DCSA obtains different fitness values of LIE, and the fitness value reach the top at the awareness probability $AP = 0.1$. Therefore, AP is set to 0.1 in this paper.

The parameter fl is the flight length of crow. A small value of fl results in a local search, and a large value results in a global search. In order to determine the optimal fl value, we set $n = 30$, $g_{max} = 100$ and $AP = 0.1$ for experiments. The value of fl ranges from 0.5, 1, 1.5, 2, 2.5 to 3. The LIE value results obtained by different values of fl on six networks are shown in the Fig. 2b. From the figure, we can observe that the effect of fl on performance is not very obvious. However, when fl is 2, DCSA performs relatively powerfully on most datasets. Therefore, in the following experiments, fl is set to 2.

5.2.2 The size of population and maximal number of iterations

In order to determine the optimal parameters of the population size n and the maximal number of iterations g_{max} of DCSA, we set awareness probability $AP = 0.1$, flight length $fl = 2$ to conduct a series of experiments. Figure 3 shows the LIE fitness values of DCSA with different n and g_{max}, and the x-axis represents six real-world social networks and the y-axis represents LIE fitness value.

Figure 3a shows the final LIE value on the six real-world social networks. The population size n are set to 10, 30, 50, 100 and 150, and the maximal number of iterations g_{max} is 100. It can be seen that with the increase of n value, the fitness value tends to increase gradually. However, when n increases from 50 to 150, the performance improvement is not obvious. When $n = 30$, the influence propagation reaches a relatively high value (Fig. 3a). Therefore, we set the parameter n to 30, taking into account the balance of performance and efficiency.

The maximal number of iterations is selected as 30, 50, 100 and 150. An interesting phenomenon can be observed from Fig. 3b. The effect of g on the performance of the algorithm is not obvious, because the fitness value increases slightly with the increase of the maximum number of iterations. However, when g is 100, DCSA performs relatively powerfully on most networks (Fig. 3b). Considering the tradeoff between performance and time cost, the maximal iteration number $g_{max} = 100$ of DCSA is reasonable.

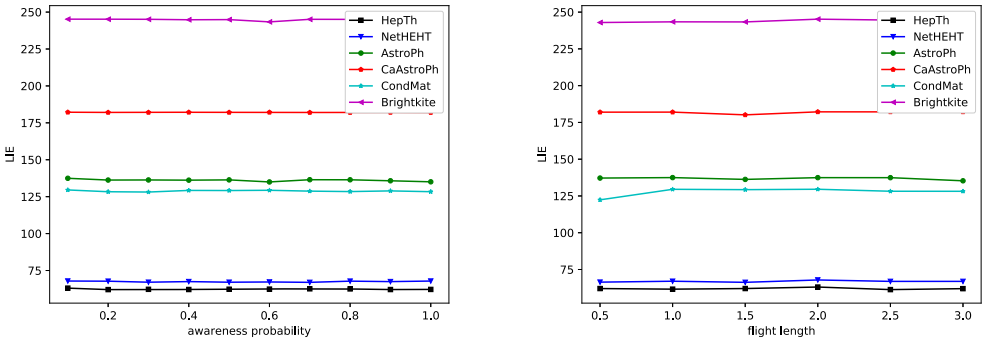


Fig. 2 The variations of fitness function LIE with different values of the awareness probability and flight length

(a) awareness probability (b) flight length

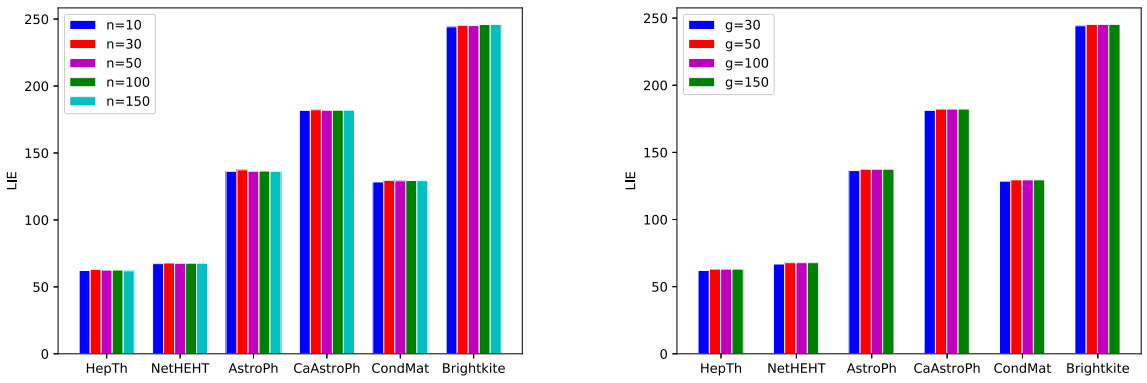
5.3 Experiments for the designed random walk

To show the performance of DCSA in optimizing the LIE and the influence of the designed random walk operator, LIE optimization experiments are conducted on six networks and DCSA-nc (a variation of DCSA, which does not adopt the random walk strategy proposed in this paper) and DPSO are compared. Besides the same seed set size k, the population size n and the maximal number of iterations g_{max}, the parameters of w, c_1 and c_2 in DPSO are set to 0.8, 2, 2, respectively. Fig. 4 shows the LIE optimization results on six networks. The x-axis represents the number of seed nodes, and the y-axis represents the logarithmic form of the fitness value. It can be seen from Fig. 4 that the three algorithms show good performance in the optimization of LIE, and are similar when the seed size k is small. When $k < 10$ in Fig. 4c-f and $k < 25$ in Fig. 4a and b, the LIE values obtained by the three algorithms are almost the same. However, when k increases, DCSA performes best in optimizing LIE in the experiments on six networks. From the figure, LIE optimization result of DCSA-nc is rising steadily, but its optimization effect is worse than DCSA. In addition, the evolution curve of DPSO simulation is unstable on large-scale networks, and the results are not comparable to DCSA and DCSA-nc, as shown in Fig. 4c-e.

The comparison between DCSA and DCSA-nc shows that the random walk strategy based on *Candidates* is effective and avoids DCSA falling into the local optimal in the early stage. Furthermore, it's showed that DPSO with local search strategy easily falls into suboptimal. Therefore, DCSA is then used to carry out influence diffusion experiments and compared with other algorithms to verify its effectiveness.

5.4 Comparison experiments

In order to further demonstrate the performance of DCSA, we carry out extensive experiments on six real-world social networks and compared it with other state-of-the-art influence maximization algorithms in terms of influence propagation and running time. In all experiments, the number of seed sets k varied from 1 to 50. And in all cases, the Monte-Carlo simulation numbers to measure the influence of seed set is set to 10000. In comparison, the parameters of the proposed algorithm are selected based



(a) size of population (b) maximal number of iterations

Fig. 3 The fitness value with different parameter n and g_{max}

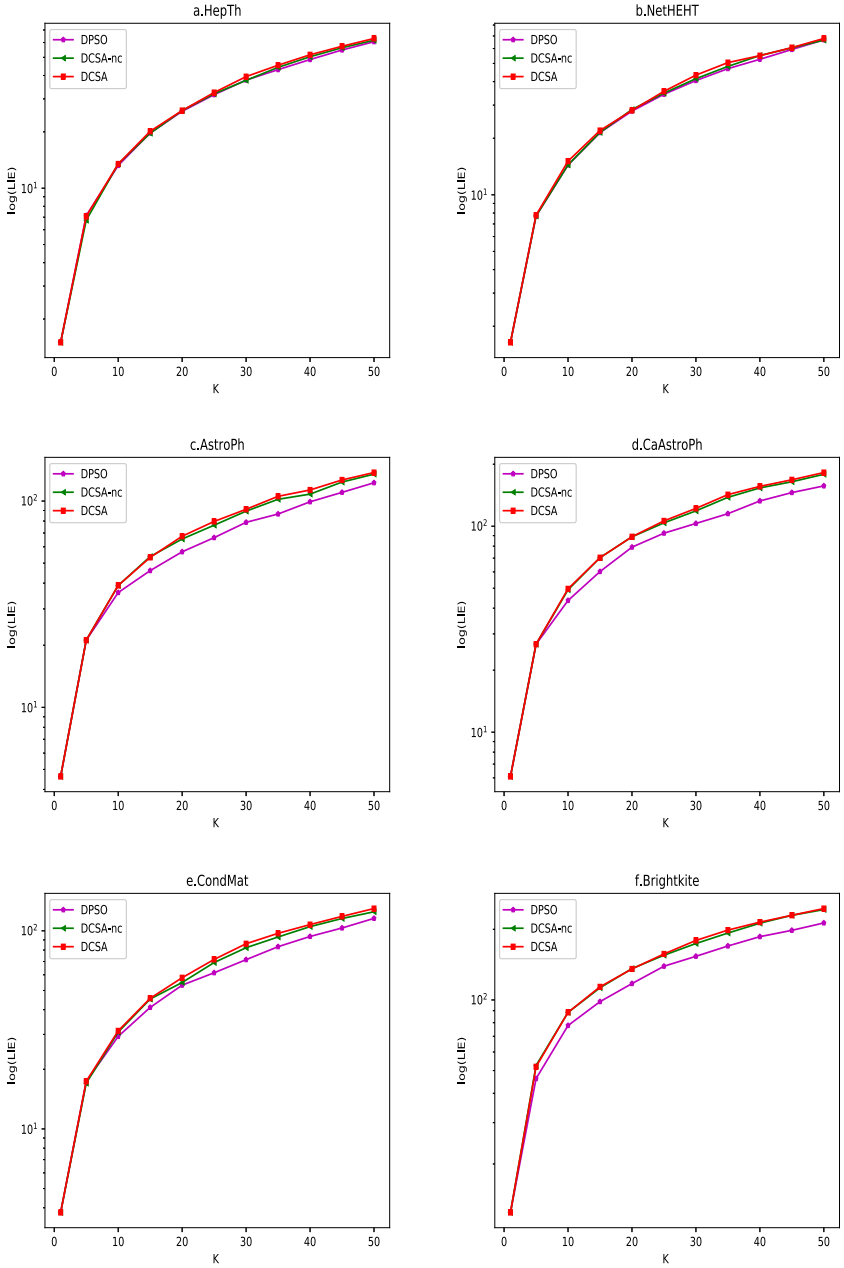


Fig. 4 Comparisons on LIE optimizaion of DCSA, DCSA-nc and DPSO on the six networks

on previous experiments as follows: AP=0.1, fl=2, n=30, g=100. The parameters of other baseline algorithms have been mentioned in Section 5.1.

5.4.1 Influence spread comparison

Given the propagation probability p=0.01, the results of the influence spread of the eight algorithms under the IC model are shown in the Fig. 5. The results in Fig. 5 reveal that all eight algorithms have stable influence spread performance on six social networks. DCSA achieves satisfying influence spread and a stable rising marginal gain with the increase of the target seed size, which indicates that DCSA is robust in finding the influential nodes.

In Fig. 5a and b, the influence spread of the DCSA algorithm is comparable to CELF, and better than the other algorithms. Compared with CELF and DCSA, SSA and DBA achieve less influence spread. LAIM and DDSE are ranked at a medium level among these algorithms. DPSO and IMPC have a relatively poor performance. Moreover, DPSO performs better than IMPC. In Fig. 5c and d, the proposed DCSA and CELF both have the best performance. The influence spread of DBA can match DCSA in Fig. 5c, but yield to DCSA in Fig. 5d. LAIM, DPSO and DDSE

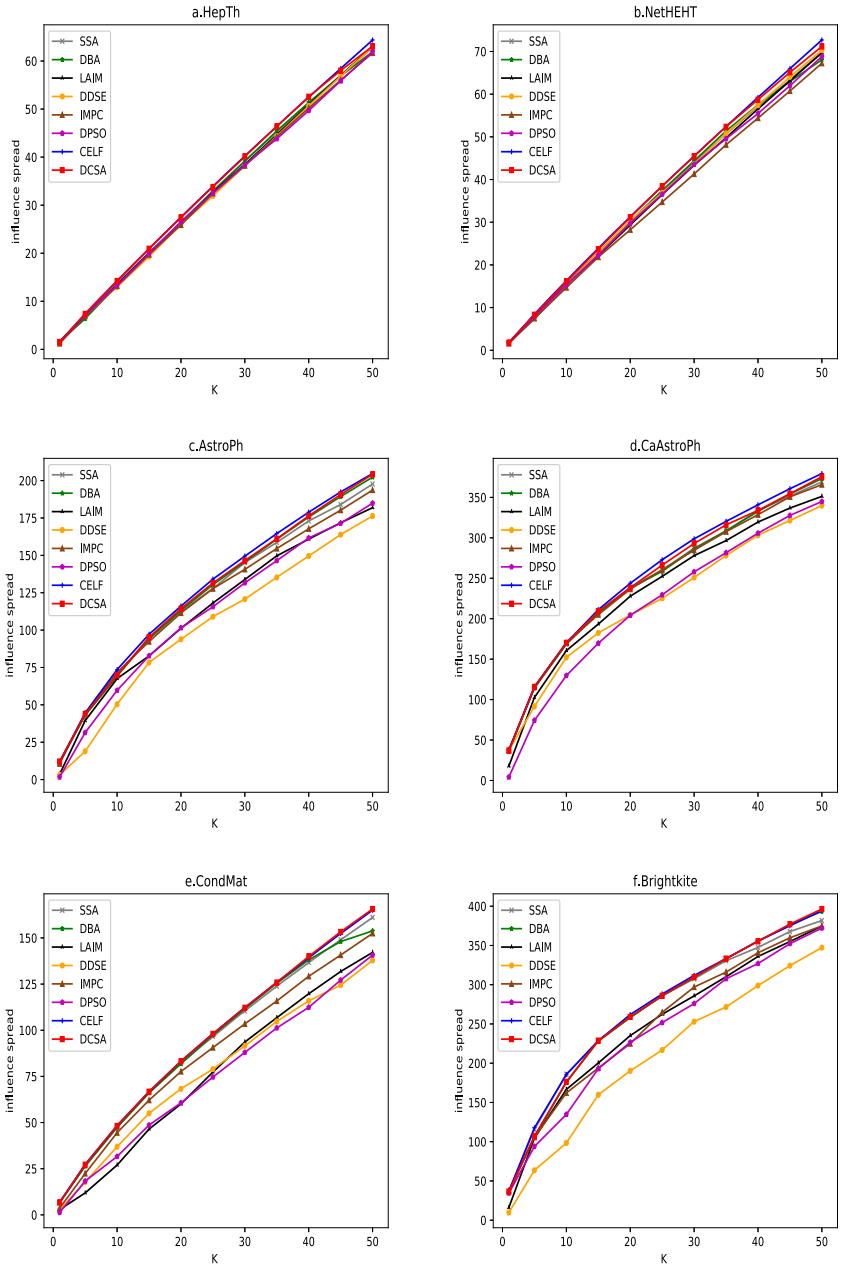


Fig. 5 Comparisons on influence spread of the eight algorithms under IC model ($p = 0.01$) on the six networks

have poor performance. In Fig. 5e and f, DCSA, CELF, SSA and DBA still perform powerfully. DCSA even shows a better solution than CELF. At the same time, DCSA is superior to SSA, DBA, LAIM, DDSE, IMPC and DPSO. The performance of DPSO and DDSE are worse than other algotithms, which shows that these two algorithms have instable performance in term of influence spread.

These experimental results in Fig. 5 prove DCSA algorithm can accurately identify influential nodes because of its effective evolutionary rule. And in the random walk startegy, the crow position is updated by potential more influential nodes in *Candidates*, which effectively avoids blind search in the whole network space. The performance of DCSA is slightly lower than that of CELF in four datasets, which is mainly due to the fact that DCSA uses two-hop neighbors area to estimate the influence of nodes.

The evolution rule of DPSO is effective, But when the local search cannot find a more influential node to replace the current candidate node, the local search strategy will terminate, and DPSO algorithm is easy to fall into the local optimal solution. In addition, DPSO has more parameters to determine than DCSA. As for DDSE, the algorithm uses EDV to estimate the influence of node set. However, EDV only considers one-hop neighbors of a node, which makes the estimation of influence inaccurate. Futhermore, the local search ability of standard difference algorithm is weak,

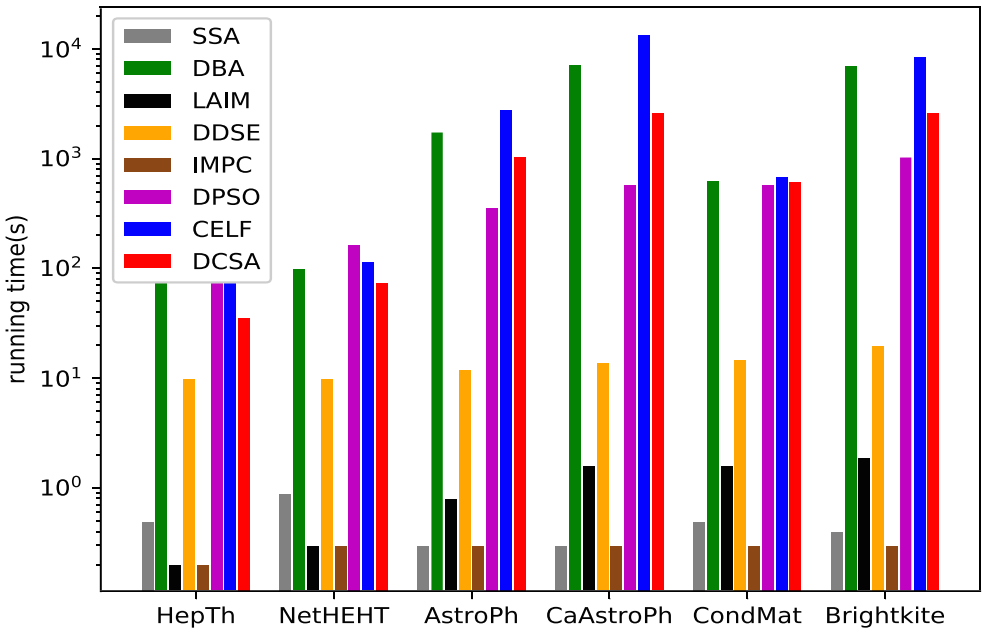


Fig. 6 Comparisons on running time of the eight algorithms at the targeted seed set size $k = 50$ on six networks

and it is difficult to maintain the diversity of population. Especially with the increase of search space, it is easy to lead to premature convergence. DDSE is easy to obtain low precision solutions. For the heuristic algorithm LAIM, its influence spread is smaller than that of CELF, DCSA, DBA and SSA, and worse than IMPC in AstroPh, CaAstroPh and CondMat networks (Fig. 5c, d and e), which shows using iterative formulas to calculate the influence of nodes is not accurate.

5.4.2 Running time comparison

In order to verify the efficiency of DCSA in searching the most influential nodes, we compared the running time of eight algorithms on six networks. Figure 6 shows the running time of all algorithms when the propagation probability p is 0.01 and k is 50. Figure 6 shows that SSA, LAIM and IMPC can identify the target seed nodes on six social networks in just tens of seconds. However, considering the influence spread obtained by these algorithms (Fig. 5), these three algorithms are often less effective than CELF and DCSA. On the contrary, Fig. 6 shows that greedy-based algorithm CELF is most time-consuming, and it even takes several hours to identify the target seed set on CaAstroPh graph, although it performs more effective than other algorithms. Compared with CELF, DCSA can identify influential nodes more efficiently and be extended to large-scale networks.

5.5 Statistical tests

In order to verify the effectiveness of DCSA, we conduct a statistical analysis to test whether the results of the eight algorithms on the six social networks of the influence maximization problem have high statistical significance.

In each network, 5 k scenarios ($k = 10, 20, 30, 40, 50$) are regarded as independent problems, and hypothesis tests are performed on each k scenarios in the six networks. The Wilcoxon rank sum test [65] are used to compare the DCSA algorithm and other comparison algorithms in pairs, and the results are shown in the Table 2. We set the level of confidence α of all cases to 0.05. From the statistical results, we can see that DCSA is significantly better than DBA, LAIM, DDSE, IMPC and DPSO algorithms. At the same time, it achieves almost the same performance guarantee as the greedy-based algorithm CELF.

Table 2 Statistical results of the Wilcoxon test for the eight algorithms at $\alpha = 0.05$

DCSA vs.	k	N+	N-	Z	p-value
SSA	10	4	2	−0.105	0.917
	20	5	1	−1.782	0.075
	30	6	0	−2.201	0.028
	40	5	1	−1.992	0.046
	50	6	0	−2.201	0.028
DBA	10	6	0	−2.201	0.028
	20	6	0	−2.201	0.028
	30	6	0	−2.201	0.028
	40	5	1	−1.992	0.046
	50	6	0	−2.201	0.028
LAIM	10	6	0	−2.201	0.028
	20	6	0	−2.201	0.028

Table 2 (continued)

DCSA vs.	k	N+	N-	Z	p-value
	30	6	0	−2.201	0.028
	40	6	0	−2.201	0.028
	50	6	0	−2.201	0.028
DDSE	10	6	0	−2.201	0.028
	20	6	0	−2.201	0.028
	30	6	0	−2.201	0.028
	40	6	0	−2.201	0.028
	50	6	0	−2.201	0.028
IMPC	10	5	1	−1.363	0.173
	20	6	0	−2.201	0.028
	30	6	0	−2.201	0.028
	40	6	0	−2.201	0.028
	50	6	0	−2.201	0.028
DPSO	10	6	0	−2.201	0.028
	20	6	0	−2.201	0.028
	30	6	0	−2.201	0.028
	40	6	0	−2.201	0.028
	50	6	0	−2.201	0.028
CELF	10	3	3	−0.524	0.600
	20	3	3	−0.105	0.917
	30	2	4	−1.363	0.173
	40	2	4	−1.153	0.249
	50	2	4	−0.734	0.463

6 Conclusion

In social network analysis, the influence maximization is still an important research topic. Therefore, furtherly developing effective, efficient and scalable methods is meaningful. In this paper, a novel evolutionary algorithm DCSA is proposed to solve the influence maximization problem on large-scale networks. A discrete coding mechanism and evolutionary rule are developed. According to the framework of the discrete crow search algorithm, a random walk strategy is redesigned. At the same time, according to the k-shell centrality and structural hole constraints of each node, a potential influential node *Candidates* is constructed to avoid the blindness of DCSA algorithm in random walk and accelerate the convergence of the algorithm. And a neighborhood optimization method called local search is used to operate the seed generated in the iterative process and get the final solution. We have carried out extensive experiments on real-world social networks and compared DCSA with the greedy-based algorithm and other algorithms. Experimental results show that DCSA with improved evolutionary rules and random walk strategy has satisfactory performance. DCSA has higher fitness value than another meta-heuristic algorithm DPSO. And it has lower time complexity than classical greedy-based algorithm CELF under IC model with propagation probability $p = 0.01$.

Although DCSA can effectively identify influential nodes, it still has some time-consuming issues when it expands to large-scale network. Therefore, the development of more effective influence diffusion estimation function and advanced evolution rules is the focus of our future work.

Funding This work is supported by the National Natural Science Foundation of China (Grant No.61702240), Lanzhou Science and Technology Project Funding (No. 2019-4-47), Fundamental Research Funds for the Central Universities (No. lzujbky-2020-6) and Key Research and Development Project of Gansu Province (No.18YF1FA132).

Declarations

Competing interests The authors declare that they have no conflict of interest.

References

1. Bond RM, Fariss CJ, Jones JJ, Kramer ADI, Marlow C, Settle JE, Fowler JH (2012) A 61-million-person experiment in social influence and political mobilization. Nature 489(7415):295–298. https://doi.org/10.1038/nature11421
2. Contractor NS, DeChurch LA (2014) Integrating social networks and human social motives to achieve social influence at scale. Proc Natl Acad Sci 111:13650–13657. https://doi.org/10.1073/pnas.1401211111
3. Peng S, Zhou Y, Cao L, Yu S, Niu J, Jia W (2018) Influence analysis in social networks: A survey. J Netw Comput Appl 106:17–32. https://doi.org/10.1016/j.jnca.2018.01.005
4. Cho Y, Hwang J, Lee D (2012) Identification of effective opinion leaders in the diffusion of technological innovation: A social network approach. Technol Forecast Soc Chang 79(1):97–106. https://doi.org/10.1016/j.techfore.2011.06.003
5. Li Y, Ma S, Zhang Y, Huang R, Kinshuk (2013) An improved mix framework for opinion leader identification in online learning communities. Knowl-Based Syst 43:43–51. https://doi.org/10.1016/j.knosys.2013.01.005
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, KDD'01. Association for Computing Machinery, New York, pp 57–66
7. Kempe D, Kleinberg J, Tardos E (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, KDD'03. Association for Computing Machinery, New York, pp 137–146
8. Leskovec J, Krause A, Guestrin C, Faloutsos C, Vanbriesen J, Glance N (2007) Cost-effective outbreak detection in networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol 420-429, pp 420–429
9. 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, KDD'09. Association for Computing Machinery, New York, pp 199–208
10. Cheng S, Shen H, Huang J, Zhang G, Cheng X (2013) Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM'13. Association for Computing Machinery, New York, pp 509–518
11. Jinha Kim HY (2013) Scalable and parallelizable processing of influence maximization for large-scale social networks? In 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp 266–277
12. Ju W, Chen L, Li B, Liu W, Sheng J, Wang Y (2020) A new algorithm for positive influence maximization in signed networks. Inf Sci 512:1571–1591. https://doi.org/10.1016/j.ins.2019.10.061, http://www.sciencedirect.com/science/article/pii/S0020025519310163
13. Guojie Song YW, Xie K (2015) Influence maximization on large-scale mobile social network: A divide-and-conquer method. IEEE Trans Parallel Distrib Syst 26(5):1379–1392. https://doi.org/10.1109/TPDS.2014.2320515
14. Shang J, Wu H, Zhou S, Zhong J, Feng Y, Qiang B (2018) Impc: Influence maximization based on multi-neighbor potential in community networks. Physica A: Stat Mech Appl 512:1085–1103. https://doi.org/10.1016/j.physa.2018.08.045
15. Huang H, Shen H, Meng Z, Chang H, He H (2019) Community-based influence maximization for viral marketing. Appl Intell 49(6):2137–2150. https://doi.org/10.1007/s10489- 018-1387-8
16. Borgs C, Brautbar M, Chayes J, Lucier B (2014) Maximizing social influence in nearly optimal time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'14. Society for Industrial and Applied Mathematics, USA, pp 946–957
17. Rui X, Meng F, Wang Z, Yuan G (2019) A reversed node ranking approach for influence maximization in social networks. Appl Intell 49(7):2684–2698. https://doi.org/10.1007/s10489-018-01398-w
18. Sabidussi G (1966) The centrality index of a graph. Psychometrika 31:581–603. https://doi.org/10.1007/BF02289527
19. Freeman L (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41. https://doi.org/10.2307/3033543
20. Freeman LC (1978) Centrality in social networks conceptual clarification. Soc Netw 1(3):215–239. https://doi.org/10.1016/0378-8733(78)90021-7
21. Bao Z-K, Liu J-G, Zhang H-F (2017) Identifying multiple influential spreaders by a heuristic clustering algorithm. Phys Lett A 381(11):976–983. https://doi.org/10.1016/j.physleta.2017.01.043
22. Yuan J, Zhang R, Tang J, Hu R, Wang Z, Li H (2019) Efficient and effective influence maximization in large-scale social networks via two frameworks. Physica A: Stat Mech Appl 526:120966. https://doi.org/10.1016/j.physa.2019.04.202
23. Alshahrani M, Fuxi Z, Sameh A, Mekouar S, Huang S (2020) Efficient algorithms based on centrality measures for identification of top-k influential users in social networks. Inf Sci 527:88–107. https://doi.org/10.1016/j.ins.2020.03.060, http://www.sciencedirect.com/science/article/pii/S0020025520302395
24. Xiong T, Bao Y, Hu Z, Chiong R (2015) Forecasting interval time series using a fully complex-valued rbf neural network with dpso and pso algorithms. Inf Sci 305:77–92. https://doi.org/10.1016/j.ins.2015.01.029
25. Wei J, Zhang R, Yu Z, Hu R, Tang J, Gui C, Yuan Y (2017) A bpso-svm algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. Appl Soft Comput 58:176–192. https://doi.org/10.1016/j.asoc.2017.04.061
26. Goyal A, Lu W, Lakshmanan LVS (2011) Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th International Conference Companion on World Wide Web, WWW'11. Association for Computing Machinery, New York, pp 47–48
27. Heidari M, Asadpour M, Faili H (2015) Smg: Fast scalable greedy algorithm for influence maximization in social networks. Physica A: Stat Mech Appl 420:124–133. https://doi.org/10.1016/j.physa.2014.10.088
28. Lu W-X, Zhou C, Wu J (2016) Big social network influence maximization via recursively estimating influence spread. Know.-Based Syst. 113(C):143–154. https://doi.org/10.1016/j.knosys.2016.09.020
29. Brin S, Page L (2012) Reprint of: The anatomy of a large-scale hypertextual web search engine. Comput Netw 56(18):3825–3833. https://doi.org/10.1016/j.comnet.2012.10.007. The WEB we live in
30. Zhang B, Wang Y, Jin Q, Ma J (2015) A pagerank-inspired heuristic scheme for influence maximization in social networks. Int. J. Web Serv. Res. 12(4):48–62. https://doi.org/10.4018/IJWSR.2015100104
31. Wang X, Su Y, Zhao C, Yi D (2016) Effective identification of multiple influential spreaders by degreepunishment. Physica A: Stat Mech Appl 461:238–247. https://doi.org/10.1016/j.physa.2016.05.020
32. 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, SIGMOD'14. Association for Computing Machinery, New York, pp 75–86
33. Tang Y, Shi Y, Xiao X (2015) Influence maximization in near-linear time: A martingale approach. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp 1539–1554
34. Nguyen HT, Thai MT, Dinh TN (2016) Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In: Proceedings of the 2016 International Conference on Management of Data, SIGMOD'16. Association for Computing Machinery, New York, pp 695–710
35. Cohen E, Delling D, Pajor T, Werneck RF (2014) Sketch-based influence maximization and computation: Scaling up with guarantees. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM'14. Association for Computing Machinery, New York, pp 629–638
36. Wang X, Zhang Y, Zhang W, Lin X, Chen C (2017) Bring order into the samples: A novel scalable method for influence maximization. IEEE Trans Knowl Data Eng 29(2):243–256. https://doi.org/10.1109/TKDE.2016.2624734
37. Kim D, Hyeon D, Oh J, Han W-S, Yu H (2017) Influence maximization based on reachability sketches in dynamic graphs. Inf Sci 394-395:217–231. https://doi.org/10.1016/j.ins.2017.02.023
38. Jiang Q, Song G, Cong G, Wang Y, Si W, Xie K (2011) Simulated annealing based influence maximization in social networks. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11. AAAI Press, pp 127–132
39. Gong M, Yan J, Shen B, Ma L, Cai Q (2016) Influence maximization in social networks based on discrete particle swarm optimization. Inf Sci 367-368:600–614. https://doi.org/10.1016/j.ins.2016.07.012
40. Gong M, Song C, Duan C, Ma L, Shen B (2016) An efficient memetic algorithm for influence maximization in social networks. Comp Intell Mag 11(3):22–33. https://doi.org/10.1109/MCI.2016.2572538
41. Cui L, Hu H, Yu S, Yan Q, Ming Z, Wen Z, Lu N (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. https://doi.org/10.1016/j.jnca.2017.12.003
42. Zareie A, Sheikhahmadi A, Jalili M (2020) Identification of influential users in social network using gray wolf optimization algorithm. Expert Syst Appl 142:112971. https://doi.org/10.1016/j.eswa.2019.112971
43. Tang J, Zhang R, Yao Y, Zhao Z, Wang P, Li H, Yuan J (2018) Maximizing the spread of influence via the collective intelligence of discrete bat algorithm. Knowl-Based Syst 160:88–103. https://doi.org/10.1016/j.knosys.2018.06.013, http://www.sciencedirect.com/science/article/pii/S0950705118303423
44. Chen W, Wang C, Wang Y (2010) Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'10. Association for Computing Machinery, New York, pp 1029–1038
45. Christakis NA, Fowler JH (2009) Connected: The surprising power of our social networks and how they shape our lives. Little, Brown
46. Pei S, Muchnik L, Andrade JJS, Zheng Z, Makse HA (2014) Searching for superspreaders of information in real-world social media. Sci Rep 4(1):5547. https://doi.org/10.1038/srep05547
47. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Comput Struct 169:1–12. https://doi.org/10.1016/j.compstruc.2016.03.001
48. Dos Santos Coelho L, Richter C, Mariani VC, Askarzadeh A (2016) Modified crow search approach applied to electromagnetic optimization. In: 2016 IEEE Conference on Electromagnetic Field Computation (CEFC), pp 1–1
49. Roy R, Sahu TP, Nagwani NK, Das S (2021) Global best guided crow search algorithm for optimization problems. In: Kumar R, Singh VP, Mathur A (eds) Intelligent Algorithms for Analysis and Control of Dynamical Systems. Springer Singapore, Singapore, pp 13–22. https://doi.org/10.1007/978-981-15-8045-1_2
50. Turgut MS, Turgut OE, Eliiyi DT (2020) Island-based crow search algorithm for solving optimal control problems. Appl Soft Comput 90:106170. https://doi.org/10.1016/j.asoc.2020.106170, http://www.sciencedirect.com/science/article/pii/S1568494620301101
51. Ke Y, Xie J, Pouramini S (2021) Utilization of an improved crow search algorithm to solve building energy optimization problems: Cases of australia. J Build Eng 38:102142. https://doi.org/10.1016/j.jobe.2020.102142, http://www.sciencedirect.com/science/article/pii/S2352710220337748
52. Cao L, Yue Y, Zhang Y, Cai Y (2021) Improved crow search algorithm optimized extreme learning machine based on classification algorithm and application. IEEE Access:1–1. https://doi.org/10.1109/ACCESS.2021.3054799
53. Dash R, Samal S, Dash R, Rautray R (2019) An integrated topsis crow search based classifier ensemble: In application to stock index price movement prediction. Appl Soft Comput 85:105784. https://doi.org/10.1016/j.asoc.2019.105784, http://www.sciencedirect.com/science/article/pii/S1568494619305654
54. Huangpeng Q, Huang W, Gholinia F (2021) Forecast of the hydropower generation under influence of climate change based on rcps and developed crow search optimization algorithm. Energy Rep 7:385–397. https://doi.org/10.1016/j.egyr.2021.01.006, http://www.sciencedirect.com/science/article/pii/S235248472100007X
55. De Souza RCT, d. S. Coelho L, De Macedo CA, Pierezan J (2018) A v-shaped binary crow search algorithm for feature selection. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp 1–8
56. Ahmad M, Abdullah M, Moon H, Yoo SJ, Han D (2020) Image classification based on automatic neural architecture search using binary crow search algorithm. IEEE Access 8:189891–189912. https://doi.org/10.1109/ACCESS.2020.3031599
57. Bharanidharan N, Rajaguru H (2021) Classification of b-cell acute lymphoblastic leukemia microscopic images using crow search algorithm. In: Lim CT, Leo HL, Yeow R (eds) 17th International Conference on Biomedical Engineering. Springer International Publishing, Cham, pp 143–154
58. Hussien AG, Amin M, Wang M, Liang G, Alsanad A, Gumaei A, Chen H (2020) Crow search algorithm: Theory, recent advances, and applications. IEEE Access 8:173548–173565. https://doi.org/10.1109/ACCESS.2020.3024108
59. Allaoui M, Ahiod B, El Yafrani M (2018) A hybrid crow search algorithm for solving the dna fragment assembly problem. Expert Syst Appl 102:44–56. https://doi.org/10.1016/j.eswa.2018.02.018, http://www.sciencedirect.com/science/article/pii/S0957417418300976
60. Haryono A, Sungkono, Agustin R, Santosa BJ, Widodo A, Ramadhany B (2021) Correction to: Model parameter estimation and its uncertainty for 2-d inclined sheet structure in self-potential data using crow search algorithm. Acta Geodaetica et Geophysica. https://doi.org/10.1007/s40328-020-00330-4
61. Laabadi S, Naimi M, Amri HE, Achchab B (2020) A binary crow search algorithm for solving two-dimensional bin packing problem with fixed orientation. Procedia Comput Sci 167:809–818. https://doi.org/10.1016/j.procs.2020.03.420, http://www.sciencedirect.com/science/article/pii/S1877050920308863
62. Kitsak M, Gallos L, Havlin S, Liljeros F, Muchnik L, Stanley H, Makse H (2010) Identification of influential spreaders in complex networks. Nat Phys 6:888–893. https://doi.org/10.1038/nphys1746
63. Lazega E, Burt R (1995) Structural holes: The social structure of competition. Rev Fran Sociol 36:779. https://doi.org/10.2307/3322456
64. Wu H, Shang J, Zhou S, Feng Y, Qiang B, Xie W (2018) Laim: A linear time iterative approach for efficient influence maximization in large-scale networks. IEEE Access 6:44221–44234
65. García S, Molina D, Lozano M, Herrera F (2008) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. J Heuristics 15(6):617. https://doi.org/10.1007/s10732-008-9080-4