Neurocomputing 334 (2019) 89–99

Contents lists available at ScienceDirect

Neurocomputing

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

A novel modularity-based discrete state transition algorithm for community detection in networks

Xiaojun Zhou[a], Ke Yang[a], Yongfang Xie[a,*], Chunhua Yang[a], Tingwen Huang[b]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, China
[b] Texas A & M University at Qatar, Doha 23874, Qatar

ARTICLE INFO

Article history:
Received 7 August 2018
Revised 11 December 2018
Accepted 3 January 2019
Available online 11 January 2019

Communicated by Prof. J. Cao

Keywords:
State transition algorithm
Complex network
Community detection
Modularity

ABSTRACT

Complex network analysis is a hot topic in the data mining area which aims to reveal the hidden information behind a network. As an important tool in complex network analysis, community detection tries to perform a network clustering operation to find the community structure, which can be formulated as an optimization problem. In the past few decades, various of community detection algorithms have been designed to address this challenging problem. Although many algorithms are feasible to detect the network partitions, most of them only get suboptimal solutions or have poor stability. The state transition algorithm (STA) is a novel intelligent paradigm for global optimization, and it exhibits powerful global search ability in various complex optimization problems. Thus, in this paper, a novel modularity-based discrete state transition algorithm (MDSTA) is proposed to obtain more optimal and stable solutions. Moreover, based on the heuristic information of the network, vertex substitute transformation operator and community substitute transformation operator are proposed for global search. Then, each initialized individual evolves through these two substitute operations. Next, an elite population that contains individuals with high fitness values is selected from these evolved individuals. Finally, a two-way crossover operation among the elite population is conducted for local search. The framework of MDSTA is pretty simple and easy to implement. Several state-of-art community detection algorithms are used to compare with MDSTA both on artificial networks and real-world networks. The experimental results demonstrate that MDSTA is effective and stable for community detection in networks.

1. Introduction

Nowadays, networks such as social networks, traffic networks, biological networks, and computer networks are everywhere, and various networks constitute the world we live in. For network science researchers, networks can be represented as graphs contain nodes and edges. Nodes are the entities in a system, while an edge indicates there is an interaction between two entities [1]. Among different features of networks, community structure has received widespread attention. Community structure, i.e. the division of a network into groups of nodes having dense intra-connections, and sparse inter-connections [2], is an important characteristic of networks. Furthermore, community detection or network clustering which can discover community structure in the network is meaningful for us to reveal the information behind the networks.

To solve the community detection problem, various types of methods have been proposed and developed, including block approximation model [3], label propagation model [4–6], and modularity maximization model [2,7–9], information-theoretical model [10–12]. Moreover, some new methods like dynamic cluster formation game [13], community detection by finding network's central nodes [14] are proposed in recent years. Of all the algorithms, the modularity maximization model is the most popular one which transforms community detection problem into an optimization problem. Since modularity was put forward by Newman [8], many evolutionary algorithms are widely used in community detection due to their powerful search capabilities. Gong et al. [15] proposed a multi-objective discrete particle swarm algorithm based on modularity for community detection. In [16], a novel clonal selection algorithm is used for finding network communities. Song et al. [17] proposed a community detection algorithm based on bat algorithm. In [18], they introduced a multi-agent genetic algorithm for community detection in complex networks. Guerrero et al. [19] used genetic algorithms for

* Corresponding author.
E-mail address: yfxie@csu.edu.cn (Y. Xie).

https://doi.org/10.1016/j.neucom.2019.01.009
0925-2312/

adaptive network clustering by optimizing modularity. However, most modularity-based algorithms can not obtain stable results and often get into local optima, which prompts us to find an algorithm to get better and more stable solutions.

In this paper, a new intelligent global optimization paradigm called state transition algorithm(STA) [20] is introduced to solve this problem. STA optimizes a problem by different kinds of state transformation operations. In STA, a solution to a specific optimization problem is considered as a state, and the procedure of updating a current solution is considered as state transition. STA generates candidate solutions from the current solution by a certain state transformation operator, and then it selects a solution from the candidate set as the new current solution. This process is repeated for several times by using different kinds of state transformation operators until the specified termination criteria are satisfied. In continuous optimization problems, STA shows fantastic performance due to its effective global search ability [21–28]. In [29], a discrete STA was proposed to solve several unconstrained integer optimization problems including traveling salesman problem, boolean integer programming and discrete value selection. Its excellent global search ability was validated by comparing with other state-of-art algorithms. Since the community detection problem is also an unconstrained integer optimization problem, we extend the discrete STA and propose a novel modularity-based discrete state transition algorithm (MDSTA) to find the optimal or approximate optimal solution for community detection problem. Experiments on both artificial networks and real-world networks are conducted by the proposed MDSTA. The experimental results show that the proposed method is more useful and effective than other modularity-based algorithms.

The key contributions of the proposed method include:

(1) The traditional methods based on evolutionary computation directly perform crossover and mutation operations in the initial population, which is often blind and ineffective, while MDSTA optimizes each initial individual separately, and then it selects individuals with high fitness value as the elite population for local search, which is more reasonable.
(2) Two kinds of state transformation operators including vertex substitute operator and community substitute operator are designed based on the priori knowledge of networks.
(3) To avoid getting trapped into local optimal solutions, a new local search strategy using the two-way crossover in the elite population is proposed.

The structure of this paper is organized as follows: Section 2 presents the description of community detection problem. Section 3 shows the proposed discrete STA in detail, then two kinds of state transformation operators and a new local search method are illustrated. Experimental analysis are shown in Section 4. Finally, Section 5 concludes the paper.

2. Problem formulation

We consider a basic unweighted and undirected network which does not has overlapping communities. Thus, a network is denoted as $G = (V, E)$, V means vertices, and E represents edges which connect two vertices. In order to describe the topological structure of a network, the adjacency matrix A has been introduced. The elements of A can be written as A_{ij}, where $A_{ij} = 1$ if nodes i and j are linked and $A_{ij} = 0$, otherwise.

In order to find communities in a network, the definition of the community should be given. A network with N communities can be denoted as $\Theta = \{\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(N)}\}$, where $\Theta^{(i)}$ is community i. Typically, for a network, the community refers to a collection of closely connected nodes, and the connections between different communities are sparse [30]. Furthermore, community

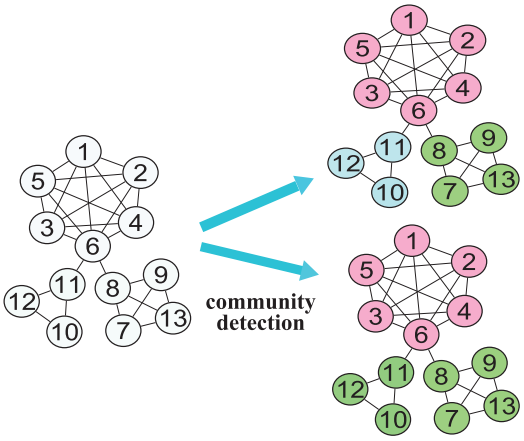


Fig. 1. Graphical illustration of community detection in a simple network

detection for a simple network can be illustrated in Fig. 1. In this network, many possible divisions can be performed and two kinds of divisions are shown in the figure. As we can imagine, for complex networks, the divisions are more diverse.

Community detection aims to find an optimal division of a network, naturally the detection of community structure in a network can be formulated as an optimization problem. If Ω_i indicates the i_{th} kind of partitions of the network, $\Omega = \{\Omega_1, \ldots, \Omega_i, \ldots, \Omega_n\}$ means n kinds of feasible network partitions. In order to discover the optimal partition Ω^*, the community detection problem can be formulated as an optimization problem

$$F(\Omega^*) = \min F(\Omega), \tag{1}$$

where $\Omega^* \in \Omega$, and F is the fitness function that can measure the quality of the obtained community structure [31].

3. Modularity-based discrete STA for community detection

In the discrete STA, state and state transition are two main elements, where a state stands for a solution and the state transition means the transformation operation aiming to generate a new solution. The basic framework of generation of a solution in STA can be described as

$$\begin{cases} x_{k+1} = A_k(x_k) \oplus B_k(u_k) \\ y_{k+1} = f(x_{k+1}) \end{cases}, \tag{2}$$

where x_k and x_{k+1} are the current state and the next state, respectively; A_k and B_k are the specific state transformation operators; u_k stands for a function of current state and historical states; $\oplus$ is a kind of operation which aims to connect two states; f is the fitness function, and y_{k+1} is the fitness value at x_{k+1}.

3.1. Fitness function

It is important for the community detection problem to find a suitable criterion function. For the choice of objective function, the most popular one is the modularity function, which was proposed and developed by Newman [8]. The modularity of a network G(V, E) often denoted as Q and it can be written as

$$Q = \frac{1}{2M} \sum \left(A_{ij} - \frac{K_i K_j}{2M} \right) \delta(i, j), \tag{3}$$

where δ is the kronecker function which equals 1 when node i and node j are in the same community, otherwise $\delta = 0$; A is the adjacency matrix; M means the number of edges in the graph; K_i and K_j represent the respective degree of nodes i and j. In general, the larger the value of Q, the better the performance of the community detection algorithm.

vertex	1	2	3	4	5	6	7	8	9	10	11	12	13
state 1	1	1	1	1	1	1	2	2	2	3	3	3	2
state 2	1	1	1	1	1	1	2	2	2	2	2	2	2

Fig. 2. State representations of the network partitions in Fig. 1.

3.2. State representation and initialization

Since the modularity is used as the fitness function, and other variables are already given by the linkage of the network, we only need to know the value of $\delta(i, j)$ which depends on the community structure of the network. Thus, a state or a solution need to be encoded to presents the corresponding community structure of a network. The proposed MDSTA uses the label-based representation as the encoding scheme. To a certain partition of a complex network $G(V, E)$ with n nodes, a state is encoded as

$$X = (x_1, x_2, \ldots, x_n) \tag{4}$$

where x_i stands for the community label, and $x_i = k$ means that node i belongs to the k_{th} community; node i and node j come from the same community when $x_i = x_j$. For the two kinds of network partitions shown in Fig. 1, their correspongding state representation can be illustrated in Fig. 2.

The initialization of solutions is an important part for an algorithm. If the state is generated by assigning random values to each node, the network will be too disordered to optimize. So, in order to reduce the time complexity and give an appropriate partition of the network, a method based on label propagation proposed by Gong et al. [32] was introduced. In this method, the label of each node is updated according to its neighbors which can be written as

$$MaxLabel = \underset{k}{argmax}\, \Sigma_{j\in\Phi(i)}\delta(l(j), k), \tag{5}$$

where $\Phi(i)$ represents the neighbor set of node i; $l(j)$ means the label of node j, and $k \in N_+$; $\delta(m, n)$ yields one if m equals n, zero otherwise.

The pseudo code of the algorithm can be seen in Algorithm 1.

Algorithm 1 State initialization with label propagation based method.

Input: network adjacency matrix: A; iterations
Output: initialized state: Best

```
Best = 1 : n
for iter = 1 : iterations do
    sequence ← randperm(n)
    for k = 1 : nodes.number do
        i ← sequence(k)
        if Φ(i).size ≥ 1 then
            MaxLabel ← Eq.(5)
            Best(i) ← random(MaxLabel)
        end if
    end for
end for
```

During the initialization process, the algorithm first initializes each node in the network as a unique label. For each iteration, a label propagation operation will be performed for every vertex in a random order. If a label value (denoted as v) in the neighbors of node i appears most frequently, the label of node i will be changed to v; if the number of different labels in the neighbors of node i appears the same, then we randomly select one of them as the label of node i. After the label propagation, closely connected nodes are quickly divided into the same community. The *iterations* in Algorithm 1 is set as 5 when $n > 1000$, and 1 otherwise. Fig. 3 displays the community structure of the Karate network after the initialization step.

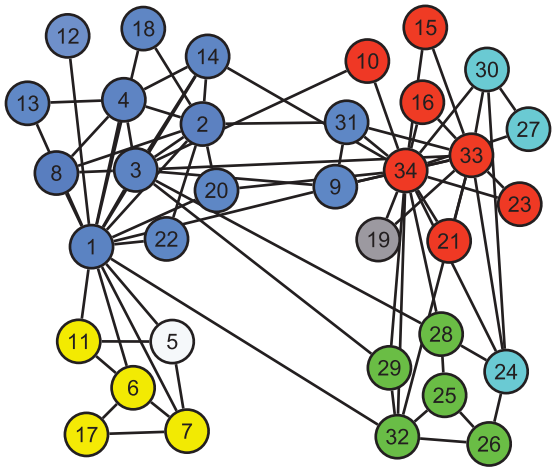


Fig. 3. Community structure of the Karate network after initialization.

3.3. State transformation operators

State transformation operators are the crucial components of STA. In our previous study, four kinds of operators including swap operator, shift operator, symmetry operator and substitute operator are designed to solve different kinds of unconstrained integer optimization problems [33]. Owing to the fact that specific optimization problems have their own characteristics, these specially designed operators perform not so good in the community detection problem. Thus, two operators called vertex substitute operator and community substitute operator are designed based on the label-based representation method and the characteristics of complex networks.

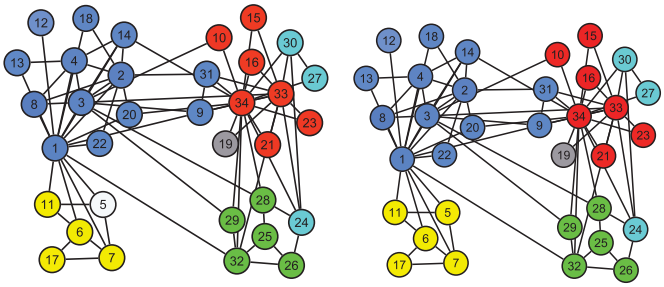
3.3.1. Vertex substitute transformation

Substitute transformation operation aims to modify the values of some positions in a state to generate new states. Therefore, it could increase the diversity of solutions and the possibility of finding better solutions. In MDSTA, the process from a state to generate a new state can be written as

$$X_{k+1} = A_k^{sub}(m_d)X_k \tag{6}$$

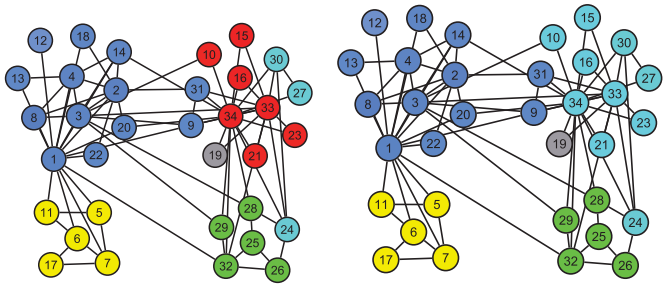
where, $A_k^{sub} \in R^{n\times n}$ stands for a substitute transformation matrix; X_k and X_{k+1} are the current state and the generated state, respectively; m_d equals n means that n positions in the state will be changed at most. In the vertex substitute transformation $m_d = 1$ and it represents that only the value in a certain position will be substituted.

Although substitute transformation should possess a certain degree of randomness to generate a variety of solutions, completely random substitute transformation could hardly produce promising states and it will increase time complexity. So, the vertex substitute transformation operator is designed according to the linkage of the network. The main idea of the operator is to substitute the label of a node according to its neighbors' label. However, not all the nodes are suitable for this operation, such as node 4 in Fig. 3, for the reason that it has the same label value with all the neighbor nodes. On this account, a definition of *potential node* is given as bellow.



(a) Q=0.3300,NMI=0.4507 (b) $Q = 0.3524, NMI = 0.4653$

Fig. 4. Illustration of vertex substitute transformation.



(a) Q=0.3524,NMI=0.4653 (b) $Q = 0.3812, NMI = 0.5054$

Fig. 5. Illustration of community substitute transformation.

Definition 1 (*Potential node*): For a given network partition, if the label of node i is not completely the same with its all neighbors, then we call node i as *potential node*.

The pseudo code of vertex substitute transformation is shown in Algorithm 2. In the Algorithm, a solution (state) and its modularity value are denoted as *Best* and *fBest*, respectively. This operator will conduct *SE times* vertex substitute operations to the *Current_Best* after finding the potential nodes based on the current network partition. In each substitute operation, a node k is randomly selected from all the potential nodes, and the neighbor nodes of node k are denoted as $\Phi(k)$. Then, the label of k is replaced with the label of a randomly selected neighbor node which is different from k's label value. After *SE* times substitute operations, the operator generates *SE* new states. The *Current_Best* and *SE* generated states constitute the candidate solution set. Finally, the operator returns the state with largest modularity value in the candidate solution set.

Algorithm 2 Vertex substitute transformation.

Input: A; *Current_Best*; *Current_fBest*; *SE*
Output: *Next_Best*; *Next_fBest*

```
potential_nodes ← find potential nodes based on Current_Best
for j = 1:SE do
    temp ← Current_Best
    k ← random(potential_nodes)
    label.Φ(r)(label.Φ(k) == temp(k)) ← []
    temp(k) ← random(label.Φ(r))
    State(j, :) ← temp
end for
[Next_Best, Next_fBest] ← select the state with the highest modularity value
if Next_fBest < Current_fBest then
    Next_Best ← Current_Best
    Next_fBest ← Current_fBest
end if
```

Fig. 4 gives a graphical illustration of vertex substitute transformation based on the community structure in Fig. 3. In this illustration, the potential node 5 is selected. Then, the label of node 5 is substituted by its neighbor's label.

3.3.2. Community substitute transformation

We also design a substitute operator with $m_d > 1$ which is called community substitute operator, and it will perform substitute operation to a randomly selected community. A lot of small-scale communities generated after the initialization process, including some communities that have tight connection. Therefore, community substitute transformation is designed to merge these tight connection communities into the same community to optimize the partition of the networks. Similar with the vertex substitute transformation, we use the label information of the neighbor communities to determine the label of the selected community. Here, the definition of *neighbor community* is given as follows:

Definition 2 (*Neighbor community*): Given a partition $\Theta = \{\Theta^{(1)}, \ldots, \Theta^{(i)}, \ldots, \Theta^{(N)}\}$ of the network G, and the number of communities is N. For any community $\Theta^{(i)} \in \Theta$, as long as it has one node directly connected to the community $\Theta^{(k)} \in \Theta$, we define $\Theta^{(i)}$ as a *neighbor community* of community $\Theta^{(k)}$.

The whole procedure can be described in Algorithm 3. As shown in Algorithm 3, the potential nodes are used to search for the neighbor communities. The only difference between the vertex substitute transformation and the community substitute transformation lies on the method for generating candidate solutions. The community substitute transformation conducts *SE* times of substitute operations to a state (*Next_Best*). In each procedure, we randomly choose a node k which is a potential node in the current network partition, and select its community $\Theta^{(k)}$. Then, the label of $\Theta^{(k)}$ is changed to the label of a randomly selected neighbor community, and the lable of the selected neighbor community is different from $\Theta^{(k)}$'s label. In the end, *SE* new states and *Next_Best* constitute the candidate solutions.

Algorithm 3 Community substitute transformation.

Input: A; *Next_Best*; *Next_fBest*; *SE*
Output: *New_Best*; *New_fBest*

```
potential_nodes ← find potential nodes based on Next_Best
for j = 1:SE do
    temp ← Next_Best
    k ← random(potential_nodes)
    Θ(k) ← the community of node k
    k_potential_nodes ← potential nodes in Θ(k)
    label_value ← label.Φ(k_potential_nodes)
    label_value(label_value == temp(k)) ← []
    temp(temp = temp(k)) ← random(label_value)
    State(j, :) ← temp
end for
[New_Best, New_fBest] ← select the state with the highest modularity value
if New_fBest < Next_fBest then
    New_Best ← Next_Best
    New_fBest ← Next_fBest
end if
```

Fig. 5 gives the result of a kind of community substitute transformation. In this illustration, the potential node 10 and its corresponding community are selected. Then, the label of the select community is substituted by its neighbor community's label.

3.3.3. Two-way crossover for local search

Two-way crossover is a crossover operation which was proposed by Gong et al. [32]. An example of two-way crossover is

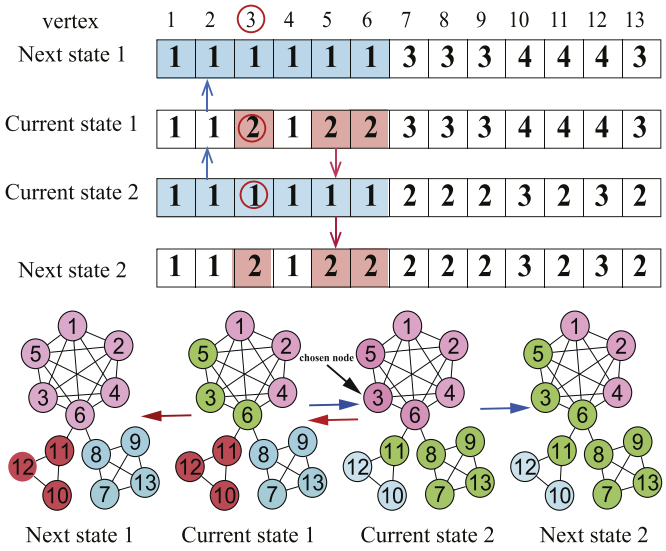


Fig. 6. Illustration of two-way crossover operation.

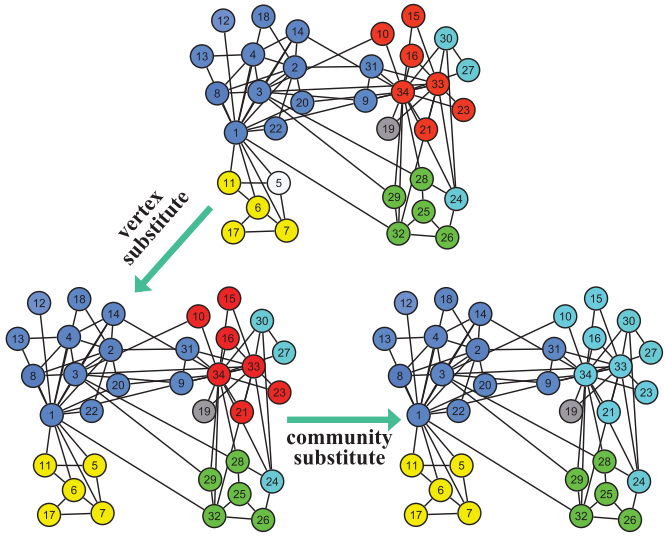


Fig. 7. Illustration of the Initialization and Evolution step.

given in Fig. 6. We will only introduce the steps that how *next state 1* is generated from *current state 1*, since it is changed in the same way from *current state 2* to *next state 2*. Firstly, the vertex 3 (labeled as pink) is randomly selected in *current state 2*. In this state, the community whose label is pink includes vertexes 1, 2, 3, 4, 5 and 6. Then these vertexes in *current 1* will be chosen for substitute operation. Finally, the labels of these nodes in *current 1* are substituted by pink (the label of vertex 3 in *current state 2*), and thus *next state 1* is generated.

In traditional methods, the two-way crossover is usually used as a genetic operator in the early period of an algorithm to enhance the global search ability. However, in the early stages of the algorithm, the solutions are not good enough to adopt the two-way crossover operation to product more promising solutions, and it often brings extra computational cost. While at the later stage of the algorithm, many solutions approach the global optimal solution and they are generally seen as potential solutions. Hence, the two-way crossover operation is introduced for local search after using of the state transformation operators in our method. Experiments show that this local search strategy can help MDSTA to find the

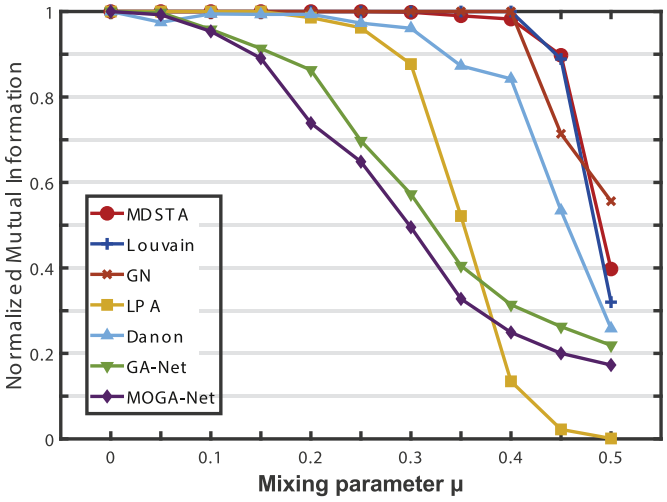


Fig. 8. Experimental results in the artificial networks.

optimal solution and thus improve the performance in the community detection problem.

3.4. States-updating strategies

An optimization algorithm with good performance should have two key properties. One is that the algorithm can generate good candidate solutions, and the other is that it has appropriate selection strategies to select promising candidate solutions. In MDSTA, state transformation operators are designed to generate new candidate solutions which are often better than the current solution, and thus the first property is satisfied.

Meanwhile, there are two kinds of selection strategies for states-updating in our algorithm. For one thing, according to Algorithms 2 and 3, the modularity value will not be worse after each state transformation operation. Actually, it uses the greedy search strategy. For another, according to Algorithm 4, every initial state (*Current_Best*0) will repeatedly performs vertex substitute transformation and community substitute transformation until the operators could not get a state with larger modularity value for a certain (*stagnate_number*) times. If the new state (*New_Best*) which is updated by the two substitute transformation operators can generate larger fitness value than the state which is updated in the last iteration, the new solution will be saved to constitute an optimal solution set (*Best_History*). Next, we select the solutions from the optimal solution set whose fitness values are in the top *n* (*elite_popsize*) as the elite population. Finally, two individuals (*Best1* and *Best2*) which are randomly selected from the elite population will conduct the two-way crossover operation for several (*MaxIter*· *SE*) times.

3.5. Algorithm framework and complexity analysis

The framework of the proposed MDSTA for community detection is given in Algorithm 4. The entire algorithm is divided into three main steps: Initialization and Evolution, Generate Elite Population, and Local Search. Besides, Fig. 7 gives a graphical illustration of one iteration in the Initialization and Evolution step to better understand the process.

For a network with *n* nodes and *m* edges, according to the algorithm outline, two substitute operators play a vital role and they need the most computational time. According to Algorithm 2, the main computational cost is to find the potential nodes. The algorithm determines whether the node is a potential node by

Algorithm 4 The pseudo code of MDSTA.

Input: network adjacency matrix A; search enforcement SE; *stagnate_number*; *MaxIter*;*popsize*; *elite_popsize*;
Output: the network partition finded by MDSTA

Initialization and Evolution:
Best_History ← [], *fBest_History* ← []
for $i = 1$: *popsize* **do**
  Initial state: *Current_Best*; fitness value: *Current_fBest*; *stagnate* ← 0
  while *stagnate* < *stagnate_number* **do**
    [*Next_Best*,*Next_fBest*]← *Vertex_substitute*(*A*, *Current_Best*, *Current_fBest*, *SE*)
    [*New_Best*,*New_fBest*]← *Community_substitute*(*A*, *Next_Best*, *Next_fBest*, *SE*)
    if *New_fBest* > *Current_fBest* **then**
      stagnate ← 0
      Best_History ← [*Best_History*; *New_Best*]
      fBest_History ← [*fBest_History*; *New_fBest*]
    else
      stagnate ← *stagnate* + 1
    end if
    Current_Best←*New_Best*, *Current_fBest*←*New_fBest*
  end while
end for
Generate Elite Population:
if *length*(*fBest_History*) > *elite_popsize* **then**
  [*fBest_History*,order]←sort(*fBest_History*,'descend')
  Best_History←*Best_History*(order, :)
  Best_History←*Best_History*(1:*elite_popsize*, :)
  fBest_History←*fBest_History*(1:*elite_popsize*, :)
end if
Local Search:
for $i = 1$: *MaxIter* **do**
  Randomly select *Best1* and *Best2* from the elite population
  [New_Best,New_fBest] ← *twoway_crossover(A,Best1,Best2,SE)*
  Best_History ← [*Best_History*; *New_Best*]
  fBest_History ← [*fBest_History*; *New_fBest*]
end for
[*fBest*,position]←max(*fBest_History*)
Best←*Best_History*(position, :)

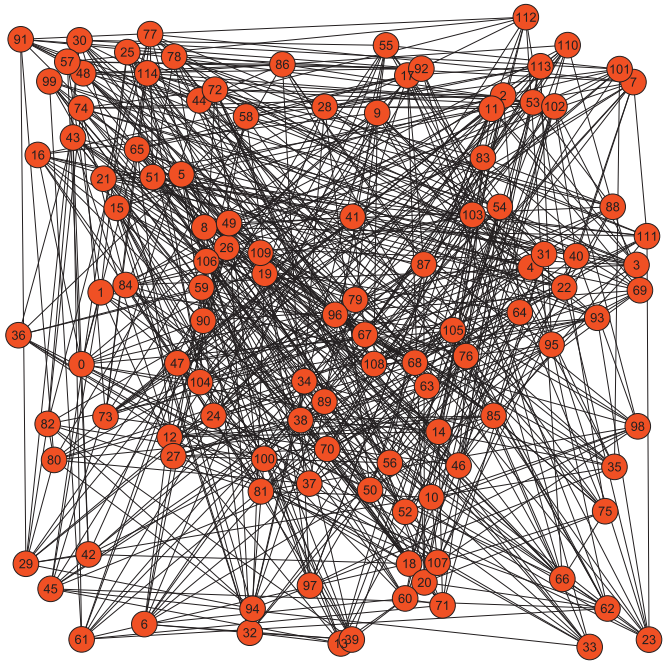
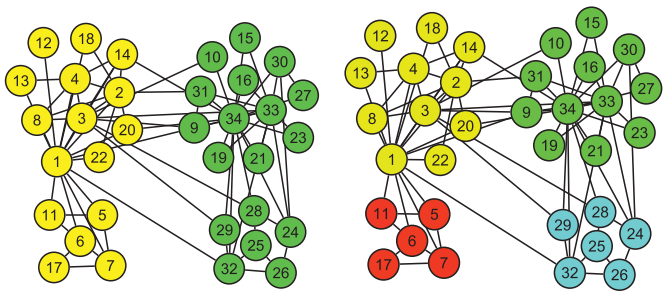


Fig. 9. Original College Football network.



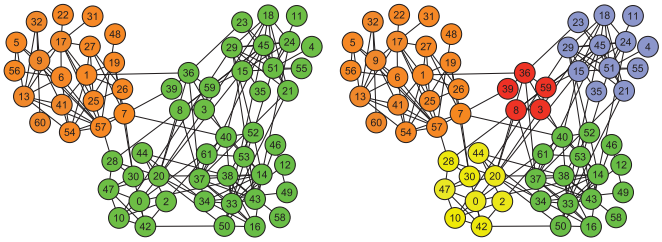
(a) NMI=1 (b) $Q = Q_{best}$

Fig. 10. Community structure of Karate network.

comparing its label with all its neighbors. For node i, it requires k_i comparisons. Therefore, it need Σk_i comparisons for all n nodes. Since $\Sigma k_i = 2m$, the vertex substitute transformation requires $O(m)$ comparisons. Similarly, based on Algorithm 3, the time complexity of community substitute transformation is $O(m)$. Besides, the complexity of state initialization is $O(n)$. In summary, the Initialization and Evolution step need $O(popsize \cdot (n + iterations \cdot 2m))$ comparisons; as m is larger than n in general, so it can be simply written as $O(popsize \cdot iterations \cdot m)$. The Generate Elite Population step and the Local Search step need $O(MaxIter \cdot n)$ comparisons. In a word, the computational complexity of MDSTA is $O(m)$.

Table 1
Network parameters.

Real-world network	Community	Node	Edge
Karate	2	34	78
Dolphins	2	62	159
Polbooks	3	105	441
Football	12	115	613
Jazz	Unknown	198	2742
E-mail	Unknown	1133	5451
Netscience	Unknown	1589	2742
Power grid	Unknown	4941	6594



(a) NMI=1 (b) $Q = Q_{best}$

Fig. 11. Community structure of Dolphins network.

4. Experiment analysis

In order to evaluate the performance of MDSTA in finding suitable community structure in networks, several experiments in both artificial networks and real-world networks are conducted. In addition, we compare MDSTA with some state-of-art methods including: Louvain [34], GN [35], LPA [4], Danon [36], GA-Net [37], MOGA-Net [38]. The parameter settings for each algorithm are based on their respective papers. As for MDSTA, its parameters will be shown in Section 4.4. The experiments are conducted in MATLAB 2017b Windows10 based on a personal computer with Intel i5-6500 @3.2 GHz CPU and 8 GB RAM.

4.1. Performance measure

Similar with clustering, the results of community detection need to be measured and quantified to determine the performance of different algorithms. There are two main indexes which could evaluate the results of network partitions, including the modularity (denoted as Q) and the normalized mutual information (denoted as NMI).

For the modularity index, we consider a partition with higher value of Q that is superior than other partitions and it does not need to know the real community structure of the network. The detail description of modularity is given in Eq. (3). Furthermore, a modularity value greater than 0 indicates that community structure begins to emerge in the network, and when the modularity value is higher than 0.3, we think that the effect of community division is good.

As for the normalized mutual information index, the calculation formula is given as follows

$$NMI = \frac{-2\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} N_{ij}\log\frac{N_{ij}n}{N_i N_j}}{\sum_{i=1}^{N_A} N_{i.}\log\frac{N_{i.}}{n} + \sum_{j=1}^{N_B} N_{.j}\log\frac{N_{.j}}{n}} \tag{7}$$

where $A(B)$ denotes the community partition which includes $N_A(N_B)$ communities in a network with n nodes. Community i in the actual partition and community j in the real partition have N_{ij} nodes in common, and then N_{ij} forms a matrix N. Meanwhile, $N_{i.}$ ($N_{.j}$) represent the sum of the ith row (jth colum) of matrix N.

The NMI index assesses the differences between the community division detected by the algorithm and the real community division of the network. A high NMI value reflects that the detected community structure is very similar to the real community structure. Particularly, NMI(A,B)=0 when A and B are completely different. If network partition A and B are exactly the same, then NMI(A,B)=1.

4.2. Artificial networks

We choose the extented Girvan–Newman benchmark network to perform an artificial networks test which is one of the most commonly used analog data set in current community detection research [39]. The extented GN network mainly includes the following parameters: n represents the number of nodes; k represents the average degree of nodes in the network; $k_{\max}$ represents the maximum degree of the node; β represents the parameter of community size distribution; $c_{\max}$ ($c_{\min}$) represents the number of nodes contained in the largest (smallest) community; μ is a mixed parameter, indicating the probability of the node connecting with the outside of the community, and the larger the value of μ, the more difficult the community detection will be. Network parameters are set to n=128, k=16, $k_{\max}$=16, $c_{\min}$=32, $c_{\max}$=32, β=1. Experimental data including 11 different types of networks which are generated by setting different μ parameters by increasing 0.05 from 0 to 0.5.

The extented GN network has a known real community structure, and the NMI indicator can be used to evaluate the quality of community partitions obtained by various algorithms. All the algorithms are conducted by 30 times and the average NMI values are recorded in Fig. 8. From the line chart, it can be seen that only MDSTA, GN and Louvain could detect the exactly real network partition when $\mu = 0.4$. The community structure is rather complex after $\mu > 0.4$, so it is very difficult to find the true division of the network. However, MDSTA is still able to detect the community structure and it demonstrates the feasibility and effectiveness for the community detection problem.

4.3. Real-world networks

In this section, the performance of MDSTA is verified by experiments in eight real-word networks including Zackarys Karate Club [40], Bottlenose Dolphins [41], American College Football [2], Krebs Political books [9], Jazz Musicians [42], Netscience [43], E-mail [44], and Power grid [45]. Different from the artificial networks, all nodes and edges in a real-word network have their own practical meanings. Here is an example of the practical meaning of a real network:

College Football network Newman created a complex social network based on the American college football League. The network contains 115 nodes and 616 edges, the original network can be seen in Fig. 9. The nodes in the network represent the football teams. The edge between the two nodes represents there is a match between the two teams. The 115 participating college teams are divided into 12 leagues. The process of the game is that the teams within the league perform the first group match, and then the teams between the leagues. This shows that the number of matches between teams within the league is more than the number of matches between teams in different leagues. The league can be expressed as the community of the college football network.

The basic parameters of these networks are given in Table 1. Furthermore, in order to visually see the actual effect of the algorithm, illustrations are given including the real community structure and the community structure discovered by MDSTA both in the Karate network and the Dolphins network in Figs. 10 and 11. According to the illustrations, the ground truth division divided more roughly than the community division which is detected by

Table 2
Experimental results for the networks with known ground truth.

Network	Algorithm	MDSTA	Louvain	GN	LPA	Danon	GA-Net	MOGA-Net
Karate	Q_{best}	**0.4198**	0.4188	0.4013	0.4020	0.4087	**0.4198**	0.4172
	Q_{mean}	**0.4198**	0.4188	0.4013	0.3157	0.4062	0.4123	0.4121
	Q_{worst}	**0.4198**	0.4188	0.4013	6.4051e-18	0.4033	0.3929	0.3872
	NMI_{mean}	0.6873	0.5866	0.5603	0.6253	0.5389	0.6881	**0.9580**
Dophins	Q_{best}	**0.5285**	0.5188	0.5194	0.5265	0.5136	0.5187	0.5227
	Q_{mean}	**0.5284**	0.5188	0.5194	0.4850	0.5136	0.4677	0.4742
	Q_{worst}	**0.5276**	0.5188	0.5194	0.3787	0.5136	0.4116	0.4110
	NMI_{mean}	0.5872	0.5162	0.5542	0.6592	0.5743	0.4830	**0.8236**
Polbooks	Q_{best}	**0.5272**	0.4986	0.5168	0.4986	0.5269	0.5223	0.5248
	Q_{mean}	**0.5272**	0.4986	0.5168	0.4239	0.5250	0.5229	0.5057
	Q_{worst}	**0.5272**	0.4986	0.5168	1.7874e-17	0.5237	0.4785	0.4725
	NMI_{mean}	0.5603	**0.5745**	0.5585	0.4863	0.5429	0.5099	0.5637
Football	Q_{best}	**0.6046**	**0.6046**	0.5996	0.6032	0.5773	0.5935	0.5477
	Q_{mean}	**0.6046**	**0.6046**	0.5996	0.5812	0.5705	0.5307	0.4713
	Q_{worst}	0.6044	**0.6046**	0.5996	0.5496	0.5580	0.2187	0.3770
	NMI_{mean}	0.8892	**0.8903**	0.8735	0.8681	0.7534	0.7742	0.7231

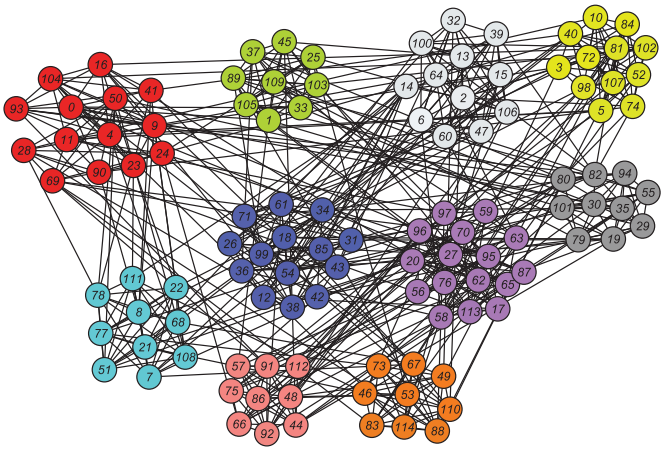


Fig. 12. Structure of College Football Network finding by MDSTA.

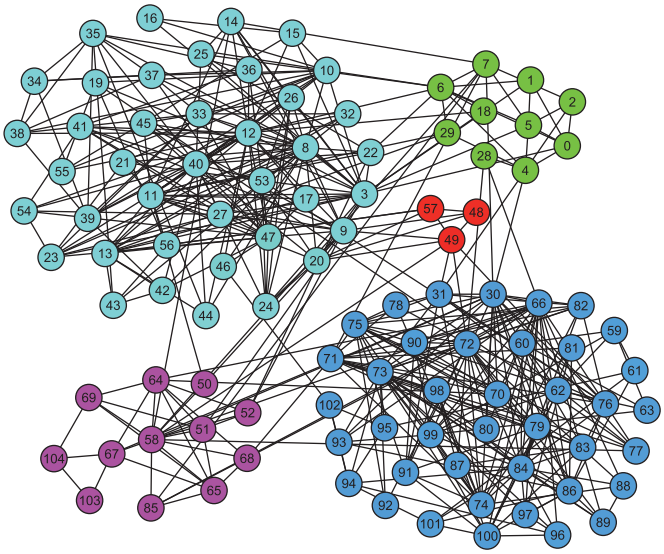


Fig. 13. Structure of Polbooks Network finding by MDSTA.

our algorithm because smaller communities can be detected in our method.

In addition, Table 2 shows the experimental results by running independently 30 times for each algorithm. By analyzing the data in the table, the modularity values obtained by LPA, Danon, GA-Net and MOGA-Net are unstable and often fall into local optimal. While, Louvain and GN get the same results in each network, but they couldn't obtain the highest Q values. However, it should be highlight that MDSTA not only can get the highest modularity values but also has a very stable performance among these several state-of-art algorithms. Therefore, the experimental results precisely indicate that the proposed algorithm has powerful global search capabilities to solve this problem.

As for Polbooks network and Football network, the community structure detected by MDSTA is not completely consistent with the real community division as exhibited in Figs. 12 and 13. However, according to the data presented in Table 2, MDSTA is the most effective algorithm, because it can obtain the highest modularity values and relative high NMI values. So, it can be verified that our algorithm can find a meaningful result.

In the Jazz Musicians network, its real community structure is not given, so that we can only compute the modularity values and make a comparison. According to Table 3, it can be noted that MDSTA, Louvain and LPA can reach high modularity values, but the performance of MDSTA is more stable. The main reasons can be summarized as two points. On the one hand, modularity is directly used as the fitness function in our method. On the other hand, we first use the state transformation operators for individual evolution to form an elite population, and then carry out group evolution among high quality individuals, and this strategy can fully utilize the global and local search ability of the algorithm.

Also, MDSTA is tested on three large-scale real-word networks, including the E-mail network, the Netscience network and the Power grid network. The number of nodes and edges of these networks are much larger than the networks studied above. Like the Jazz Musicians network, the real network partitions of the three networks are unknown. Thus, we obtain the modularity values of each algorithm for a comparison. Table 3 gives the experimental results.

The E-mail network is a complex network which indicates the email communications of a university. From Table 3, LPA algorithm gets poor performance, and the modularity values are too small and unstable. As for the two evolutionary algorithms: GA-Net and MOGA-Net, although they can obtain relatively stable network partitions, the modularity values are not good enough than other algorithms. Danon, Louvain and MDSTA can get good and stable modularity values among these algorithms. Meanwhile, MDSTA gets the highest modularity value of 0.5785 which can show the good performance of our algorithm.

The Netscience network studied the collaborate relationship among scientists. It contains 1589 nodes and 2742 edges. This

Table 3
Experimental results for the networks with unknown ground truth.

Network	Algorithm	MDSTA	Louvain	GN	LPA	Danon	GA-Net	MOGA-Net
Jazz	Q_{best}	**0.4451**	0.4431	0.4051	0.4428	0.4401	0.3926	0.3880
	Q_{mean}	**0.4449**	0.4431	0.4051	0.4353	0.4391	0.2971	0.3056
	Q_{worst}	**0.4445**	0.4431	0.4051	0.2816	0.4387	0.2248	0.2459
E-mail	Q_{best}	**0.5785**	0.5412	0.5323	0.2361	0.5472	0.3626	0.2485
	Q_{mean}	**0.5666**	0.5412	0.5323	0.0081	0.5424	0.3194	0.2076
	Q_{worst}	**0.5464**	0.5412	0.5323	1.1298e-16	0.5364	0.2341	0.2032
Netscience	Q_{best}	**0.9599**	0.9543	0.9579	0.9255	0.9588	0.9211	0.9125
	Q_{mean}	**0.9597**	0.9543	0.9579	0.9197	0.9585	0.8396	0.9101
	Q_{worst}	**0.9592**	0.9543	0.9579	0.9114	0.9583	0.8393	0.8990
Power grid	Q_{best}	**0.9376**	0.9335	0.9330	0.7532	0.9366	0.6763	0.7087
	Q_{mean}	0.9345	0.9335	0.9330	0.7471	**0.9354**	0.6691	0.7026
	Q_{worst}	**0.9340**	0.9335	0.9330	0.7408	0.9339	0.6607	0.6968

Table 4
Experimental results for different parameter configurations in three real-world networks.

Network	SE	Popsize				
		10	20	30	40	50
Dophins	10	0.5275 ± 7.3783e−4	0.5280 ± 6.5612e−4	0.5282 ± 4.8042e−4	0.5282 ± 5.2077e−4	0.5281 ± 4.7582e−4
	20	0.5278 ± 7.0640e−4	0.5282 ± 4.5793e−4	0.5282 ± 3.9937e−4	0.5282 ± 5.0587e−4	0.5282 ± 4.1388e−4
	30	0.5281 ± 6.6716e−4	0.5284 ± 4.0817e−4	0.5284 ± 3.8419e−4	0.5281 ± 4.6851e−4	0.5283 ± 3.7014e−4
	40	0.5280 ± 6.2089e−4	0.5284 ± 2.4145e−4	0.5283 ± 3.8380e−4	0.5284 ± 3.3897e−4	0.5283 ± 3.6304e−4
	50	0.5281 ± 4.3347e−4	0.5283 ± 3.7014e−4	0.5283 ± 3.7014e−4	0.5284 ± 3.0952e−4	0.5283 ± 3.5581e−4
Football	10	0.6041 ± 6.1384e−4	0.6044 ± 2.8263e−4	0.6044 ± 9.6788e−5	0.6045 ± 7.4653e−05	0.6045 ± 7.0788e−5
	20	0.6043 ± 4.0137e−4	0.6045 ± 8.1860e−5	0.6045 ± 6.7380e−5	0.6045 ± 5.6859e−05	0.6045 ± 5.2593e−5
	30	0.6045 ± 9.4583e−5	0.6046 ± 4.5321e−5	0.6046 ± 2.9638e−5	0.6046 ± 3.5791e−05	0.6046 ± 2.0616e−17
	40	0.6045 ± 6.0025e−5	0.6046 ± 2.9638e−5	0.6046 ± 2.9638e−5	0.6046 ± 2.9638e−05	0.6046 ± 0
	50	0.6045 ± 5.2593e−5	0.6046 ± 2.9638e−5	0.6046 ± 2.9156e−17	0.6046 ± 2.0616e−17	0.6046 ± 2.9156e−17
Jazz	10	0.4440 ± 5.8660e−4	0.4441 ± 3.5917e−4	0.4442 ± 5.0299e−4	0.4445 ± 2.6939e−4	0.4444 ± 2.4627e−4
	20	0.4445 ± 2.9234e−4	0.4446 ± 1.9396e−4	0.4447 ± 2.0828e−4	0.4448 ± 1.3454e−4	0.4448 ± 1.5392e−4
	30	0.4446 ± 2.6855e−4	0.4447 ± 1.7286e−4	0.4448 ± 1.6435e−4	0.4448 ± 1.5652e−4	0.4448 ± 1.8246e−4
	40	0.4448 ± 2.2437e−4	0.4448 ± 1.9897e−4	0.4448 ± 1.6270e−4	0.4449 ± 2.3620e−4	0.4448 ± 2.2084e−4
	50	0.4447 ± 2.0623e−4	0.4448 ± 2.1589e−4	0.4448 ± 2.0628e−4	0.4448 ± 2.4621e−4	0.4448 ± 2.5130e−4
network	MaxIter	*elite_popsize*				
without local search		10	20	30	40	50
Dophins	10	0.5281 ± 5.7061e−4	0.5281 ± 4.6712e−4	0.5280 ± 6.4156e−4	0.5281 ± 5.0468e−4	0.5280 ± 5.5991e−4
	20	0.5283 ± 3.8197e−4	0.5283 ± 3.7014e−4	0.5282 ± 4.7833e−4	0.5281 ± 4.8611e−4	0.5279 ± 5.5723e−4
	30	0.5280 ± 6.1123e−4	0.5282 ± 5.2498e−4	0.5282 ± 5.2469e−4	0.5282 ± 4.1929e−4	0.5282 ± 4.2720e−4
0.5278 ± 6.0440e-04	40	0.5280 ± 4.7906e−4	0.5284 ± 3.0950e−4	0.5283 ± 3.8380e−4	0.5283 ± 3.6304e−4	0.5281 ± 5.2972e−4
	50	0.5282 ± 5.5839e−4	0.5284 ± 3.3897e−4	0.5283 ± 4.3726e−4	0.5282 ± 4.2965e−4	0.5281 ± 4.6979e−4
Football	10	0.6045 ± 6.1903e−5	0.6045 ± 6.4699e−5	0.6045 ± 7.0788e−5	0.6045 ± 5.2593e−5	0.6045 ± 8.4882e−5
	20	0.6046 ± 3.5791e−5	0.6046 ± 3.8588e−5	0.6045 ± 5.8464e−5	0.6045 ± 5.4397e−5	0.6046 ± 5.0725e−5
	30	0.6046 ± 4.1183e−5	0.6046 ± 4.7476e−5	0.6046 ± 3.5791e−5	0.6046 ± 2.5751e−5	0.6045 ± 6.1903e−5
0.6043 ± 3.8822e-04	40	0.6046 ± 3.8588e−5	0.6046 ± 2.9638e−5	0.6046 ± 2.5751e−5	0.6046 ± 3.5791e−5	0.6046 ± 4.7476e−5
	50	0.6046 ± 3.8588e−5	0.6046 ± 3.5791e−5	0.6046 ± 2.0616e−17	0.6046 ± 2.9156e−17	0.6046 ± 4.1183e−5
Jazz	10	0.4447 ± 2.2318e−4	0.4447 ± 1.9726e−4	0.4447 ± 2.5332e−4	0.4446 ± 2.0325e−4	0.4447 ± 2.4037e−4
	20	0.4448 ± 2.1884e−4	0.4447 ± 1.6565e−4	0.4448 ± 1.9598e−4	0.4447 ± 2.1427e−4	0.4447 ± 1.9887e−4
	30	0.4448 ± 2.1872e−4	0.4448 ± 1.9632e−4	0.4447 ± 1.9522e−4	0.4448 ± 1.7521e−4	0.4447 ± 1.9458e−4
0.4445 ± 1.9911e−04	40	0.4448 ± 1.7669e−4	0.4448 ± 1.8097e−4	0.4447 ± 1.6444e−4	0.4448 ± 1.4206e−4	0.4447 ± 1.9005e−4
	50	0.4448 ± 2.0681e−4	0.4448 ± 1.8706e−4	0.4448 ± 1.5588e−4	0.4448 ± 1.2942e−4	0.4448 ± 2.0950e−4

± denotes "mean ± standard deviation".

network has an obvious community structure and all the algorithms can get high modularity values. For MDSTA, it obtains the highest modularity value of 0.9599. After 30 times experiments, the mean modularity value of MDSTA is 0.9597, which is almost the same with the highest value. According to the experimental results, MDSTA can find good network partitions efficiently.

The Power grid network was constructed based on the high-voltage power grid in the US. The network is very large which contains 4941 nodes and 6594 edges. In this network, the modularity values obtained by Louvain, GN, Danon and MDSTA are pretty high. What's more, the best results (0.9376) and the worst result (0.9340) of MDSTA are all larger than other algorithms. Based on the experimental results, it is clear that our algorithm possesses very competitive performance with other state-of-art algorithms.

4.4. Parameter analysis

In the Initialization and Evolution step of Algorithm 4, it includes three parameters: *popsize, stagnate_number* and *SE. Popsize* means the number of the population, while *SE* controls the search strength of state transformation operators. From Algorithm 4, it can be noted that *stagnate_number* is used as the termination condition. If both state transformation operators can not get a state with better fitness value for *stagnate_number* times, the two substitute transformation operations will be terminated. In the proposed MDSTA, *stagnate_number* is set as 10, which is empirically determined. As for *popsize* and *SE*, Table 4 records the experimental results over 30 times independent experiments in three real networks. Generally, from Table 4, we observe that with the values of *SE* and *popsize* increased, the mean of modularity values becomes larger and the stand deviation becomes smaller, which indicate good results.

Moreover, by comparing the change rate of modularity values per row and per column separately, it can be found that the impact of *SE* on the experimental results is greater than that of *popsize*. The main reason for this phenomenon is that *SE* controls the number of candidate solutions in each state transformation operator which is used for global search. In addition, small values of *SE* and *popsize* cannot benefit the algorithm's global exploration ability, whereas large values will cost a good deal of time. Considering both the global search ability and the computation time, *SE* and *popsize* are set as 40 and 20, respectively. Particularly, *popsize* is set as 1 when the number of network nodes is greater than 1000 to avoid consuming too much time.

In the local search procedure, two parameters are used to control the local exploitation ability which are denoted as *MaxIter* and *elite_popsize*. From the pseudo code of MDSTA in Algorithm 4, the *elite_popsize* means the size of the elite population, while the *MaxIter* gives the iterations of local search. Based on the definitions of these two parameters, the parameter comparison experiment is conducted, and the statistical results can be found in Table. 4. Firstly, it can be noted that the modularity value becomes larger after the local search procedure. So it can be concluded that the strategy of local search used in our algorithm is efficient to generate more promising states, and then it gives a possibility for MDSTA to jump out of the local optimal solution. Secondly, in general, the greater *SE*, the better the algorithm, while the *elite_popsize* does not have an obvious impact to the performance of the algorithm. On the one hand, small values of *elite_popsize* will cause a low diversity of solutions in the elite population. On the other hand, large *elite_popsize* values normally reduce the quality of the population. Therefore, considering the above analysis and the experimental results, *MaxIter* is set as 40 and *elite_popsize* is set as 20 in the proposed MDSTA.

5. Conclusion

In this paper, a new discrete state transition algorithm based on the concept of modularity called MDSTA is designed for the community detection problem. Different from other evolutionary algorithms, state transition algorithm is a flexible algorithm that we can design operators based on the specific problem. Thus, according to the topology of the network, two kinds of operators named as vertex substitute operator and community substitute operator are respectively proposed. Furthermore, in order to avoid falling into the local optimal solution, we performed the two-way crossover operation to the elite individuals which were optimized by the two substitute operators. Finally, we compared the proposed method with other six state-of-art community detection algorithms by testing them both in the artificial and real-world networks. The experimental results show that our algorithm is very promising and effective to solve the community detection problem.

Acknowledgement

This study was supported by the National Natural Science Foundation of China (Grant nos. 61873285, 61533021, 61621062, 61725306), the Innovation-Driven Plan in Central South University (Grant no. 2018CX12), the 111 Project (Grant No. B17048), the Hunan Provincial Natural Science Foundation of China (Grant no. 2018JJ3683), and the National Priority Research Project NPRP 9 166-1-031, funded by Qatar National Research Fund.

References

[1] S. He, G. Jia, Z. Zhu, D.A. Tennant, Q. Huang, K. Tang, J. Liu, M. Musolesi, J.K. Heath, X. Yao, Cooperative co-evolutionary module identification with application to cancer disease module discovery, IEEE Trans. Evolut. Comput. 20 (6) (2016) 874–891.
[2] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[3] J. Chen, Y. Saad, Dense subgraph extraction with application to community detection, IEEE Trans. Knowl. Data Eng. 24 (7) (2012) 1216–1230.
[4] U.N. Raghavan, R. Albert, S.R.T. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (3) (2007) 036106.
[5] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Phys. Rev. E 80 (2) (2009) 026129.
[6] L. Subelj, M. Bajec, Unfolding communities in large complex networks: combining defensive and offensive label propagation for core extraction., Phys. Rev. E 83 (3) (2011) 036103.
[7] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization., Phys. Rev. E 72 (2) (2005) 027104.
[8] M.E. Newman, Fast algorithm for detecting community structure in networks., Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 69 (6 Pt 2) (2004) 066133.
[9] M.E.J. Newman, Modularity and community structure in networks., Proc. Natl. Acad. Sci. USA. 103 (23) (2006) 8577–8582.
[10] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA. 105 (4) (2008) 1118–1123.
[11] M. Rosvall, C.T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, Proc. Natl. Acad. Sci. USA. 104 (18) (2007) 7327–7331.
[12] B. Hajek, Y. Wu, J. Xu, Information limits for recovering a hidden community, IEEE Trans. Inf. Theory 63 (8) (2017) 4729–4745.
[13] Z. Bu, H.-J. Li, J. Cao, Z. Wang, G. Gao, Dynamic cluster formation game for attributed graph clustering, IEEE Trans. Cybern. (99) (2017) 1–14.
[14] H.-J. Li, Z. Bu, A. Li, Z. Liu, Y. Shi, Fast and accurate mining the community structure: integrating center locating and membership optimization, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2349–2362.
[15] M. Gong, Q. Cai, X. Chen, L. Ma, Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition, IEEE Trans. Evolut. Comput. 18 (1) (2014) 82–97.
[16] Q. Cai, M. Gong, L. Ma, L. Jiao, A novel clonal selection algorithm for community detection in complex networks, Comput. Intell. 31 (3) (2015) 442–464.
[17] A. Song, M. Li, X. Ding, W. Cao, K. Pu, Community detection using discrete bat algorithm, IAENG Int. J. Comput. Sci. 43 (1) (2016) 37–43.
[18] Z. Li, J. Liu, A multi-agent genetic algorithm for community detection in complex networks, Phys. A Stat. Mech. Appl. 449 (2016) 336–347.
[19] M. Guerrero, F.G. Montoya, R. Banos, A. Alcayde, C. Gil, Adaptive community detection in complex networks using genetic algorithms, Neurocomputing 266 (2017) 101–113.
[20] X. Zhou, C. Yang, W. Gui, State transition algorithm, J. Ind. Manag. Optim. 8 (4) (2012) 1039–1056.
[21] X. Zhou, J. Zhou, C. Yang, W. Gui, Set-point tracking and multi-objective optimization-based Pid control for the goethite process, IEEE Access 6 (2018) 36683–36698.
[22] X. Zhou, C. Yang, W. Gui, A statistical study on parameter selection of operators in continuous state transition algorithm, IEEE Trans. Cybern. 99 (2018).
[23] X. Zhou, P. Shi, C. Lim, C. Yang, W. Gui, A dynamic state transition algorithm with application to sensor network localization, Neurocomputing 273 (2018) 237–250.
[24] J. Han, C. Yang, X. Zhou, W. Gui, A new multi-threshold image segmentation approach using state transition algorithm, Appl. Math. Model. 44 (2017) 588–601.
[25] F. Zhang, C. Yang, X. Zhou, W. Gui, Fractional-order Pid controller tuning using continuous state transition algorithm, Neural Comput. Appl. 29 (10) (2018) 795–804.
[26] J. Han, C. Yang, X. Zhou, W. Gui, A two-stage state transition algorithm for constrained engineering optimization problems, Int. J. Control Autom. Syst. 16 (2) (2018) 522–534.
[27] M. Huang, X. Zhou, T. Huang, C. Yang, W. Gui, Dynamic optimization based on state transition algorithm for copper removal process, Neural Computing and Applications (2017) 1–13.
[28] Z. Huang, C. Yang, X. Zhou, W. Gui, A novel cognitively inspired state transition algorithm for solving the linear bi-level programming problem, Cognit. Comput. (2018) 1–11.
[29] X. Zhou, D.Y. Gao, C. Yang, W. Gui, Discrete state transition algorithm for unconstrained integer optimization problems, Neurocomputing 173 (2016) 864–874.
[30] Fortunato, Santo, Community detection in graphs, Phys. Rep. 486 (3) (2009) 75–174.
[31] C. Pizzuti, Evolutionary computation for community detection in networks: a review, IEEE Trans. Evolut. Comput. (2017). 1–1
[32] M. Gong, Q. Cai, Y. Li, J. Ma, An improved memetic algorithm for community detection in complex networks, in: Proceedings of the Evolutionary Computation, 2012, pp. 1–8.
[33] X. Zhou, D.Y. Gao, C. Yang, W. Gui, Discrete state transition algorithm for unconstrained integer optimization problems, Neurocomputing 173 (P3) (2012) 864–874.
[34] V.D. Blondel, J. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) 10008.

[35] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[36] L. Danon, A. Diazguilera, A. Arenas, The effect of size heterogeneity on community identification in complex networks, J. Stat. Mech. Theory Exp. 2006 (11) (2006) 11010.
[37] C. Pizzuti, GA-Net: A genetic algorithm for community detection in social networks, in: Proc. of the 10th International Conference on Parallel Problem Solving from Nature (PPSN 2008), Springer, 2008, pp. 1081–1090.
[38] C. Pizzuti, A multiobjective genetic algorithm to find communities in complex networks, IEEE Trans. Evolut. Comput. 16 (3) (2012) 418–430.
[39] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110.
[40] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[41] D. Lusseau, The emergent properties of a dolphin social network, Proc. R. Soc. Lond. B Biolog. Sci. 270 (2003) S186–S188. Suppl 2
[42] P.M. Gleiser, L. Danon, Community structure in jazz, Adv. Compl. Syst. 06 (04) (2003) 565–573.
[43] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (6) (2003) 065103.
[44] M.E. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[45] D.J. Watts, S.H. Strogatz, Collective dynamics of mall-worldetworks, Nature 393 (6684) (1998) 440.

Xiaojun Zhou received his Bachelor's degree in Automation in 2009 from Central South University, Changsha, China and received the Ph.D degree in Applied Mathematics in 2014 from Federation University Australia. He is currently an Associate Professor at Central South University, Changsha, China. His main interests include modeling, optimization and control of complex industrial process, optimization theory and algorithms, state transition algorithm, duality theory and their applications.

Ke Yang received the B.S. degree in electrical engineering and automation from Hunan Institute of Engineering, Xiangtan, China, in 2017, where he is currently pursuing the M.S. degree from Central South University, Changsha, China. His current research interests include community detection, state transition algorithm, and data mining.

Yongfang Xie received the B.S., M.S., and Ph.D. degrees in control science and engineering from Central South University, Changsha, Hunan, China, in 1993, 1996, and 1999, respectively. From 1999 to 2003, he was with the Tokyo International Information Science Research Institute, Tokyo, Japan, and was also a Visiting Scholar with the PToPA Research Institute, Tokyo. He is currently a Full Professor with the School of Information.

Chunhua Yang received her M.Eng. in Automatic Control Engineering and her Ph.D. in Control Science and Engineering from Central South University, China in 1988 and 2002 respectively, and was with the Electrical Engineering Department, Katholieke Universiteit Leuven, Belgium from 1999 to 2001. She is currently a full professor in the School of Information Science & Engineering, Central South University. Her research interests include modeling and optimal control of complex industrial process, intelligent control system, and fault-tolerant computing of real-time systems.

Tingwen Huang received the B.S. degree from Southwest Normal University (currently, Southwest University), Chongqing, China, in 1990, the M.S. degree from Sichuan University, Chengdu, China, in 1993, and the Ph.D. degree from Texas A&M University, College Station, TX, USA, in 2002. He is a Professor with Texas A&M University at Qatar, Doha, Qatar. He has published over 70 peer reviewed journal papers. He has edited a book entitled *Advances in Intelligent and Soft Computing* (Springer). His research on chaotic dynamical systems received Qatar National Priority Research Project support from Qatar Research Fund. His current research interests include chaotic dynamical systems, neural networks, optimization and control, and traveling wave phenomena. Dr. Huang is one of the Editors for the five volumes of the Proceedings of the 19th International Conference on Neural Information Processing published in Lecture Notes in Computer Science by Springer. He currently serves as an Editorial Board Member for four international journals, such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Cognitive Computation, Advances in Artificial Neural Systems, and Intelligent Control and Automation.*