Physica A 503 (2018) 762–779

Contents lists available at ScienceDirect

Physica A

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

Convergence improvement of differential evolution for community detection in complex networks

Jing Xiao [a,b,c], Yong-Jian Zhang [a], Xiao-Ke Xu [a,b,*]

[a] *College of Information and Communication Engineering, Dalian Minzu University, Dalian 116600, China*
[b] *Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China*
[c] *College of Automation, Harbin Engineering University, Harbin 150001, China*

HIGHLIGHTS

- Two key factors affecting the convergence performance of EA-based modularity optimization algorithm are summarized.
- A series of effective measures are designed to improve the global convergence ability of differential evolution algorithm.
- The performance of the proposed algorithm is validated by various networks and compared with several representative algorithms.
- The new algorithm can detect communities with high accuracy and stability.

ARTICLE INFO

Article history:
Received 16 October 2017
Received in revised form 14 January 2018
Available online 21 February 2018

Keywords:
Complex networks
Community detection
Modularity optimization
Differential evolution
Convergence performance

ABSTRACT

To improve the quality of optimal partitions obtained by modularity optimization in community detection of complex networks, we summarize two key factors determining the convergence performance of the EA-based (Evolutionary Algorithm) optimization algorithms and present a new Classification-based Differential Evolution algorithm for Modularity Optimization (CDEMO). On the one hand, CDEMO redesigns the main evolutionary operators of the standard Differential Evolution (DE), including the mutation, parameter adjustment and selection strategy, to improve the global convergence ability of the optimization strategy, which is often been overlooked in available EA-based modularity optimization algorithms. On the other hand, CDEMO improves the community modification method to better utilize the known topology information of networks, which reduces the search space of DE and ensures adequate space for the global optimum at the same time. The performance of CDEMO is evaluated on both artificial computer-generated and real-world social networks, and experimental results prove the validity of the improvement measures and the superiority of CDEMO over several existing state-of-the-art modularity optimization algorithms.

1. Introduction

Community detection of complex networks can be seen as the process of identifying meaningful modules or hierarchical structures from networks, which enables us to gain insights into the topological features, functional properties and dynamic characteristics of complex networks. Based on the detected community structure, some important information, such as the attribute relevancy of communities and the behavior of networks, can be revealed in depth [1–6]. In recent years, the

* Corresponding author at: College of Information and Communication Engineering, Dalian Minzu University, Dalian 116600, China.
E-mail address: xuxiaoke@foxmail.com (X.-K. Xu).

https://doi.org/10.1016/j.physa.2018.02.072
0378-4371/© 2018 Published by Elsevier B.V.

study of community detection of complex networks has attracted more and more research interests from several different scientific fields. Besides, its practical application has also become increasingly significant, including on-line retail product recommendation, political election prediction, social computing, and so on.

In the past few years, many kinds of methods have been proposed for community detection, among them the most popular one is the modularity maximization method based on the definition of modularity. However, modularity maximization is essentially a Nondeterministic Polynomial (NP) hard problem [1,2]. Therefore, traditional deterministic optimization algorithms, such as the mathematical programming [2], greedy algorithm [3], spectral analysis [4], and extremal optimization [5], usually suffer from problems of premature convergence or stagnation due to their poor convergence abilities. In addition, as real-world networks grow in scale and fuzziness, the extreme degeneracy problem becomes more serious, which means that the global optimal partition of communities becomes more difficult to be find among exponentially growing number of suboptimal but competitive alternatives. As a result, the accuracy and stability of the detected community structures are seriously impacted.

Recently, stochastic optimization algorithms, especially the Evolutionary Algorithms (EAs), have been successfully employed to optimize modularity, such as Genetic Algorithm (GA) [7–12], Particle Swarm Optimization (PSO) [13–15], memetic algorithm [16,17], ant colony optimization [18–20], clone selection [21,22] and Differential Evolution (DE) [23,24,2,25]. Especially, EA-based modularity optimization algorithms have been demonstrated to be superior to others in many cases [1,2], which is mainly due to their strong global optimization abilities. What is more, EA-based algorithms do not need any specific mathematic model or prior knowledge (e.g., the number of communities), which is very difficult to be obtained in real-world networks. However, although these algorithms have achieved relatively satisfactory results on several networks, the problems of premature convergence and extreme degeneracy have not been fully addressed.

In order to overcome these problems and improve quality of optimal partitions of communities, the convergence performance of EA-based modularity optimization algorithms should be further enhanced. Experimental results have shown that, the convergence ability of EA-based algorithms is mainly determined by two key factors. The first and the most important factor is how to improve the global convergence ability of EAs themselves. While the other factor is how to efficiently utilize the topological information of networks to reduce the huge search space in the optimization. However, to the best of our knowledge, for most of available algorithms, original basic EAs have often been used directly as optimization strategies and the convergence ability of EAs has often been overlooked, which usually results in premature convergence of EAs and poor quality of optimal partitions. At the same time, though existing algorithms have paid much attention on the second key factor, and evolutionary operators in EAs have often been redesigned by incorporating some topology information to meet the requirement of community detection, the known topological information of networks are usually inappropriately used, which destroys the search space for the global optimal community partition.

To address the above-mentioned problems, in this study we propose a novel Classification-based Differential Evolution algorithm for Modularity Optimization (CDEMO). In view of that DE is one of the most efficient evolutionary optimization algorithm, and has been proved very efficient in handling the modularity optimization problem, CDEMO adopts DE as its optimization strategy and follows the basic framework of the DE-based modularity optimization widely used in [2,23,24]. To enhance the convergence ability of DE in the modularity optimization and improve the qualities of final community partitions, a set of effective measures are designed and utilized in CDEMO, in which both of the two key factors mentioned above are considered. On the one hand, to improve the global convergence ability of DE, main evolutionary operators in the original basic DE are redesigned, including a classification-based self-adaptive mutation strategy, a dynamic self-adaptive parameter adjustment strategy, and a selection operation based on the historical information. On the other hand, in order to better utilize the known topology information, an improved community modification strategy based on the neighborhood information is proposed, which is used to reduce the search space of DE and ensure adequate search space for the global optimal partition at the same time.

Compared with existing EA-based modularity optimization algorithms, the main feature of CDEMO is its overall convergence performance improvement from both of the two aspects. In our study, the new designed DE algorithm is tested on 18 benchmark functions, and CDEMO is evaluated on both artificial computer-generated and real-world social networks. Experimental results prove that, the convergence ability of DE is greatly improved in terms of the solution precision and robustness, which can significantly enhance its performance in the modularity optimization and the quality of the detected communities. As a result, CDEMO has a very competitive performance compared with many state-of-the-art module identification algorithms.

The rest of this paper is organized as follows. In Section 2, we introduce and verify the new designed classification-based self-adaptive DE algorithm. In Section 3, we explain and test the new DE-based modularity optimization algorithm CDEMO. In Section 4, the performance of CDEMO is evaluated and discussed on both artificial and real-world complex networks, compared with several existing state-of-the-art modularity optimization algorithms. Finally, conclusions are drawn in Section 5.

2. Classification-based self-adaptive differential evolution

To enhance qualities of optimal partitions obtained by CDEMO, a set of measures are designed and incorporated in the original basic DE algorithm to improve its global convergence ability. In the improved DE, three main evolutionary operators are redesigned, including a classification-based self-adaptive mutation strategy, a dynamic self-adaptive parameter adjustment strategy and a selection operation based on the historical information.

2.1. Classification-based self-adaptive DE mutation

Apart from few prior knowledge of networks, EA-based modularity optimization algorithms rely entirely on their global convergence ability to identify the optimal partition from a huge search space. Therefore, the convergence performance of EA has a major impact on the accuracy and stability of the final partition. Moreover, mutation is the most important evolutionary operator in DE, which utilizes the difference between population individuals to promote information interaction and guide the search direction of the whole population towards the global optimum. The performance of mutation is crucial to the quality of offspring individuals and the optimal solution. However, almost all the existing DE-based modularity optimization algorithms adopt the following original basic DE mutation strategy "rand/1/bin" to generate trial vectors [2,23–25], that is

$$\mathbf{v}_{i,t} = \mathbf{x}_{r3,t} + F \cdot \left(\mathbf{x}_{r1,t} - \mathbf{x}_{r2,t}\right). \tag{1}$$

In Eq. (1), $\mathbf{v}_{i,t}$ is the mutant vector generated corresponding to the ith target vector $\mathbf{x}_{i,t}$ in the population at generation t, where $i \in \{1, 2, \dots, NP\}$, and $t \in \{1, 2, \dots, t_{max}\}$. Here $\mathbf{x}_{r1,t}$, $\mathbf{x}_{r2,t}$ and $\mathbf{x}_{r3,t}$ are three different individuals randomly selected from population and satisfy $r1 \neq r2 \neq r3 \neq i$. The mutant factor F is usually a real number between 0 and 1. In "rand/1/bin", both of the basic vector $\mathbf{x}_{r3,t}$ and the difference vector $\mathbf{x}_{r1,t} - \mathbf{x}_{r2,t}$ are random component. Though random selection helps to keep the population diversity, it may also lead the target individual $\mathbf{x}_i$ deviate from the search direction towards the global optimum and result in premature convergence or expensive computational cost. Therefore, it is desirable to decrease the randomness in the mutation and intensify directional information for each target individual, guaranteeing the quality of the mutant vectors and speeding up convergence.

Based on the analyses above, a new classification-based self-adaptive DE mutation strategy is proposed in this section. Major improvement measures include:

1. The best-so-far solution $\mathbf{x}_{gbest,t}$ of the entire population and the best previous solution $\mathbf{x}_{pbesti,t}$ of each individual are utilized to guide the mutation direction, instead of randomly selected individuals;
2. a new self-adaptive classification mechanism is designed and utilized to balance the exploration and exploitation ability of individuals with different fitness characteristics;
3. variation extent of each individual is adjusted in a dynamic self-adaptive manner during the evolution by parameter adjustment.

Detailed operations of the new mutation strategy are described as follows.

For each target individual $\mathbf{x}_{i,t}$, if its fitness value f_i is larger than the average fitness value of all the individuals in the current population, it can be classified as an excellent individual, whose position in the search space is relatively close to the global optimum. Therefore, good genes in $\mathbf{x}_{i,t}$ should be retained to intensify its local search and the corresponding mutant vector $\mathbf{v}_{i,t}$ is generated as follows:

$$\mathbf{v}_{i,t} = F_{i,t} \cdot \mathbf{x}_{pbesti,t} + W_{i,t} \cdot \left(\mathbf{x}_{r2,t} - \mathbf{x}_{r3,t}\right), \tag{2}$$

where $\mathbf{x}_{pbesti,t}$ indicates the best previous vector of $\mathbf{x}_{i,t}$ before generation t, which is utilized to enhance the exploration ability of $\mathbf{x}_{i,t}$. $\mathbf{x}_{r2,t}$ and $\mathbf{x}_{r3,t}$ are two different individuals randomly selected from population and satisfy $r2 \neq r3 \neq i$. $F_{i,t}$ and $W_{i,t}$ are the control parameters of $\mathbf{x}_i$, which are dynamically adapted according to the fitness value of $\mathbf{x}_{i,t}$ as well as the number of generations (see Section 2.2).

For each target individual $\mathbf{x}_{i,t}$, if its fitness value f_i is less than the average fitness value of all the individuals in the population, it can be classified as a poor individual, whose position in the search space is relatively far away from the global optimum. Therefore, communication between it and excellent individuals in the population should be intensified to promote its global search and the corresponding mutant vector $\mathbf{v}_{i,t}$ is generated as follows:

$$\mathbf{v}_{i,t} = W_{i,t} \cdot \mathbf{x}_{r1,t} + K_{i,t} \cdot \left(\mathbf{x}_{gbest,t} - \mathbf{x}_{i,t}\right), \tag{3}$$

where $\mathbf{x}_{r1,t}$ is an individual randomly selected from population and satisfy $r1 \neq i$. $\mathbf{x}_{gbest,t}$ indicates the current best individual in the population at the tth generation, which is utilized to enhance the exploitation ability of $\mathbf{x}_{i,t}$. $W_{i,t}$ and $K_{i,t}$ are the control parameters of $\mathbf{x}_{i,t}$, which are dynamically adapted according to the fitness value of $\mathbf{x}_{i,t}$ as well as the number of generations (see Section 2.2).

The above classification-based self-adaptive mutation will be implemented on all of the population individuals in each generation till the end of the evolution, therefore the mutation of every individual can be adjusted pointedly. On the one hand, exploration of excellent individuals can be enhanced to increase the probability of finding the global optimum in their neighborhood. On the other hand, exploitation of poor individuals can also be strengthened to accelerate their variation towards the global optimum. To sum up, the evolutionary demands of individuals with different fitness characteristics can be better satisfied by the new mutation strategy. Under the guide of directional information, blindness in the search process can be efficiently reduced, and qualities of the offspring individuals and the optimal solution can be improved.

2.2. Dynamic self-adaptive parameter adjustment

In the above classification-based self-adaptive mutation strategy, there are three main control parameters W, K, F involved, corresponding to the random, social and cognitive part of the mutation equations respectively. Besides, there is also a key control parameter CR used in the crossover operation, determining the percentage of components in each trial vector $\mathbf{u}_{i,t}$ inherited from the mutation vector $\mathbf{v}_{i,t}$.

In order to adjust the variation extent of each individual in a dynamic self-adaptive manner during the evolution, a dynamic self-adaptive parameter adjustment strategy is also proposed. All of the parameters are adjusted according to both of the fitness characteristic of each individual and the number of generations. Since parameters W, K and F control the random part, social part and cognitive part of the mutation respectively, CR decides the inheritance from the mutation, their values could be adjusted following two principles:

1. Parameters are adjusted self-adaptively according to the fitness values of individuals. For poor individuals, variation extent in the mutation and crossover should be strengthened to introduce more directional information in the evolution. Therefore, both of the random and social parts in the mutation, and the inheritance in the crossover should be intensified, corresponding to large values of W and K in Eq. (3) as well as the CR in the crossover. On the contrary, for excellent individuals, cognitive part in the mutation should be reinforced and parameters should be adjusted following an opposite principle, corresponding to large value of F and small value of W in Eq. (2);
2. Parameters are updated dynamically according to the evolutionary generation at the same time. In the earlier stage of evolution, exploration ability of individuals should be strengthened to ensure sufficient search within their own neighborhood. On the contrary, in the later stage, exploitation ability of individuals should be enhanced to intensify communication among individuals and speed up the convergence of the whole population. According to this principle, the values of F, W, CR are gradually decreased during the evolution, while the value of K follows an opposite trend.

Through the aforementioned approaches, the evolution process of each individual and the balance between the exploration and exploitation could be controlled dynamically. Detailed operations can be described as follows:

$$W_{i,t} = W_{min} + (W_{max} - W_{min}) \times ((2 - exp(\frac{t}{t_{max}} \times ln2)) \times \frac{1}{2} + \frac{f_{max,t} - f_{i,t}}{f_{max,t} - f_{min,t}} \times \frac{1}{2}); \tag{4}$$

$$K_{i,t} = K_{min} + (K_{max} - K_{min}) \times ((exp(\frac{t}{t_{max}} \times ln2) - 1) \times \frac{1}{2} + \frac{f_{max,t} - f_{i,t}}{f_{max,t} - f_{min,t}} \times \frac{1}{2}); \tag{5}$$

$$F_{i,t} = F_{min} + (F_{max} - F_{min}) \times ((2 - exp(\frac{t}{t_{max}} \times ln2)) \times \frac{1}{2} + \frac{f_{i,t} - f_{min,t}}{f_{max,t} - f_{min,t}} \times \frac{1}{2}); \tag{6}$$

$$CR_{i,t} = CR_{min} + (CR_{max} - CR_{min}) \times ((2 - exp(\frac{t}{t_{max}} \times ln2)) \times \frac{1}{2} + \frac{f_{max,t} - f_{i,t}}{f_{max,t} - f_{min,t}} \times \frac{1}{2}). \tag{7}$$

The variation of all these parameters considers the influence of evolution generations and fitness values equally. For example, in Eq. (6), $F_{i,t}$ varies within the range of values $[F_{min}, F_{max}]$. With the increasing of iterations, $F_{i,t}$ decreases during the evolution (the exponent and logarithm makes a much smoother change), corresponding to a transition from exploration to exploitation. At the same time, if fitness value of the ith individual at the tth generation is relatively poor ($f_{i,t}$ is small in an maximization problem), then the value of parameter $F_{i,t}$ will be relatively small, since the maximum and minimum fitness values of all the population individuals ($f_{max,t}$ and $f_{min,t}$) are fixed at one generation. Other parameters are designed following the similar principle.

2.3. DE selection based on historical information

Selection is the last key evolutionary operation in the basic DE apart from the mutation and crossover operations. It is used to select the better individual between each target vector $\mathbf{x}_{i,t}$ and the corresponding trial vector $\mathbf{u}_{i,t}$, and the better one will be saved into the next generation. However, this traditional selection operation often miss several competitive solutions during the evolution and thus lower the quality of offspring individuals.

Our previous experimental results show that individuals with good fitness values may be generated in different evolutionary stages, including the initialization, mutation, crossover, and so on [26,27]. For a randomly selected individual $\mathbf{x}_{i,t}$, its mutant vector $\mathbf{v}_{i,t}$ is often superior to the trial vector $\mathbf{u}_{i,t}$ especially at the initial stage of evolution due to the greater variation extent. However, such competitive solutions may be destroyed by the subsequent evolutionary operations, therefore some potential areas in the search space for the global optimum may be omitted.

To overcome the problem, a new DE selection operation based on the historical information is proposed. In the new selection, competitive solutions generated during the whole evolutionary process will be saved as historical information and be utilized in the following evolutionary operations. This will be achieved by introducing a special population *pbest_pop* constructed by the historical optimal solution $\mathbf{x}_{pbesti,t}$ of each individual. The *pbest_pop* is generated in the initialization stage

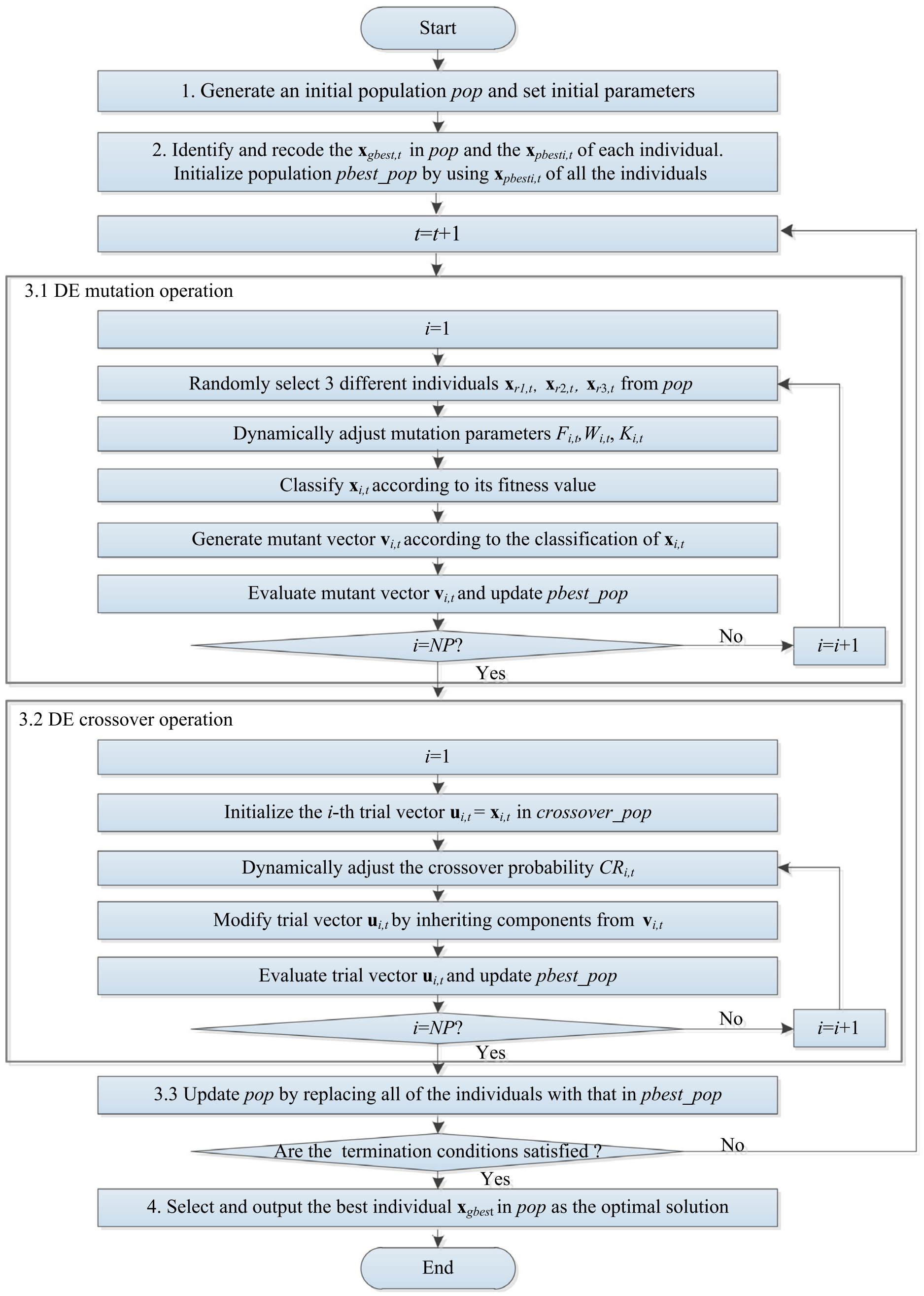
Start
1. Generate an initial population pop and set initial parameters
2. Identify and recode the x_gbest,t in pop and the x_pbesti,t of each individual. Initialize population pbest_pop by using x_pbesti,t of all the individuals
t=t+1
3.1 DE mutation operation
i=1
Randomly select 3 different individuals x_r1,t, x_r2,t, x_r3,t from pop
Dynamically adjust mutation parameters F_i,t, W_i,t, K_i,t
Classify x_i,t according to its fitness value
Generate mutant vector v_i,t according to the classification of x_i,t
Evaluate mutant vector v_i,t and update pbest_pop
i=NP?
No
i=i+1
Yes
3.2 DE crossover operation
i=1
Initialize the i-th trial vector u_i,t = x_i,t in crossover_pop
Dynamically adjust the crossover probability CR_i,t
Modify trial vector u_i,t by inheriting components from v_i,t
Evaluate trial vector u_i,t and update pbest_pop
i=NP?
No
i=i+1
Yes
3.3 Update pop by replacing all of the individuals with that in pbest_pop
Are the termination conditions satisfied ?
No
Yes
4. Select and output the best individual x_gbest in pop as the optimal solution
End

Fig. 1. The flowchart of the classification-based self-adaptive differential evolution.

and then it will be updated after each evolutionary operation. For each individual $\mathbf{x}_{i,t}$ in the population, if its fitness value is improved during an operation, the new individual will be seemed as the current best historical solution of $\mathbf{x}_{i,t}$ and be saved into the *pbest_pop*. At the end of each generation, individuals in *pop* will be substituted by all of the members in *pbest_pop*, and the current best vector $\mathbf{x}_{gbest,t}$ will also be selected from the *pbest_pop*.

2.4. Convergence performance validation of improved DE

The above three improvement measures are all designed to enhance the global convergence ability of DE algorithm. The flowchart of the improved DE algorithm is given in Fig. 1. Different from the original basic DE, the mutation operation combines more directional information and individuals could be mutated more pointedly. In addition, the selection operation is not executed after the crossover operation, it is performed by keeping and refreshing the *pbest_pop* after each evolutionary operation.

In order to validate the above improvement measures on DE, we test the improved algorithm on 18 standard benchmark functions. f_1–f_5 are unimodal functions, f_6–f_{14} are basic multimodal functions, f_{15}–f_{16} are expanded functions, and f_{17}–f_{18} are composition functions. Table 1 provides benchmark function details.

The improved DE is compared with 4 competitive and widely used DE variants, including $DE/rand/2/dir$, $DE/rand/1/bin$, $DE/current\text{-}to\text{-}best/2/bin$ and $DE/best/1/bin$. For ease of comparison, DE with the new mutation and parameter adjustment strategies is named as $DE\_version1$, and DE incorporating all of the three improvement measures is named as $DE\_version2$.

In the experiment, populations for all of the algorithms over each problem instance are initialized with the same population size $NP = 100$, the same dimension $D = 30$, and the same termination criterion $Max\_FEs = 5.0e + 05$. In addition, parameters F and CR in all of the DE variants are updated in a self-adaptive manner as shown in Eq. (6) and Eq. (7). The associated value ranges are set as $W \in [0.1, 0.9]$, $K \in [0.3, 0.9]$, $F \in [0.3, 0.9]$ and $CR \in [0.1, 0.9]$.

Six kinds of DE variants are compared in terms of the solution precision and robustness. Experiment results are presented in Table 2, including the mean and standard deviation (within parentheses) of the best-of-run fitness values tested on 30 independent runs. The best solution in each case is shown in bold. From Table 2 we can see clearly that, $DE\_version1$ and $DE\_version2$ outperform the other 4 DE variants significantly on almost all of the test problems. $DE\_version2$ successfully convergences to the real global optimum on 50.0% of test functions and acquires the best solution on 88.9% of test functions. The above results prove that, the classification-based self-adaptive mutation strategy can efficiently enhance the quality of the offspring individuals and the accuracy of the final optimal solutions. In addition, compared with $DE\_version1$, there is a significant improvement of $DE\_version2$ in terms of the precision, which demonstrates that the global convergence ability of DE can be greatly enhanced by the new selection operation based on the historical information.

The above experiment results make us believe that the improvement approaches of DE proposed in this paper are successful and effective, which can efficiently enhance the convergence ability of the original standard DE algorithm. It provides us an efficient global optimization method for the modularity optimization problem in the community detection of complex networks.

3. Modularity optimization based on improved differential evolution

Based on the classification-based self-adaptive differential evolution algorithm described in Section 2, a new DE-based modularity optimization algorithm, named CDEMO, is proposed in this section. To enhance the quality of optimal partitions of communities, improvement measures are designed from both of the two key factors to improve the overall convergence ability of CDEMO. For one thing, the convergence performance of DE algorithm has been improved, providing an efficient optimization strategy for CDEMO. For another, a modified community modification strategy based on the neighborhood information is proposed in this section. It is designed to better utilize the known topology information and thus promote the convergence of CDEMO.

3.1. Neighborhood-based community modification

EA-based modularity optimization algorithms need to search the global optimal partition in huge search space. For a network with n nodes, the search space of possible partitions under the community identifier-based representation reaches n^n. Though DE is one of the most efficient evolutionary optimization algorithm, and our improved DE variant described previously has proved to be highly efficient in handling high-dimensional global optimization problem. However, it is still difficult for the DE-based modularity optimization algorithm to converge to the global optimal partition in the huge search space. The DE-based algorithm may get stuck in local optimum due to the contradiction between the stochastic search property of EAs and the limited computing resources of real community detection problems.

To overcome the drawback, topology information of complex networks is usually utilized as prior knowledge in the process of optimization, so as to reduce the huge search space of EAs and promote convergence. Basing on the idea that node and its neighbors in the network are more likely to belong to the same community, the general approach utilized in available algorithms is to measure the similarity of community membership between each node and its neighbors. For example, for a randomly selected node i from an population individual in EAs, if $D(i)$ is its degree, E is the set of edges, $C(i)$ and $C(j)$ are the communities containing node i and j respectively, then we can calculate a parameter $P(i)$, as shown in Eq. (8), which represents the probability that node i is not in the same community as its neighbors.

$$P(i) = \frac{\sum_{(i,j)\in E} \{j | C(i) \neq C(j)\}}{D(i)}, \tag{8}$$

If the probability $P(i)$ is larger than a predefined threshold value δ, it means that the similarity between node i and its neighbors is relatively low, and node i may probably be put into a wrong community. Therefore, the community membership of node i should be modified and it should be placed into a new community, which usually contains the largest number of its neighbors.

Table 1
The detail of benchmark functions.

Name	Formulation	Range	Optimum
f_1: Sphere	$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
f_2: Schwefel	$f_2(\mathbf{x}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^D$	0
f_3: High conditioned Elliptic	$f_3(\mathbf{x}) = \sum_{i=1}^{D}\left(10^6\right)^{(i-1)/(D-1)} z_i^2$	$[-100, 100]^D$	0
f_4: Schwefel with noise	$f_4(\mathbf{x}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2 \cdot (1 + 0.4\,\lvert N(0, 1)\rvert)$	$[-100, 100]^D$	0
f_5: Noisy Quartic	$f_5(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$	$[-1.28, 1.28]^D$	0
f_6: Rosenbrock	$f_6(\mathbf{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$	$[-100, 100]^D$	0
f_7: Griewank	$f_7(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[0, 600]^D$	0
f_8: Ackley	$f_8(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$	0
f_9: Rastrigin	$f_9(\mathbf{x}) = \sum_{i=1}^{D}\left(x_i^2 - 10cos(2\pi x_i) + 10\right)$	$[-5.12, 5.12]^D$	0
f_{10}: Rotated Rastrigin	$f_{10}(\mathbf{x}) = \sum_{i=1}^{D}\left(z_i^2 - 10cos(2\pi z_i) + 10\right)$, $Z = X \cdot M, \quad cond(M) = 2$	$[-5.12, 5.12]^D$	0
f_{11}: Noncontinuous Rastrigin	$f_{11}(\mathbf{x}) = \sum_{i=1}^{D}\left(z_i^2 - 10cos(2\pi z_i) + 10\right)$, $z_i = \begin{cases} x_i & if\ \lvert x_i < 0.5\rvert \\ round(2x_i/2) & else \end{cases}$	$[-5.12, 5.12]^D$	0
f_{12}: Weierstrass	$f_{12}(\mathbf{x}) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k(x_i + 0.5))]\right) - D\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k \cdot 0.5)]$, $a = 0.5, b = 3, k_{max} = 20$	$[-0.5, 0.5]^D$	0
f_{13}: Generalized penalized function 1	$f_{13}(\mathbf{x}) = \frac{\pi}{D}\left\{10sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2 \cdot \left[1 + 10sin^2(\pi y_{i+1}) + (y_D - 1)^2\right]\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$	$[-50, 50]^D$	0
f_{14}: Generalized penalized function 2	$f_{14}(\mathbf{x}) = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2 \cdot \left[1 + sin^2(3\pi x_{i+1}) + (x_D - 1)\left[1 + sin^2(2\pi x_D)\right]\right]\right\}$ $+\sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$	0
f_{15}: Expanded Griewank + Rosenbrock	$f_{15}(\mathbf{x}) = f_7(f_6(x_1, x_2)) + f_7(f_6(x_2, x_3)) + \ldots + f_7(f_6(x_D, x_1))$	$[-3, 1]^D$	0
f_{16}: Expanded Scaffer	$f_{16}(\mathbf{x}) = f(x_1, x_2) + f(x_3, x_4) + \ldots + f(x_D, x_1)$, $f(x_i, x_j) = 0.5 + \frac{sin\sqrt[2]{x_i^2 + x_j^2} - 0.5}{1 + 0.001\left(x_i^2 + x_j^2\right)^2}$	$[-100, 100]^D$	0
f_{17}: Composition Function 1 (CF1)[a]	$C = 2000, [\sigma_1, \sigma_2, \ldots \sigma_{10}] = [1, 1, \ldots, 1], [\lambda_1, \lambda_2, \ldots \lambda_{10}] = [5/100, 5/100, \ldots, 5/100]$	$[-5, 5]^D$	0
f_{18}: Hybrid Composition Function[b]	$C = 2000, [\sigma_1, \sigma_2, \ldots \sigma_{10}] = [1, 1, \ldots, 1]$, $[\lambda_1, \lambda_2, \ldots \lambda_{10}] = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$	$[-5, 5]^D$	0

[a] The Composition function (CF1) is composed by using ten sphere functions.

[b] The Hybrid Composition function is composed by using ten different benchmark functions, i.e., two Rastrigin??s functions, two Weierstrass functions, two Griewank??s functions, two Ackley??s functions, and two Sphere functions.

Table 2
Convergence performance comparison of DE variants.

Test Functions	rand/2/dir	rand/1/bin	rand-to-best/2/bin	best/1/bin	DE_version1	DE_version2
f_1: Sphere	3.6994e−14 (1.367e−13)	6.560e−37 (2.116e−37)	4.543e−52 (1.237e−51)	2.154e−105 (1.080e−103)	6.654e−299 (1.186e−296)	**0.000e+00 (0.000e+00)**
f_2: Schwefel	6.759e−01 (1.97e−01)	2.069e−04 (1.08e−04)	1.027e−04 (3.26e−04)	6.225e+00 (4.59e−01)	6.570e−52 (5.161e−52)	**0.000e+00 (0.000e+00)**
f_3: High conditioned Elliptic	3.257e−25 (2.116e−22)	5.463e−58 (3.220e−57)	1.667e−88 (2.100e−86)	8.573e−102 (3.357e−101)	1.680e−302 (4.074e−302)	**0.000e+00 (0.000e+00)**
f_4: Schwefel with noise	4.442e−00 (3.61e−00)	9.817e−06 (6.11e−05)	9.886e−07 (2.29e−06)	4.733e+02 (3.34e+01)	2.705e−51 (2.640e−51)	**0.000e+00) (0.000e+00)**
f_5: Noisy Quartic	2.839e−02 (3.57e−02)	8.128e−03 (3.29e−03)	1.012e−03 (1.29e−03)	3.937e−03 (1.43e−02)	2.4890e−05 (1.475e−05)	**5.123e−07 (1.532e−07)**
f_6: Rosenbrock	5.750e−01 (9.040e−00)	4.746e−01 (3.570e−00)	5.198e−01 (3.120e−00)	**1.140e−15 (3.767e−12)**	2.579e+01 (1.724e+00)	2.579e+01 (1.724e+00)
f_7: Griewank	8.460e−08 (2.237e−07)	7.622e−32 (2.267e−31)	8.860e−40 (1.106e−40)	9.802e−38 (1.960e−36)	**0.000e+00 (0.000e+00)**	**0.000e+00 (0.000e+00)**
f_8: Ackley	**−1.829e−06 (3.66e−06)**	**−1.829e−06 (2.20e−06)**	**−1.829e−06 (2.19e−06)**	**−1.829e−06 (1.44e−06)**	**−1.829e−06 (2.071e−06)**	**−1.829e−06 (1.260e−06)**
f_9: Rastrigin	7.219e+01 (8.84e−00)	2.759e+01 (6.14e−00)	9.714e+01 (3.83e−00)	3.487e+01 (5.77e−00)	5.6838e−05 (5.77e−00)	**0.000e+00 (0.000e+00)**
f_{10}: Rotated Rastrigin	4.895e+02 (3.66e−00)	4.680e+01 (2.20e−00)	5.700e+01 (4.17e−00)	3.471e+01 (2.16e−00)	1.980e+01 (1.087e−01)	**8.4649e−05 (2.036e−04)**
f_{11}: Noncontinuous Rastrigin	9.835e+01 (2.47e−01)	6.775e+02 (3.11e−01)	1.285e+02 (9.36e−01)	2.417e+02 (4.23e−01)	1.105e+00 (1.227e+00)	**0.000e+00 (0.000e+00)**
f_{12}: Weierstrass	3.763e+01 (2.33e+00)	7.788e+00 (2.10e+00)	2.767e+00 (3.81e+00)	3.671e+00 (3.66e+00)	**0.0000e+00 (0.000e+00)**	**0.0000e+00 (0.000e+00)**
f_{13}: Generalized penalized function 1	**1.206e−24 (2.30e−24)**	**1.5705e−32 (1.09e−32)**	**1.5705e−32 (3.11e−32)**	**1.5705e−32 (4.80e−32)**	1.280e−02 (3.216e−02)	1.2800e−02 (2.006e−02)
f_{14}: Generalized penalized function 2	−1.1504+00 (1.28e+00)	−1.1504+00 (1.12e+00)	−1.1504+00 (2.62e+00)	−1.1504+00 (3.20e+00)	3.430e−02 (2.020e−02)	**3.171e−02 (1.324e−02)**
f_{15}: Expanded Griewank +Rosenbrock	8.770e+01 (3.96e−01)	2.738e+01 (2.05e−01)	3.336e+01 (7.20e−01)	1.320e+01 (2.94e−01)	9.442e+00 (1.392e−01)	**6.7857e+00 (1.724e+00)**
f_{16}: Expanded Scaffer	7.776e+01 (5.22e−01)	4.180e+01 (5.51e−01)	2.560e+01 (4.76e−01)	2.790e+01 (3.27e−01)	8.453e−01 (3.002e−01)	**3.4748e+00 (3.2260e+00)**
f_{17}: Composition Function 1	6.483e−27 (2.28e−27)	6.813e−66 (3.10e−66)	5.6679e−93 (2.68e−93)	1.6266e−106 (2.35e−106)	**0.0000e+00 (0.000e+00)**	**0.0000e+00 (0.000e+00)**
f_{18}: Hybrid Function Composition	9.806e−06 (6.70e−06)	5.250e−06 (2.22e−06)	4.617e−06 (3.82e−06)	6.680e−06 (4.51e−06)	9.980e−07 (3.002e−07)	**3.6630e−07 (2.852e−07)**

In the global search process of available DE-based modularity optimization algorithms, the above-mentioned community modification operation is usually conducted several times on each individual in every generation. For each individual in the population, a certain number of nodes will be selected randomly, and then each node will be checked whether its community membership need to be modified. The community modification operation can efficiently enhance the quality of the individuals and speed up convergence of the whole population. However, the DE-based algorithms still suffer from the problem of premature convergence. The main reason is the inappropriate use of topology information during the modification, which destroys the search space for the global optimal partition.

Let us take the real-world Dolphin network for example. In a certain individual obtained from the population during the evolution, which represents a feasible partition of communities, the 21th node has 9 neighbor nodes belonging to 4 different communities. The largest community contains nodes {17, 39, 45, 51}, and the smallest community contains only one node {19}. Besides, there are two same second-largest communities, containing nodes {29, 48} and {9, 37} respectively. According to the existing community modification operation, if the community of the 21th node needs to be modified, it can only be put into its largest neighbor community {17, 39, 45, 51} as illustrated in Fig. 2 . However, our experiment results prove that, the maximum value of modularity can only be obtained when the 21th node is placed into the second-largest community {29, 48}, which means that if the node belongs to the largest community then the algorithm gets stuck in local optimum.

Based on the above phenomenon, we know that excessive restriction of the community modification is implemented on a node, and thus the search space for the global optimal partition is artificially reduced. In order to avoid this kind of inappropriate use of topology information, a new community modification strategy is proposed in CDEMO. If the criteria of community modification is satisfied, a certain node in an individual will be put into all of its neighbor communities with different probabilities which are proportional to the scale of the neighbor communities. This new strategy can also reduce

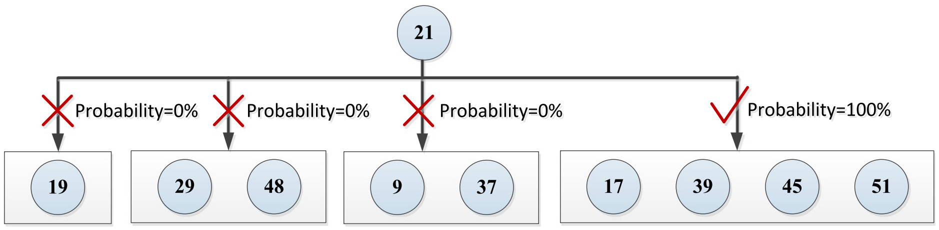

Fig. 2. Community modification of the 21th node in Dolphin.

Table 3
Features of benchmark real-world networks.

Networks	Number of nodes	Number of edges	Number of communities
Karate	34	78	[2,10]
Dolphins	62	159	[2,10]
Polbooks	105	441	[2,10]
Football	115	613	[5,15]

the search space as efficient as the existing one. Besides, it can relax the restriction during the community modification and thus provide adequate search space for the global optimal partition. The new strategy can make a better use of the known topology information of networks and promote the convergence of CDEMO.

3.2. DE-based modularity optimization algorithm

According to the improvement measures mentioned above, the flowchart of the new DE-based modularity optimization algorithm CDEMO could be constructed as shown in Fig. 3. CDEMO takes the modularity as its objective function and selects the improved DE as the optimization strategy. In CDEMO, population individuals are constructed by using the community identifier-based representation, where all the communities and nodes belonging to each community can be identified straightforwardly from each individual. Therefore, the number of communities can be automatically acquired and no decoding process or any prior information is required. As the population converges to the global optimum, the number of communities will also been optimized accordingly. At each generation, there are 5 steps to be conducted for every target individual $\mathbf{x}_{i,t}$ in the population *pop*. Specially, in the crossover operation on each individual $\mathbf{x}_{i,t}$, if conditions are satisfied, both of the *j*th target component in the mutant vector $\mathbf{v}_{i,t}$ and all of the components belonging to the same community will be saved into the trial vector $\mathbf{u}_{i,t}$. Otherwise, components in $\mathbf{u}_{i,t}$ will be set as the same as that in $\mathbf{x}_{i,t}$. Under the community identifier-based representation, such kind of crossover helps to maintain community structure information and avoid generating illegal partition. After evolutionary operations of mutation and crossover, the neighborhood-based community modification will be conducted on each individual to promote convergence.

The detailed steps of CDEMO are shown in Algorithm 1. The main complexity of CDEMO relies on its cycling process. Suppose that the network has n nodes and the average degree of each node is d. The cycling process of CDEMO mainly contains 4 steps and 3 operations, including the mutation, crossover and community modification operations. Since the computation of modularity can be accomplished in $O(n^2)$ times, the time complexity of Steps 3.1–3.4 is $O(n+n^2)$, $O(nd^2)$, $O(n+n^2)$, $O(nd^2)$ in the worst case, respectively. Therefore, the total time complexity of CDEMO is $O(t_{max} * NP * max(O(n+n^2), O(nd^2), O(n+n^2), O(nd^2)))$, which can be simplified as $O(t_{max} * NP * n^2)$. The time complexity of CDEMO is equal to its most competitive contrast algorithm IDDE.

3.3. Performance validation of CDEMO

In this section, experiments are conducted to prove the efficiency of the new community modification operation, and to verify whether the convergence improvement of DE benefits its application in the modularity optimization. To accomplish the experiments, 6 kinds of DE-based Modularity Optimization algorithms, called DEMO1-6, are constructed. These algorithms adopt different DE variants (with different trial vector generation strategies) as the optimization strategy to optimize the modularity. In DEMO1-4, different DE variants *DE/rand/2/dir*, *DE/rand/1/bin*, *DE/current-to-best/2/bin* and *DE/best/1/bin* are utilized respectively. DEMO5 adopts a widely used random mutation strategy, which means that communities of nodes are adjusted in a completely random manner. DEMO6 takes our improved *DE_version2* as its optimization strategy. Based on DEMO6, CDEMO is constructed by incorporating the novel community modification operation proposed in Section 3.1.

All the algorithms are tested on 4 real-world social networks, as illustrated in Table 3, including the Zachary's karate club network [28], the Dolphin network [29], the American politics books network [4] and the American college football network [4]. All the networks are well-known benchmark examples for the community detection algorithms and have been

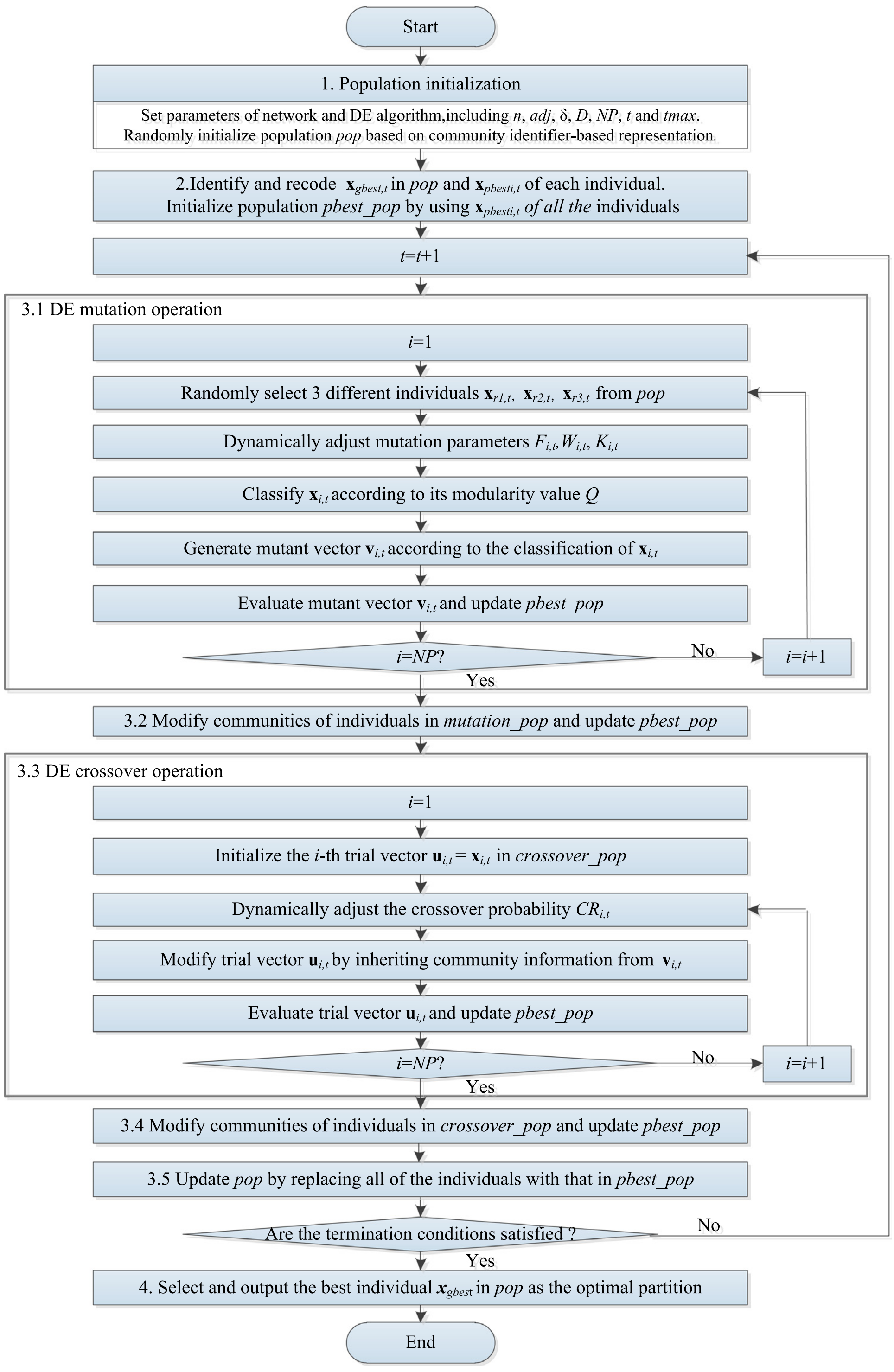
Start
1. Population initialization
Set parameters of network and DE algorithm,including n, adj, δ, D, NP, t and tmax.
Randomly initialize population pop based on community identifier-based representation.
2.Identify and recode $\mathbf{x}_{gbest,t}$ in pop and $\mathbf{x}_{pbesti,t}$ of each individual.
Initialize population pbest_pop by using $\mathbf{x}_{pbesti,t}$ of all the individuals
t=t+1
3.1 DE mutation operation
i=1
Randomly select 3 different individuals $\mathbf{x}_{r1,t}$, $\mathbf{x}_{r2,t}$, $\mathbf{x}_{r3,t}$ from pop
Dynamically adjust mutation parameters $F_{i,t}$, $W_{i,t}$, $K_{i,t}$
Classify $\mathbf{x}_{i,t}$ according to its modularity value Q
Generate mutant vector $\mathbf{v}_{i,t}$ according to the classification of $\mathbf{x}_{i,t}$
Evaluate mutant vector $\mathbf{v}_{i,t}$ and update pbest_pop
i=NP?
No
i=i+1
Yes
3.2 Modify communities of individuals in mutation_pop and update pbest_pop
3.3 DE crossover operation
i=1
Initialize the i-th trial vector $\mathbf{u}_{i,t} = \mathbf{x}_{i,t}$ in crossover_pop
Dynamically adjust the crossover probability $CR_{i,t}$
Modify trial vector $\mathbf{u}_{i,t}$ by inheriting community information from $\mathbf{v}_{i,t}$
Evaluate trial vector $\mathbf{u}_{i,t}$ and update pbest_pop
i=NP?
No
i=i+1
Yes
3.4 Modify communities of individuals in crossover_pop and update pbest_pop
3.5 Update pop by replacing all of the individuals with that in pbest_pop
Are the termination conditions satisfied ?
No
Yes
4. Select and output the best individual x_{gbest} in pop as the optimal partition
End

Fig. 3. The flowchart of the DE-based modularity optimization algorithm CDEMO.

well studied in literatures. Experiment results are presented in Table 4, including the mean and standard deviation of the modularity Q calculated by each algorithm in 30 independent runs.

From Table 4 we can see clearly that, different DE variants perform dissimilarly in the modularity optimization problem due to their different convergence abilities. Compared with DEMO3 and DEMO4, DEMO1-2 and DEMO5 obtain better Q_{avg} and Q_{std} because of their stronger exploration ability brought by the random components in the mutation strategies. In addition,

Algorithm 1 CDEMO algorithm

```
1: Population Initialization;
   1.1 Set network parameters, including the number of nodes $n$, adjacent matrix $adj$, the threshold value ?? of community modification. Set parameters of DE algorithm, including individual dimension $D$, population size $NP$, number of generation $t$, and the maximum number of generations $t_{max}$;
   1.2 Randomly initialize the population $pop$ based on the community identifier-based representation;
2: Identify and Recode the Best Individuals;
   2.1 Identify and record the best individual $\mathbf{x}_{gbest,t}$ in $pop$ at the $t$-th generation;
   2.2 Identify and record the historical optimal solution $\mathbf{x}_{pbesti,t}$ of each individual $\mathbf{x}_{i,t}$ at the $t$-th generation. Construct the initial population $pbest\_pop$ by using the $\mathbf{x}_{pbesti,t}$ of all the individuals;
3: while $t < t_{max}$ do;
   $t = t + 1$;
   3.1 Construct $mutation\_pop$ by the classification-based self-adaptive DE mutation;
   for $i = 1$ to $NP$ do;
   a) Randomly select 3 different individuals $\mathbf{x}_{r1,t}$, $\mathbf{x}_{r2,t}$, $\mathbf{x}_{r3,t}$ from $pop$;
   b) Dynamically adjust mutation parameters $F_{i,t}$, $W_{i,t}$, $K_{i,t}$ (See Section 2.2 for details);
   c) Classify $\mathbf{x}_{i,t}$ according to its fitness value $Q$ (See Section 2.1 for details);
   d) Generate the mutant vector $\mathbf{v}_{i,t}$ according to the classification-based DE mutation strategy (See Section 2.1 for details);
   e) Evaluate the modularity value of the $\mathbf{v}_{i,t}$, compare it with the $\mathbf{x}_{i,t}$, and save the better one into the $pbest\_pop$.
   end for;
   3.2 Community modification based on the neighborhood information (See Section 3.1 for details);
   Randomly select several nodes from each individual $\mathbf{v}_{i,t}$ in the $mutation\_pop$, perform the neighborhood-based community modification, and update the $pbest\_pop$;
   3.3 Construct $crossover\_pop$ based on $mutation\_pop$ and $pop$;
   for $i = 1$ to $NP$ do;
   a) Initialize the $i$-th trial vector $\mathbf{u}_{i,t} = \mathbf{x}_{i,t}$ in $crossover\_pop$;
   b) Dynamically adjust the crossover parameter $CR_{i,t}$ (See Section 2.2 for details);
   c) Modify the trial vector $\mathbf{u}_{i,t}$ by inheriting community information from the mutant vector $\mathbf{v}_{i,t}$;
   d) Evaluate the modularity value of the $\mathbf{u}_{i,t}$, compare it with the $i$-th individual in $pbest\_pop$ and save the better one into the $pbest\_pop$.
   end for;
   3.4 Community modification based on the neighborhood information (See Section 3.1 for details);
   Randomly select several nodes from each individual $\mathbf{u}_{i,t}$ in the $crossover\_pop$, perform the neighborhood-based community modification, and update the $pbest\_pop$;
   3.5 Update $pop$ by replacing all of the individuals with that in the $pbest\_pop$;
   end while;
4: Terminate the algorithm if the stop criteria are satisfied, and output the $\mathbf{x}_{gbest,t}$ in $pop$ as the final optimal partition of community. Otherwise go to Step 3.
```

Table 4
Performance comparison of DE variants on modularity optimization.

Networks	DEMO1	DEMO2	DEMO3	DEMO4	DEMO5	DEMO6	CDEMO
Karate	0.4176 (2.00e−03)	0.4173 (2.31e−03)	0.4150 (3.40e−03)	0.4172 (1.92e−03)	0.4172 (5.22e−03)	0.4197 (3.10e−04)	**0.4198 (0.00e−00)**
Dolphin	0.5189 (3.70e−03)	0.5166 (5.40e−03)	0.4873 (3.80e−03)	0.5136 (9.00e−03)	0.5266 (1.74e−04)	0.5277 (1.42e−04)	**0.5283 (3.11e−04)**
PolBooks	0.5262 (2.25e−04)	0.5264 (2.14e−04)	0.5236 (1.70e−03)	0.5262 (1.00e−04)	0.5266 (1.03e−04)	0.5268 (1.45e−04)	**0.5270 (4.03e−04)**
Football	**0.6046 (0.00e−00)**	**0.6046 (0.00e−00)**	0.6043 (6.11e−04)	0.6041 (7.04e−04)	0.6045 (8.55e−05)	**0.6046 (0.00e−00)**	**0.6046 (0.00e−00)**

DEMO6 performs better than DEMO1-5 in terms of the accuracy and stability of the optimal partitions, which proves that the convergence performance improvement of DE is beneficial for its application in the modularity optimization. Compared with DEMO6, CDEMO obtains even better Q_{avg} and Q_{std} on Karate, Dolphin and PolBooks networks. The accuracy of the detected communities can be further improved, and the reason is that the novel community modification strategy can keep sufficient search space for the global optimal partition and prevent DE from trapping in the local optimum.

Based on the test results we can draw a conclusion that, the convergence performance improvement from both of the global convergence ability of DE and the utilization of topological information is indeed effective in improving the quality of the optimal community partitions in the modularity optimization problem.

4. Experimental performance of community detection

4.1. Experimental settings

In this section, the performance of CDEMO is evaluated on a class of widely used artificial computer-generated networks and real-world social networks. The CDEMO algorithm is implemented in MATLAB 7.0 and all the experiments are performed on Windows 7 with a Pentium Dual-Core 2.5 GHz processor and 2.0GB RAM. Parameters in CDEMO are set as follows: the population size NP is 100, the maximum number of generations t_{max} is 200, the value ranges of control parameters are set to be $W \in [0.1, 0.9]$, $K \in [0.3, 0.9]$, $F \in [0.3, 0.9]$ and $CR \in [0.1, 0.9]$.

4.2. Performance metrics

(1) Modularity Q: For those real world networks, since they do not have known community structures, the modularity Q defined as Eq. (9),

$$Q = \frac{1}{2M} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2M} \right) \delta(i, j), \tag{9}$$

is used as the performance metric to measure the significant level of the community structures detected in networks [2]. In Eq. (9), M is the total number of edges in the network; $A = (a_{ij})_{n*n}$ is the adjacency matrix; k_i and k_j are the degrees of nodes i and j; $\delta(i, j)$ indicates the community relationship between nodes i and j, if they belong to the same community the value is 1, else the value is 0. When the value of Q is larger than 0.3, then the community structure in the network is relatively obvious. Through modularity suffers from the so-called resolution limit problem, it still is the most widely used performance metric in existing studies.

(2) Normalized Mutual Information (NMI): NMI is an information-theoretic measure of the agreement between two partitions [30]. For those artificial computer-generated networks, since the real community structures are known, the NMI defined as Eq. (10) is used as the performance metric to measure the similarity between the detected community structures and the real ones. Suppose that A is the real partition of the network, and B is a predicted partition. We can define a confusion matrix C, where the rows correspond to the real communities defined in A, and the columns correspond to the predicted communities found in B. The element C_{ij} in C is the number of nodes appearing in the ith real community and the jth predicted community simultaneously. Based on the definition of C, metric NMI is defined as follows:

$$NMI(A, B) = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_B} C_{ij} log \left(\frac{C_{ij} N}{C_i C_j} \right)}{\sum_{i=1}^{C_A} C_i log \left(\frac{C_i}{N} \right) + \sum_{j=1}^{C_B} C_j log \left(\frac{C_j}{N} \right)}, \tag{10}$$

where N is the total number of nodes in the network; C_A and C_B are the numbers of communities in partitions A and B; C_i is the sum over row i, representing the number of nodes in the ith community of partition A; C_j is the sum over column j, representing the number of nodes in the jth community of partition B. From the Eq. (10), we can see that if A is equal to B, NMI takes its maximum value of 1. On the contrary, if A is completely different from B, NMI is equal to 0 [2].

4.3. Algorithms for performance comparison

In recent years, several kinds of efficient evolutionary algorithms, such as GA, PSO and DE, have been successfully incorporated in the single-objective or multi-objective optimization algorithms for community detection. According to the embedded optimization strategies, these algorithms can be roughly divided into five categories:

(1) Traditional deterministic optimization algorithms: In early studies, traditional deterministic optimization algorithms were usually adopted to optimize modularity for getting meaningful community structures. One of the most popular algorithms is the well-known GN algorithm [31] which is a divisive method that iteratively removes the edges with the greatness edge betweenness value based on edge betweenness centrality. Later, Newman [6] presented an agglomerative hierarchical clustering method based on the greedy optimization of the network modularity. This method iteratively joins communities of nodes in pairs and chooses the join with the greatest increase in the modularity at each step. Moreover, based on the original strategies, its faster version CNM [32] was proposed by using some shortcuts and some sophisticated data structures. BGLL [33] is also a fast multistep greedy technique that optimizes modularity for community detection. This method aggregates nodes belonging to the same community to build a new network when the modularity gain is the largest. Two kinds of Fuzzy Modularity Maximization (FMM) algorithms, MSFCM [34] and FMM/H1[35] have also been proposed recently, which find fuzzy communities by maximizing a generalized form of Newman?s modularity.

(2) Modularity optimization algorithms based on GA: GA is the most widely used evolutionary algorithm in modularity optimization so far. Researchers have proposed GA-Net [7], GATHB [8], LGA [9], ECGA [10] and MA [16] respectively, all of which used GA as the global optimization strategy for modularity optimization. Except for these single-objective optimization algorithms, GA-based multi-objective optimization algorithms have also been constructed. They usually optimize two or three conflicting objectives simultaneously, such as community score and community fitness, to discover

multiple potential community structures at one time. Pizzuti proposed a multi-objective genetic algorithm named MOGA-Net [12] to uncover community structure by optimizing two objective functions, one is to maximize the number of connections inside each community, and another is to minimize the number of interconnections between communities. Gong et al. presented a multi-objective evolutionary algorithms based on the decomposition framework, named MOEA/D-Net [36].

(3) Modularity optimization algorithms based on PSO: PSO is another commonly used evolutionary algorithms. Cai et al. designed GDPSO [13] algorithm utilizing the greedy discrete PSO algorithm to handle large-scale social network. To avoid being trapped into local optima, a greedy strategy specially designed for the particles to adjust their positions. Later, Zhou et al. designed a neighborhood-impact based community detection algorithm via discrete PSO, utilizing the neighborhood information of nodes to promote modularity optimization [14]. Gong et al. proposed a multi-objective PSO algorithm for complex network clustering based on the multi-objective decomposition scheme (MODPSO) [15]. In their algorithm, two evaluation objectives termed as kernel k-means and ratio cut are to be minimized. A problem-specific population initialization method based on label propagation and a turbulence operator are also introduced. Additionally, Zhang et al. constructed a memetic particle swarm optimization algorithm (MPSOA) [17], where global search operator PSO is incorporated with tabu local search operator to keep a balance between diversity and convergence.

(4) Modularity optimization algorithms based on DE: DE is still one of the most efficient evolutionary optimization algorithm up to now, and it was first utilized to optimize modularity by Jia et al. [23]. In their Differential Evolution based Community Detection algorithm (DECD), a modified binomial crossover operation was designed to transmit topological information about community structures. Moreover, a biased initialization process and a clean-up operation were also employed to improve the quality of population individuals. In the same year, He et al. presented a CCDECD algorithm [24]. In this method, a new mutation operator was designed to make use of connectivity information of networks to improve the search ability. Later, they proposed an improved version named as CoCoMi [2], which employed an improved cooperative co-evolutionary framework to handle large scale networks. Zhang et al. presented a novel immune discrete differential evolution (IDDE) [25], where the initial population is generated through label propagation, and the discrete DE strategy is utilized to ensure the global searching ability.

(5) Modularity optimization algorithms based on other EAs: Except for GA, PSO and DE, ant colony optimization [18–20], clone selection algorithm [21,22] and estimation of distribution algorithms [37] have also been utilized to solve the complex optimization problems in community detection.

In the above-mentioned EA-based modularity optimization algorithms, in order to obtain more accurate community partitions, both of the objective functions and the evolutionary operators have often been redesigned by incorporating topology information of networks, such as the neighborhood or connectivity information of nodes. Such kind of prior information can substantially reduce the huge search space of EAs during their optimization and thus speed up convergence. A set of representative algorithms are selected to make comparison with CDEMO in terms of the accuracy of the optimal partitions.

4.4. *Results of artificial computer-generated networks*

The performance of CDEMO in community detection is validated on artificial computer-generated networks. An extension of the GN benchmark network is used, which is a set of synthetic networks with known community structures proposed by Lancichinetti [38]. These networks have been widely used to benchmark community detection algorithms. Each network has 128 nodes which are divided into 4 communities each with 32 nodes. Each node has an average z_{in} edges, connecting it to members of the same community and z_{out} edges to members of other communities. Moreover, z_{in} and z_{out} are chosen to satisfy the total expected degree of a node $z_{in} + z_{out} = 16$. Community structures in the networks become vaguer with the increase of the z_{out}. An algorithm with good performance should discover all the communities in the network when $z_{out} > z_{in}$.

We test the performance of CDEMO on 9 different GN benchmark networks with gradually increased value of z_{out}. The accuracy and stability of CDEMO is measured according to the average value of *NMI* on 30 independent runs in each case and compared with 10 typical modularity optimization algorithms (i.e., CNM [32], GN [31], GATHB [8], ECGA [10], LGA [9], MA [16], UMDA [37], MOEA/D-Net [36], DECD [23], and IDDE [25]). Experimental results are presented in Fig. 4.

From Fig. 4 it can be seen that, all the algorithms can achieve the best value of *NMI* when $z_{out} \leq 3$, which means that the real community partitions of GN networks could be efficiently detected. However, as the value of z_{out} increases, the community structures in the networks become much vaguer and more difficult to be identified. As a result, all of the algorithms deteriorate with the value of *NMI* decreases. It is worth noting that the value of *NMI* obtained by CDEMO is always better than the other 10 competitors especially when $z_{out} \geq 4$. It means that CDEMO is much more accurate and stable in detecting communities of artificial networks, even those with very vague community structures.

To further investigate the scalability of CDEMO, 8 LFR benchmark networks with increasing value of mixing parameter μ are generated and used [38]. In LFR benchmark network, distribution of node degrees and size of communities are both power laws with tunable exponents, which makes it much closer to real-world networks. The mixing parameter μ determines the fraction of edges between each node and its neighbors in other communities, a big value of μ corresponds to a much vaguer network structure. In our experiments, 8 different LFR networks are generated, and the value of μ is increased from 0 to 0.7 with an interval of 0.1. Each network contains 1000 nodes and the size of communities ranges from 10 to 50. The average

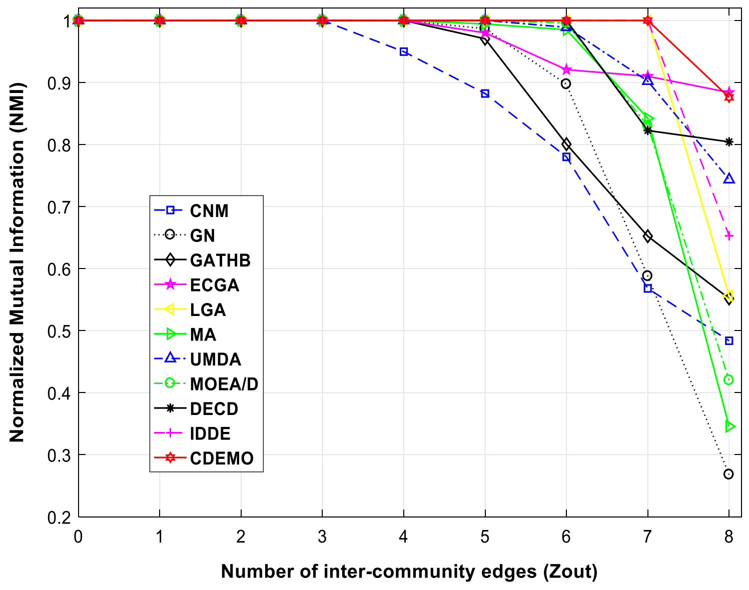

Fig. 4. Average *NMI* of CDEMO and other algorithms as z_{out} is varied for GN networks. The accuracy and stability of CDEMO is measured according to the average value of *NMI* on 30 independent runs in each case and compared with 10 typical modularity optimization algorithms (i.e., CNM [32], GN [31], GATHB [8], ECGA [10], LGA [9], MA [16], UMDA [37], MOEA/D-Net [36], DECD [23], and IDDE [25]).

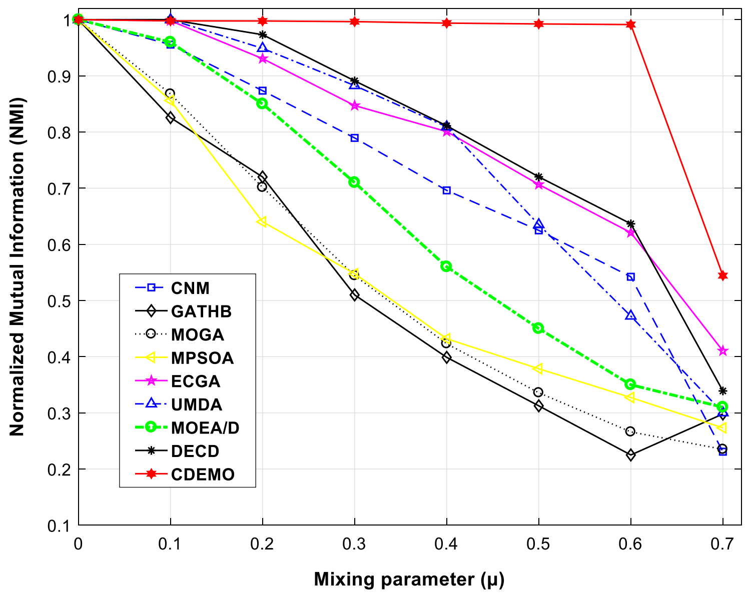

Fig. 5. Average *NMI* of CDEMO and other algorithms as μ is varied for LFR networks. We executed CDEMO 30 runs on each LFR network, which is the same as that of the other 8 comparative algorithms, including CNM [32], GATHB [8], MOGA-Net [12], MPSOA [17], ECGA [10], UMDA [37], MOEA/D-Net [36], and DECD [23].

degree of each node is 20 and the max node degree is set as 50. Communities detected by CDEMO are compared with the corresponding ground truth and the metric NMI is used to evaluate their precision. We executed CDEMO 30 runs on each LFR network, which is the same as that of the other 8 comparative algorithms, including CNM [32], GATHB [8], MOGA-Net [12], MPSOA [17], ECGA [10], UMDA [37], MOEA/D-Net [36], and DECD [23]. Experimental results are presented in Fig. 5.

From Fig. 5 it can be observed that, compared with other modularity optimization algorithms, CDEMO can always obtain the best value of *NMI* on all of the 8 different LFR networks. We can also find that, for the networks with small values of μ ($\mu < 0.2$), the performance of CDECD is just the same or slightly better than the competitors. However, with the growth of μ, the superiority of CDEMO over other algorithms becomes much more significant as illustrated by the accuracy and stability of the value of *NMI*. Such result indicates that CDEMO has comparatively better scalability and accuracy in the module identification problem of artificial computer-generated networks.

Table 5
Best values of Q obtained by Fast_Nm, CNM, GN, BGLL, MSFCM and FMM/H1 on real-world networks.

Networks	Fast_Nm	CNM	GN	BGLL	MSFCM	FMM/H1
Karate	0.3810	0.3807	0.4013	0.4150	0.4231	0.3941
Dolphins	0.4960	0.4950	0.5194	0.4950	0.3991	0.4882
Polbooks	0.5020	0.5019	0.5099	0.5150	0.4601	0.5175
Football	0.5490	0.5770	0.5994	0.6010	0.5268	0.5960

Table 6
Best values of Q obtained by GATHB, MOGA, ECGA and MOEA/D on real-world networks.

Networks	GATHB	MOGA	ECGA	MOEA/D
Karate	0.4024	0.4160	**0.4198**	0.3715
Dolphins	0.5219	0.5050	0.5242	0.3735
Polbooks	0.5176	0.5180	0.5269	0.5180
Football	0.5508	0.5220	0.6010	0.6005

Table 7
Best values of Q obtained by MPSOA , MODPSO, DECD, CCDECD, IDDE and CDEMO on real-world networks.

Networks	MPSOA	MODPSO	DECD	CCDECD	IDDE	CDEMO
Karate	**0.4198**	**0.4198**	**0.4198**	**0.4198**	**0.4198**	**0.4198**
Dolphins	0.5191	0.5268	0.5249	0.5216	0.5282	**0.5285**
Polbooks	0.5255	0.5260	0.5262	0.5268	**0.5271**	**0.5271**
Football	0.6030	0.6032	**0.6046**	**0.6046**	**0.6046**	**0.6046**

4.5. Results of real-world social networks

In this section, the performance of CDEMO is validated on real-world social networks presented in Table 3. Sixteen modularity identification algorithms are utilized to make comparison with CDEMO in terms of the accuracy of the optimal partitions. These algorithms can be classified into three groups. The first group contains 6 kinds of traditional deterministic modularity optimization algorithms, including Fast_Nm [6], CNM [32], GN [31], BGLL [33], MSFCM [34], FMM/H1 [35]. The second group contains 4 kinds of GA-based modularity optimization algorithms, including GATHB [8], MOGA-Net [12], ECGA [12], and MOEA/D-Net [36]. The last group contains 5 kinds of modularity optimization algorithms based on the PSO and DE, including MPSOA [17], MODPSO [15], DECD [23], CCDECD [24], and IDDE [25]. All of the algorithms are executed 30 runs on each test network, and the modularity Q is utilized as the performance metric to measure the quality of the optimal partitions. Experiment results are presented in Tables 5–7, recoding the best values of Q obtained by CDEMO and the other competitors. For some of the competitors in Tables 5–7, such as the MOGA-Net, MODPSO, MPSOA, Fast_Nm and BGLL, we collected part of the results from literatures (i.e., [12,15,17,6,33]), and run the codes of the other algorithms to obtain the results which could not be found from literatures.

Results in Tables 5–7 demonstrate that, though all of algorithms can identify communities of real-world networks, no significant difference is detected among the first class algorithms. Moreover, the EA-based algorithms significantly outperform the traditional deterministic modularity optimization algorithms. In EA-based algorithms, the best values of Q obtained by DECD, CCDECD, IDDE and CDEMO are much higher than others, which shows the superiority of the DE-based optimization strategy utilized in the algorithm. Though PSO-based modularity optimization algorithms (MPSOA and MODPSO) can detect the best community of Karate network, their performance on the other networks is barely satisfactory. Compared with DECD, CCDECD and IDDE, only CDEMO can always obtain the best values of Q, especially on the Dolphin and Polbooks networks.

Optimal community partitions detected by CDEMO on the 4 real-world social networks are shown in Figs. 6–9. The ground truth of the Karate network has two real communities, and the value of Q is equal to 0.3715. In the optimal partition discovered by CDEMO, there are four clusters and the optimal value of Q is equal to 0.4198, much greater than the real partition. It can be observed that, each of the real community is further divided into two smaller sections. The test result proves that, our algorithm can not only correctly identify the two independent clubs, but also recognize sub-communities with much closer relationship between club members. The above phenomenon also occurred in the dolphin network as shown in Fig. 7, the two real dolphin communities with different genders are divided into four communities. Comparing with the modularity of the real structure ($Q = 0.3722$), CDEMO acquires a much better optimization result ($Q = 0.5285$). Apart from the Karate and Dolphin networks, the clustering results of CDEMO on the Books ($Q = 0.5271$) and Football ($Q = 0.6046$) networks are also given in Figs. 8–9. In the partitions with the highest modularity, our algorithm yields five and ten clusters on the two networks respectively. Therefore, from the perspective of modularity Q, our algorithm is very effective and promising.

Experiment results mentioned above indicate that, apart from the artificial computer-generated networks, CDEMO proposed in this paper can also effectively identify modules of the real-world social networks. Compared with several state-of-the-art modularity optimization algorithms, it is more accurate and stable, which proves the efficiency of the overall convergence performance improvement.

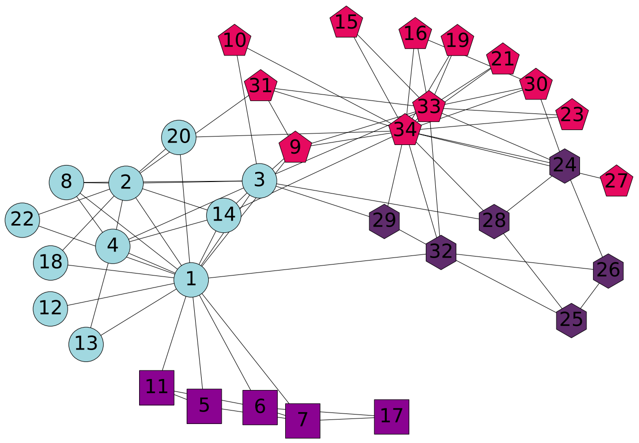

Fig. 6. Communities identified by CDEMO on Karate network.

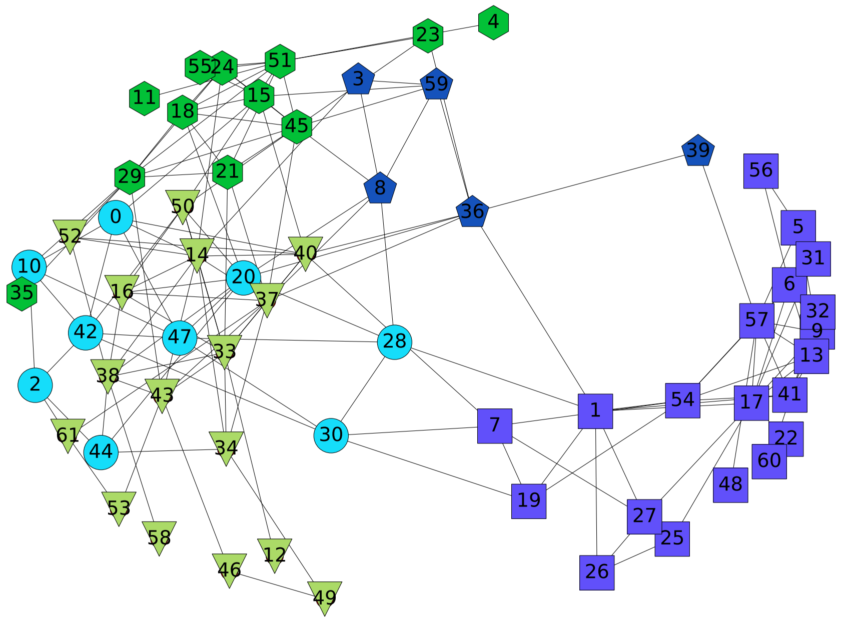

Fig. 7. Communities identified by CDEMO on Dolphins network.

5. Conclusion

To summarize, the quality of the optimal partition obtained by EA-based modularity optimization algorithm mainly depends on two important aspects: one is the convergence performance of EA, another is the complete but small-scale search space of the global optimal partition. In order to improve the quality of the optimal partition, a novel DE-based modularity optimization algorithm, called CDEMO, is proposed in this paper. In CDEMO, improvement measures are designed from both of the two main aspects. On the one hand, a classification-based self-adaptive mutation strategy, a dynamic self-adaptive parameter adjustment method, and a historical information-based selection operation are designed to improve the global convergence ability of the original standard DE algorithm, providing an efficient optimization strategy for the modularity optimization in the community detection of complex networks. On the other hand, an novel community modification strategy is also proposed. The efficiency in the use of the topology information of networks is enhanced by relaxing the restriction during the community modification and providing adequate search space for the global optimal partition.

The efficiency of every improvement measures in CDEMO is proved and the performance of CDEMO is tested on a set of artificial computer-generated networks and real-world social networks, compared with several state-of-the-art algorithms. Experimental results demonstrate that, the overall performance improvement from both of the two key aspects is indeed

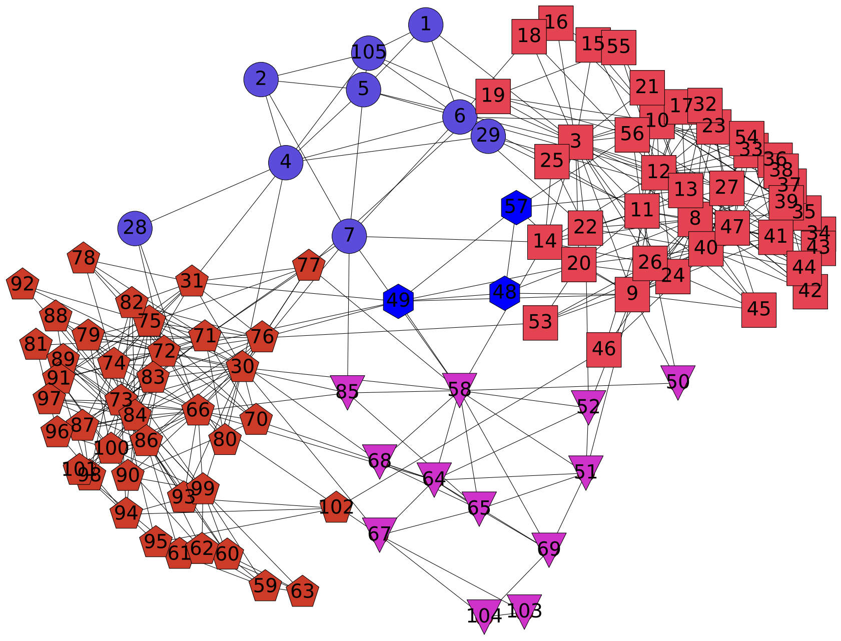

Fig. 8. Communities identified by CDEMO on Polbooks network.

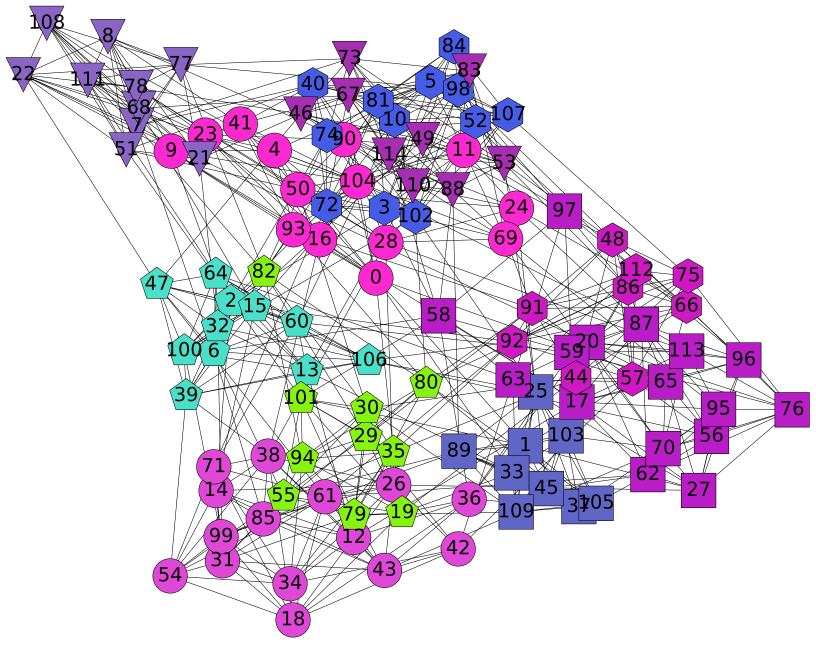

Fig. 9. Communities identified by CDEMO on Football network.

effective in enhancing the convergence ability of DE-based modularity optimization algorithm. CDEMO can effectively identify communities of complex networks and enhance the accuracy and stability of the optimal partitions, including those with very vague community structures. Since CDEMO requires little prior knowledge about the community structure and is not sensitive to the properties of networks, it can be widely used in various kinds of networks. It is worth pointing out that the test networks used to evaluate our CDEMO algorithm is relative small, and its performance in handling more large-scale complex networks remain to be studied. How to enhance the precision and efficiency of DE-based modularity optimization algorithms on large-scale social networks is still an open question to the researchers in the field.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61773091, 61603073, 61501107, 61374170), Liaoning Provincial Natural Science Foundation of China (201602200), Heilongjiang Provincial Postdoctoral Science Foundation (LBH-Z12073), the Dalian Youth Technology Star Project (2015R091), and the Open Project Program of Guizhou Provincial Key Laboratory of Public Big Data (2017BDKFJJ001).

References

[1] Q. Cai, L. Ma, M. Gong, D. Tian, A survey on network community detection based on evolutionary computation, Int. J. Bio-Inspired Comput. 8 (2) (2016) 84–98.
[2] 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. Evol. Comput. 20 (6) (2016) 874–891.
[3] Z.F. Leng, Community detection in complex networks based on greedy optimization, Acta Electron. Sin. 42 (4) (2014) 723–729.
[4] M. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. USA 103 (23) (2006) 8577–8582.
[5] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2) (2005) 027104.
[6] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2003) 066133.
[7] C. Pizzuti, GA-Net: A genetic algorithm for community detection in social networks, in: International Conference on Parallel Problem Solving from Nature: PPSN X, 2008, pp. 1081–1090.
[8] M. Tasgin, A. Herdagdelen, H. Bingol, Community detection in complex networks using genetic algorithms, 2007. ArXiv e-prints, arXiv:0711.0491.
[9] D. Jin, J. Liu, B. Yang, D. He, D. Liu, Genetic algorithm with local search for community detection in large-scale complex networks, Acta Automat. Sinica 37 (7) (2011) 873–882.
[10] J. Li, Y. Song, Community detection in complex networks using extended compact genetic algorithm, Soft Comput. 17 (6) (2013) 925–937.
[11] Z. Li, J. Liu, A multi-agent genetic algorithm for community detection in complex networks, Physica A 449 (2016) 336–347.
[12] C. Pizzuti, A multiobjective genetic algorithm to find communities in complex networks, IEEE Trans. Evol. Comput. 16 (3) (2012) 418–430.
[13] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, L. Jiao, Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. 316 (20) (2015) 503–516.
[14] D. Zhou, X. Wang, A neighborhood-impact based community detection algorithm via discrete PSO, Math. Probl. Eng. 2016 (4) (2016) 1–15.
[15] M. Gong, Q. Cai, X. Chen, L. Ma, Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition, IEEE Trans. Evol. Comput. 18 (1) (2014) 82–97.
[16] M. Gong, B. Fu, L. Jiao, H. Du, Memetic algorithm for community detection in networks, Phys. Rev. E 84 (5) (2011) 056101.
[17] C. Zhang, X. Hei, D. Yang, L. Wang, A memetic particle swarm optimization algorithm for community detection in complex networks, Int. J. Pattern Recognit. Artif. Intell. 30 (02) (2016) 1659003.
[18] Y. Liu, J. Luo, H. Yang, L. Liu, Finding closely communicating community based on ant colony clustering model, in: International Conference on Artificial Intelligence and Computational Intelligence, 2010, pp. 127–131.
[19] S. Sadi, S. Ödücü, A.S. Uyar, An efficient community detection method using parallel clique-finding ants, Evol. Comput. (2010) 1–7.
[20] D. Jin, D. Liu, B. Yang, C. Baquero, D. He, Ant colony optimization with Markov random walk for community detection in graphs, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2011, pp. 123–134.
[21] G. Chen, Y. Wang, Y. Yang, Community detection in complex networks using immune clone selection algorithm, Int. J. Digit. Content Technol. Appl. 5 (6) (2011) 182–189.
[22] 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.
[23] G. Jia, Z. Cai, M. Musolesi, Y. Wang, A.T. Dan, R.J.M. Weber, J.K. Heath, S. He, Community detection in social and biological networks using differential evolution, Lecture Notes in Comput. Sci. (2012) 71–85.
[24] Q. Huang, T. White, G. Jia, M. Musolesi, N. Turan, K. Tang, S. He, J.K. Heath, X. Yao, Community Detection using Cooperative Co-Evolutionary Differential Evolution, Springer Berlin Heidelberg, 2012, pp. 235–244.
[25] Y.J. Zhang, Z.H. Gong, Q.K. Chen, Community detection in complex networks using immune discrete differential evolution algorithm, Acta Automat. Sinica 41 (4) (2015) 749–757.
[26] R. Storn, K. Price, Differential evolution –A simple and efficient heuristic for global optimization over continuous spaces, Global Optim. 11 (1997) 341–359.
[27] X.J. Bi, J. Xiao, Classification-based self-adaptive differential evolution with fast and reliable convergence performance, Soft Comput. 15 (8) (2011) 1581–1599.
[28] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99 (12) (2002) 7821–7826.
[29] D. Lusseau, M.E. Newman, Identifying the role that animals play in their social networks, Proc. R. Soc. B 271 (Suppl. 6) (2004) S477–S481.
[30] L. Danon, A. Dazguilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (2005) 09008.
[31] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[32] A. Clauset, M.E. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2) (2004) 066111.
[33] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) 155–168.
[34] T.C. Havens, J.C. Bezdek, C. Leckie, K. Ramamohanarao, M. Palaniswami, A soft modularity function for detecting fuzzy communities in social networks, IEEE Trans. Fuzzy Syst. 21 (6) (2013) 1170–1175.
[35] J. Su, T.C. Havens, Quadratic program-based modularity maximization for fuzzy community detection in social networks, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1356–1371.
[36] M. Gong, L. Ma, Q. Zhang, L. Jiao, Community detection in networks by using multiobjective evolutionary algorithm with decomposition, Physica A 391 (15) (2012) 4050–4060.
[37] M.G. Parsa, N. Mozayani, A. Esmaeili, An EDA-based community detection in complex networks, in: International Symposium on Telecommunications, 2015, pp. 476–480.
[38] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (2) (2008) 046110.