Community Detection in Social and Biological Networks Using Differential Evolution

Guanbo Jia[1], Zixing Cai[1], Mirco Musolesi[4], Yong Wang[1], Dan A. Tennant[3], Ralf J.M. Weber[2], John K. Heath[2], and Shan He[2,4,⋆]

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China
[2] Center for Systems Biology, School of Biological Sciences
[3] School of Cancer Sciences
[4] School of Computer Science, University of Birmingham, Birmingham, B15 2TT, United Kingdom

Abstract. The community detection in complex networks is an important problem in many scientific fields, from biology to sociology. This paper proposes a new algorithm, Differential Evolution based Community Detection (DECD), which employs a novel optimization algorithm, differential evolution (DE) for detecting communities in complex networks. DE uses network modularity as the fitness function to search for an optimal partition of a network. Based on the standard DE crossover operator, we design a modified binomial crossover to effectively transmit some important information about the community structure in evolution. Moreover, a biased initialization process and a clean-up operation are employed in DECD to improve the quality of individuals in the population. One of the distinct merits of DECD is that, unlike many other community detection algorithms, DECD does not require any prior knowledge about the community structure, which is particularly useful for its application to real-world complex networks where prior knowledge is usually not available. We evaluate DECD on several artificial and real-world social and biological networks. Experimental results show that DECD has very competitive performance compared with other state-of-the-art community detection algorithms.

Keywords: Community structure, graph partitioning, evolutionary computation, Differential Evolution.

1 Introduction

In the fields of science and engineering, there exist various kinds of complex systems which can be represented as complex networks naturally, such as social networks [26] and the Internet [7]. A complex network consists of nodes (or vertices) and edges (or links) which respectively represent the individual members and their relationships in systems [5]. In recent years, the study of complex networks has attracted more and more attention [1,13,16,30].

⋆ Corresponding author.

Complex networks possess many distinctive properties [12], of which community structure [4] is one of the most studied. The community structure is usually considered as the division of networks into subsets of vertices within which intra-connections are dense while between which inter-connections are sparse [4,12]. Identifying the community structure is very helpful to obtain some important information about the relationship and interaction among nodes.

To detect the underlying community structure in complex networks, many successful algorithms have been proposed so far [4,12]. However, the community detection in networks is a nondeterministic polynomial (NP) hard problem. Most of current community detection algorithms based on greedy algorithms perform poorly on large complex networks. Moreover, many algorithms for community detection also require some prior knowledge about the community structure, e.g., the number of the communities, which is very difficult to be obtained in real-world networks.

To overcome these drawbacks, this paper proposes a new community detection algorithm based on Differential Evolution (DE), named DECD. To the best of our knowledge, it is the first time DE is introduced for community detection. In DECD, DE is used to evolve a population of potential solutions for network partitions to maximize the network modularity [20]. It is worth mentioning that DECD does not require any prior knowledge about the community structure when detecting communities in networks, which is is beneficial for its applications to real-world problems where prior knowledge is usually not available.

Apart from introducing DE for community detection, other key contributions of this paper include: 1) the design of an improved version of the standard binomial crossover in DE to transmit some important information about the community structure during evolution in DECD; 2) a biased process and a clean-up operation similar to [31] is introduced to DECD to improve the quality of the individuals in the population; 3) a thorough evaluation of the performance of DECD on artificial and two real-world social networks, which achieved better results than other state-of-the-art community detection algorithms. 4) the application of DECD to a Yeast interacting protein dataset [10], which achieve the best results in the literature.

The remainder of this paper is organized as follows. Section 1.1 introduces some basic ideas of DE. In Section 1.2, some of the most popular algorithms for community detection are briefly reviewed. Section 2 presents a detailed description of DECD. In Section 3, the performance of DECD is tested on artificial and real-world networks and then the experimental results are discussed. Finally, Section 4 concludes this paper.

1.1 Differential Evolution

Differential evolution (DE) is a very simple yet efficient evolutionary algorithm proposed by Storn and Price in 1995 [29]. DE starts the search with an initial population containing NP individuals randomly sampled from the search space. Then, one individual called the target vector in the population is used to generate a mutant vector by the mutation operation. The most popular mutation strategy [17,18] which is also employed in DECD is the "rand/1" strategy as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \times (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}), \tag{1}$$

where $i \in \{1, 2, \ldots, NP\}$, $r1$, $r2$ and $r3$ are integers randomly selected from $1, 2, \ldots, NP$ and satisfy $r1 \neq r2 \neq r3 \neq i$, the scaling factor F is usually a real number between 0 and 1, the decision vector $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$ with n decision variables is the individual in the population and also called the target vector, and $\boldsymbol{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n})$ is the mutant vector.

After mutation, all the components of the mutant vector are checked whether they violate the boundary constraints. If the jth component $v_{i,j}$ of the mutant vector $\boldsymbol{v}_i$ violates the boundary constraint, $v_{i,j}$ is reflected back from the violated boundary constraint as follows [14]:

$$v_{i,j} = \begin{cases} 2LB_j - v_{i,j}, & \text{if } v_{i,j} < LB_j \\ 2UB_j - v_{i,j}, & \text{if } v_{i,j} > UB_j \\ v_{i,j} & \text{otherwise ,} \end{cases} \tag{2}$$

where LB_j and UB_j are the lower and upper bounds of the ith decision variable x_i, respectively.

Subsequently, the crossover operation is implemented on the mutant vector $\boldsymbol{v}_i$ and the target vector $\boldsymbol{x}_i$ to generate a trial vector $\boldsymbol{u}_i$. A commonly used crossover operation is the binomial crossover which is executed as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise ,} \end{cases} , \tag{3}$$

where $i \in 1, 2, \ldots, NP$, $j \in 1, 2, \ldots, n$, $rand$ is a uniformly distributed random number between 0 and 1, j_{rand} is a randomly selected integer from 1 to n, CR is the crossover control parameter, and $u_{i,j}$ is the jth component of the trial vector $\boldsymbol{u}_i$.

Finally, the target vector $\boldsymbol{x}_i$ is compared with the trial vector in terms of the objective function value and the better one survives into the next generation:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & \text{otherwise.} \end{cases} \tag{4}$$

1.2 Related Work

During the past decade, the research on analyzing the community structure in complex networks has drawn a great deal of attention. Meanwhile, various kinds of algorithms have been proposed. Some of the most known algorithms are reviewed as follows.

Girvan and Newman [12] proposed the Girvan-Newman (GN) algorithm which is one of the most known algorithms proposed so far. This algorithm is a divisive method and iteratively removes the edges with the greatest betweenness value based on betweenness centrality [9]. Newman [19] 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 network modularity at each step. Moreover, based on the original strategies, its faster version [4] was proposed by using some shortcuts and some sophisticated data structures. Radicchi et al. [24] presented the definitions of communities in both a strong sense and a weak sense. Moreover, in their paper a division algorithm [24] was proposed to detect communities by removing edges with the smallest value of edge cluster coefficient. Duch and Arenas [6] presented a division method which uses a heuristic search based on the extremal optimization to optimize the network modularity to detect communities in networks. Rosvall and Bergstrom [25] developed an algorithm based on an information-theoretic framework which identifies the communities by finding an optimal compression of the topology and capitalizing on regularities in the structure of networks.

However, some of the above community detection algorithms have large computational complexity and are unsuitable for very large networks. Moreover, a priori knowledge about the community structure (e.g., the number of communities) which is not easy or impossible to obtain in real-world networks is also required in most of the above algorithms [31]. To overcome the drawbacks, algorithms based on evolutionary algorithms have been proposed. These algorithms are very effective for community detection especially in very large complex networks. Tasgin and Bingol [31] presented an approach based on a genetic algorithm to optimize the network modularity in order to find community structures in networks. Pizzuti [21] proposed a method based on a genetic algorithm to discover communities in networks. This method defines the community score to measure the quality of a partitioning in communities of networks and uses a genetic algorithm to optimize the community score. Chen et al. [3] presented an algorithm based on the immune clone selection algorithm which is employed to optimize the modularity density [15] to identify communities in networks.

2 The Proposed Algorithm

In this paper, a new algorithm based on DE called DECD is proposed for community detection in complex networks. DECD uses DE as the search engine and employs the network modularity as the fitness function to evolve the population. Next, DECD is described in detail.

2.1 Individual Representation

DECD uses the community identifier-based representation proposed in [31] to represent individuals in the population for the community detection problem. For a graph $G = (V, E)$ with n nodes modeling a network, the kth individual in the population is constituted of n genes $\boldsymbol{x}_k = \{x_1, x_2, \ldots, x_n\}$ in which each gene x_i can be assigned an allele value j in the range $\{1, 2, \ldots, n\}$. The gene and allele represent the node and the community identifier (commID) of communities in G respectively. Thus, $x_i = j$ denotes that the node i belongs to the community whose commID is j, and nodes i and d belong to the same community if $x_i = x_d$. Since DECD puts nodes in communities randomly when initializing, at most n communities exist in G and then the maximum value of commID is n.

In the above representation, all the communities in G and all the nodes belonging to each community can be identified straightforwardly from individuals in the population. The community identifier-based representation is very simple and effective. Moreover, the number of communities is automatically determined by the individuals and no decoding process is required in this representation.

For example, Figure 1.(a) shows a network containing 12 nodes numbered from 1 to 12. According to the definition of the community structure, the network is divided into three communities visualized by different colors of nodes. Figure 1.(b) is the genotype of the optimal solution for the community structure of the network, while the graph structure of the genotype is given in Figure 1.(c).

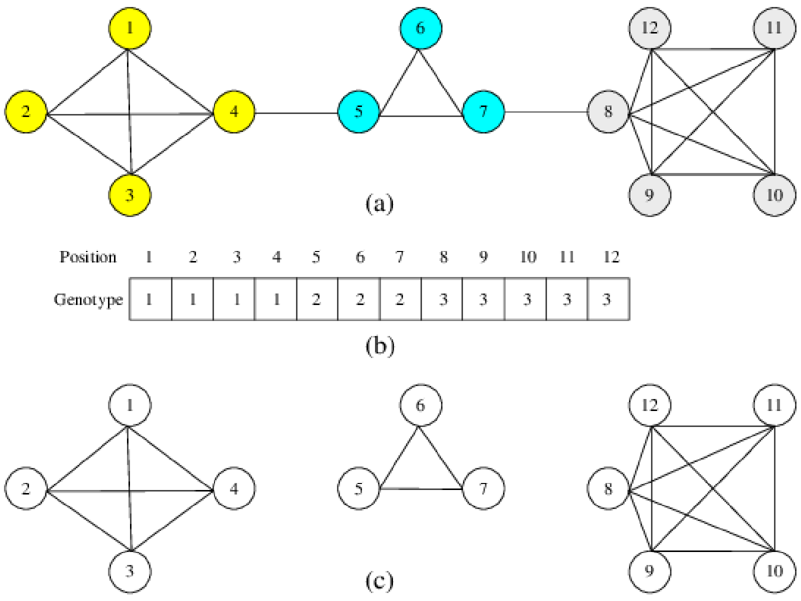
(a)
Position 1 2 3 4 5 6 7 8 9 10 11 12
Genotype 1 1 1 1 2 2 2 3 3 3 3 3
(b)
(c)

Fig. 1. (a) a graph model of a network; (b) the community identifier-based representation of a genotype; (c) the graph structure of the genotype

2.2 Fitness Function

Newman and Girvan [20] proposed the network modularity to measure the strength of the community structure found by algorithms. The network modularity is a very efficient quality metric for estimating the best partition of a network into communities. It has been used by many community detection algorithms recently [4,19,31].

DECD also employs the network modularity which is maximized as the fitness function to evaluate individuals in the population. The network modularity is defined as follows [31].

$$Q = \sum_{j=1}^{m} \left[\frac{l_j}{L} - \left(\frac{d_j}{2L} \right)^2 \right], \tag{5}$$

where j is the commID, m is the total number of communities, l_j is the number of links in module j, L is the total number of edges in the network and d_j is the degree of all nodes in module j.

2.3 Initialization

At the beginning of the initialization process, DECD places each node into a random community by assigning a random commID and generates individuals in the initial population.

However, the above random generation of individuals is likely to cause some unreasonable results such that a community contains some nodes having no connectivity with each other in the original graph. Considering that nodes in the same community should connect with each other and in the simple case are neighbors, a biased process [31] is used to overcome the above drawbacks, that is, once an individual is generated, some nodes represented by genes in the individual are randomly selected and their commIDs are assigned to all of their neighbors. By the biased process, the space of the possible solutions is restricted and the convergence of DECD is improved. Through these operations, the initial population P_0 is generated.

2.4 Mutation

DECD employs the "rand/1" strategy to mutate individuals in the population. The "rand/1" strategy is a very efficient mutation strategy. It has no bias to any special search directions and chooses new search directions in a random manner by randomly selecting individuals for mutation [33].

When implementing the "rand/1" strategy, firstly three different individuals $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$ and $\boldsymbol{x}_{r3}$ are randomly selected from P_t, where $r1, r2, r3 \in \{1, \ldots, NP\}$, NP is the population size and t is the generation number. Then these three individuals follow the equation (1) and generate a mutant vector $\boldsymbol{v}$ which is put into the mutant population V_t. The above two steps are executed iteratively until the population size of V_t is NP.

Subsequently, all the components of each mutant vector in V_t are checked whether they violate the boundary constraints. If violating the boundary constraint, the component is reflected back from the violated boundary by following the equation (2). Then, a mutant population V_t satisfying all the boundary constraints is obtained.

2.5 Crossover

Since DECD randomly assigns an integer in the range $\{1, 2, \ldots, n\}$ to each individual in the population as its commID, no one-to-one absolute corresponding relationship exists between the communities and commIDs. That is, for different individuals the same community may have different commIDs while the same commID is likely to represent different communities. For example, with respect to two individuals $(1, 2, 1, 3)$ and $(2, 1, 2, 3)$, the community with the commID equals to 1 for the first individual and the one with the commID equals to 2 for the second individual are the same community. Meanwhile, the communities with the commID equals to 1 for these two individuals are different although they have the same commID.

For individuals represented by the community identifier-based representation, the traditional crossover operators (e.g., the binomial crossover) do not work well. This is because they just simply change the commIDs and never consider nodes in their communities. As a result, the offspring individuals fail to inherit good genes from the parent individuals and the search ability is heavily impaired.

Considering the above reasons, DECD designs a modified binomial crossover based on the binomial crossover to enhance the search ability. Inspired by the modified crossover operation [31], the modified binomial crossover assigns the commIDs of some nodes in an individual to those corresponding nodes in another individual. The implementation of the modified binomial crossover is as follows.

Firstly, the trial vector $\boldsymbol{u}_i = \boldsymbol{x}_i$ is set for every $i \in 1, 2, \ldots, NP$. Subsequently, the jth components in $\boldsymbol{u}_i$ and the mutant vector $\boldsymbol{v}_i$ are considered for every $j \in \{1, 2, \ldots, n\}$ and every $i \in \{1, 2, \ldots, NP\}$. If $rand \leq CR$ or $j = j_{rand}$, all the nodes in the community whose commID is $v_{i,j}$ of $\boldsymbol{v}$ are found, and then the commIDs of all those corresponding nodes of $\boldsymbol{u}_i$ are assigned the value $v_{i,j}$ which means all those corresponding nodes of $\boldsymbol{u}_i$ are put into the community whose commID is $v_{i,j}$; otherwise, no operation will be performed on $\boldsymbol{u}_i$. Therein, $rand$ is a uniformly distributed random number between 0 and 1, j_{rand} is a randomly selected integer from 1 to n, and CR is the crossover control parameter. Finally, the trial vectors $U_t = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{NP}\}$ are obtained.

From the above process, it can be seen that the trial vectors are able to obtain some useful information about the community structure from both the target vectors and the trial vectors. Therefore, the modified binomial crossover is very helpful for identifying communities in networks.

2.6 Clean-Up Step

Since DE is a stochastic optimization algorithm, the solutions for the community division are likely to have some mistakes in evolution, that is, some nodes may be put into wrong communities. These mistakes impair the search ability of DECD and make it get stuck in a local optimum, ultimately leading to community divisions with inferior quality.

To solve the above problem, DECD adopts the clean-up operation proposed by Tasgin and Bingol [31], which effectively corrects the mistakes of putting nodes into wrong communities in both mutant and trial vectors and improves the search ability of. The clean-up operation is based on the community variance $CV(i)$, which is defined as the fraction of the number of different communities among the node i and its neighbors to the degree of the node i as follows:

$$CV(i) = \frac{\sum_{(i,j)\in E} \mathrm{neq}(i,j)}{\deg(i)}, \tag{6}$$

where $\mathrm{neq}(i,j) = \begin{cases} 1, & \text{if commID}(i) \neq \text{commID}(j) \\ 0, & \text{otherwise} \end{cases}$, $\deg(i)$ is the degree of the ith node, E is the set of edges, and commID is the community containing ith node.

According to the classic definition of the community structure, a community should contain more internal edges among nodes inside the community than external edges with other communities. Thus, a node and all its neighbors should be in the same community with a high probability, and then the community variance of this node should be low in a good community division. Based on the above analysis, the clean-up operation is performed as follows. Firstly, some nodes are randomly selected. Then, for each of these nodes, its community variance is computed and compared with a threshold value η which is a predefined constant obtained after some experiments. If the community variance of this node is larger than this threshold value η, which indicates that the node has been put into a wrong community, then this node and all its neighbors are placed into the same community containing the highest number of nodes in the neighborhood of this node. Otherwise, no operation is executed for this node.

2.7 DECD Algorithm Framework

Finally, the framework of DECD is described as follows:

Step 1) Set $t = 0$ where t denotes the generation number.

Step 2) Generate the initial population $P_0 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{NP}\}$ by uniformly and randomly sampling NP points from the search space S.

Step 3) Compute the network modularity value $Q(\boldsymbol{x}_i)$ of each individual $\boldsymbol{x}_i$ in P_0.

Step 4) Perform the mutation operation (see Section 2.4 for details) on each individual $\boldsymbol{x}_i$ in P_t and obtain the mutant vectors $V_t = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{NP}\}$.

Step 5) Correct the mistakes in each mutant vector $\boldsymbol{v}_i$ in V_t by executing the clean-up operation (see Section 2.6 for details).

Step 6) Execute the modified binomial crossover (see Section 2.5 for details) on each mutant vector $\boldsymbol{v}_i$ in V_t and generate the trial vectors $U_t = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{NP}\}$.

Step 7) Correct the mistakes in each trail vector $\boldsymbol{u}_i$ in U_t by executing the clean-up operation (see Section 2.6 for details).

Step 8) Calculate the network modularity value $Q(\boldsymbol{u}_i)$ of each trial vector $\boldsymbol{u}_i$ in U_t.

Step 9) Compare $\boldsymbol{x}_i$ with $\boldsymbol{u}_i$ $(i = 1, \ldots, NP)$ in terms of the network modularity value by following the equation (4), and put the winner into the next population P_{t+1}.

Step 10) Set $t = t + 1$.

Step 11) If the termination criterion is not satisfied, go to **Step 4**; otherwise, stop and output the best individual $\boldsymbol{x}_{best}$ in P_t.

3 Experiments and Results

In this section, the performance of DECD is tested on a class of widely used artificial networks and some well studied real world social and biological networks. DECD is implemented in MATLAB and all the experiments are performed on Windows XP SP2 with Pentium Dual-Core 2.5GHz processor and 2.0GB RAM. The parameters in DECD are set as follows: the population size $NP = 200$, the scaling factor $F = 0.9$, the control crossover parameter $CR = 0.3$, the threshold value $\eta = 0.35$ and the maximum number of generations is 200. In each run, DECD is stopped when the maximum number of generations is reached.

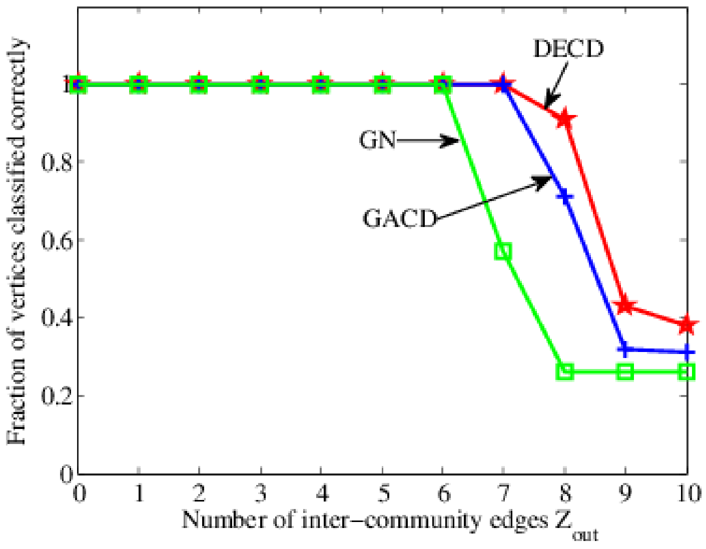

Fig. 2. The fraction of vertices correctly classified by DECD, GACD and GN as the average z_{out} of inter-community edges per vertex is varied for the computer-generated networks

For comparison, we implement another community detection algorithm based on GA, named GACD. We adopt the MATLAB Genetic Algorithm Optimization Toolbox (GAOT) to optimize the network modularity to detect communities in networks. The GA we use is real encoded GA with heuristic crossover and uniform mutation. Moreover, for the sake of fairness, the same biased process and the clean-up operation in DECD are employed. The values of all the parameters use in the experiments are the default parameters in GAOT. We also adopt MATLAB implementations of Girvan-Newman (GN) algorithm from Matlab Tools for Network Analysis (http://www.mit.edu/~gerganaa) for comparison.

3.1 Artificial Networks

To evaluate the performance of DECD of detecting network community structure, artificial computer-generated networks are employed. The computer-generated networks were proposed by Girvan and Newman and have been widely used to benchmark the community detection algorithms [12]. 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 $z_{in} + z_{out} = 16$. The community detection algorithms with good performance should discover all the communities in the network with $z_{in} > z_{out}$ which indicates that the neighbors of a node inside its community are more than the neighbors contained by the other three communities. According to the definition of the community structure, the community structure in the network becomes vaguer with the z_{out} increases.

Figure 2 summarizes the experimental results and shows the fraction of nodes correctly divided into the four communities with respect to z_{out} by DECD, GACD and GN [12], respectively.

From Figure 2, it can be seen that DECD performs significantly better than GN when $z_{out} > 6$ while they perform the same when $z_{out} \leq 6$. Comparing with GACD, DECD also performs much better when $z_{out} > 7$ and they have the same performance when $z_{out} \leq 7$. The results show that DECD is very effective for detecting communities in networks, even those with very vague community structures.

3.2 Real-World Social Networks

In this paper, two real-world social networks, i.e., the Zachary's karate club network [36] and the American college football network cite8Girvan2002, are also employed to further verify the performance of DECD. Both these two real-world social networks are well-known benchmark examples for the community detection algorithms and have been well studied in the literatures. Both networks have known (true) community structures, which provide gold-standard for validating our DECD algorithm.

The first social network is the Zachary's karate club network, which shows the friendships between the members of a karate club at an University. The network contains 34 nodes and 78 edges. The most interesting feature of the network is that the club split into two as a result of an internal dispute, which provides a ground-true community division of the network.

The American college football network [12] has a known community structure. The network is a representation of the schedule of Division I games for the 2000 games. In the football network, there are 115 nodes and 616 edges divided into 12 communities, where nodes and edges represent the teams (identified by their college names) and the regular season games between the two teams they connect, respectively. The teams are divided into "conferences" and each conference contains around 8 to 12 teams. The teams play an average of about 4 inter-conference games and 7 intra-conference games which indicates that games are more frequent between members of the same conference than between members of different conferences.

As pointed out in [28] and [34], performance metrics based on network modularity Q is not reliable. Therefore, apart from Q, we also adopt accuracy as a quantitative measure for validating DECD as used in [28]:

$$\text{Accuracy} = \frac{\sum_{k=1}^{n} \text{equal}(t_k, p_k)}{n}, \tag{7}$$

where

$$\text{equal}(x, y) = \begin{cases} 1, & \text{if commID}(x) = \text{commID}(y) \\ 0, & \text{otherwise} \end{cases},$$

and t_k is the kth node in the true (known) network structure, and p_k is the kth node in the predicted network structure.

Since DECD and GACD are stochastic optimization algorithms, we perform the experiments 30 times on these two networks. The average values Q_{avg} and Acc_{avg} of Q

and accuracy and their best values Q_{bst} and Acc_{bst}, are compared with that obtained by GN (a deterministic algorithm) from one run of experiment in Table 1. In order to run the GN algorithm, we need to specify the number of communities. In our experiment, we use the number of known communities in the two networks, e.g., 2 for the karate network and 12 for the football network as number of communities for the GN algorithm.

Table 1. Experimental results of the Zachary's karate club network and the American college football network. N_{pr} is the average number of communities; Q_{avg} and Q_{bst} are the average and best values of modularity Q, respectively; and Acc_{avg} and Acc_{bst} are the average and best accuracy, respectively.

Network	Algorithm	N_{pr}	Q_{avg}	Q_{bst}	Acc_{avg}	Acc_{bst}
Karate	DECD	3.467 ± 0.730	$\mathbf{0.385 \pm 0.013}$	$\mathbf{0.416}$	$\mathbf{0.972 \pm 0.009}$	$\mathbf{1}$
	GACD	2.900 ± 1.094	0.369 ± 0.022	0.402	0.959 ± 0.020	0.971
	GN	2	0.360	0.360	0.971	0.971
Football	DECD	11.233 ± 0.504	$\mathbf{0.596 \pm 0.003}$	$\mathbf{0.605}$	$\mathbf{0.930 \pm 0.004}$	$\mathbf{0.939}$
	GACD	8.767 ± 1.073	0.587 ± 0.015	$\mathbf{0.605}$	0.913 ± 0.010	0.930
	GN	12	0.455	0.455	0.904	0.904

From Table 1, it can be seen that DECD perform better than the two competitors, i.e., GACD and GN on both the karate and football networks. It is interesting to see that, for the smaller scale karate network, the performance of DECD is just slightly better than GN. However, for the larger football network, the performance of DECD is significantly better than GN. Such results indicate that DECD is more effective in detecting communities in larger complex networks than GN.

In [28], the authors tested several state-of-the-art community detection algorithms. In the paper, for the karate network, the best algorithm in terms of Q was Fast GN (denoted as FastQ in their paper) [4], which generated Q value of 0.381 from one run of experiment, which is slightly better than that from the GN algorithm used in our paper but still worse than that from our DECD. In terms of accuracy, their best algorithm (Random walks) achieved 1, which is as same as Acc_{bst} found by our DECD in 30 runs. In [28], for football network, the best result in terms of Q was 0.604 (WalkTrap [22]), which is slightly worse than Q_{bst} generated by our DECD. In terms of accuracy, DECD also generated the same $Acc_{bst} = 0.939$ as the best result in [28], which was generated by Random walks and MCL [32].

3.3 Yeast Protein-Protein Interaction Network

We apply our DECD algorithm to a Yeast Protein-Protein Interaction (PPI) Network [11], which contains 1430 proteins and 6535 interactions. We use CYC2008 [23], a complete and up-to-date set of yeast protein complexes (also called modules or communities) as reference set to evaluate the predicted modules by DECD (In the following text, we will use module and complex instead of community, which is a less popular term in bioinformatics). We compute precision, recall and F-measure to measure the

Table 2. Experimental results of the Yeast Protein-Protein Interaction Network

ω	Algorithm	#. pred. complex	Precision	Recall	F-measure
0.2	DECD	115	**0.6696**	**0.4976**	**0.5709**
	GACD	106	0.6132	0.4902	0.5488
	GECSS	157	0.5063	0.2802	0.3608
	MCODE	39	0.2278	0.4871	0.3104
	MCL	200	0.5063	0.2050	0.2918
0.5	DECD	115	0.4696	**0.2390**	**0.3168**
	GACD	106	0.4340	0.2220	0.2937
	GN	65	0.5231	0.0568	0.1025
	CMCD	65	**0.6154**	0.0691	0.1242

performance of DECD. The performance of DECD is compared with GACD and GN. We also adopt results from recent literature, e.g., [35,27] for comparison.

Similar to the experiments in [35,27], we use affinity score to decide whether a predicted module is matched with a reference complex:

$$\text{Affinity}(A, B) = \frac{|A \bigcup B|^2}{|A| \times |B|}, \tag{8}$$

where A and B are two modules of proteins, e.g., one of predicted module or reference complexes. We assume a module A matches module B if and only if $\text{Affinity}(A, B)$ is above a predefined threshold ω. Then we can define $Hit(\mathcal{A}, \mathcal{B})$ which contains all the matched modules:

$$Hit(\mathcal{A}, \mathcal{B}) = \{A_i \in \mathcal{A} | \text{Affinity}(A_i, B_j) > \omega, \exists B_j \in \mathcal{B}\}. \tag{9}$$

We define precision, recall and F-measure as follows:

$$\text{Recall} = \frac{|Hit(\mathcal{R}, \mathcal{P})|}{|\mathcal{R}|}, \tag{10}$$

$$\text{Precision} = \frac{|Hit(\mathcal{P}, \mathcal{R})|}{|\mathcal{P}|}, \tag{11}$$

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{12}$$

where $\mathcal{P}$ is the predicted module set and $\mathcal{R}$ is the reference complex set.

Following the experimental settings in [35,27], we set $\omega = 0.2$ and 0.5 in order to compare with their algorithms fairly. We adopt the results of algorithms tested in [35,27], which include Critical Module Community Detection algorithm (CMCD) [27], Gene Expression Condition Set Similarity (GECSS) algorithm [35], Molecular Complex Detection (MCODE) algorithm [2] and Markov clustering (MCL) algorithm [32] . It is worth mentioning that, due to the large size of the PPI network, the GN algorithm in Matlab Tools for Network Analysis (http://www.mit.edu/˜gerganaa) failed to produce results in reasonable time. Therefore, we adopt the results of the GN algorithm from [27] for comparison.

From Table 2, we can see that compared with GACD and other algorithms tested in [35,27], DECD has better performance. It is interesting to see that, the performance gap between DECD and GACD is not as significant as those between DECD and other non-population-based algorithms, e.g., MCL. Such results indicate that, at least for medium size networks, population-based algorithms are preferred because of their better predictive performance.

4 Conclusion

In this paper, we have introduced differential evolution (DE) to detect community structure in complex networks. To the best of our knowledge, it is the first time DE has been applied to community detection problems. The proposed algorithm DECD uses DE to search the best network partition of a complex network that can achieve an optimal network modularity value. Based on the standard binomial crossover of DE, we designed a modified binomial crossover to transmit some important information about the community structure during evolution. We have also introduced a biased process and a clean-up operation similar to [31] to improve the quality of the individuals in the population.

We have tested our DECD on the artificial networks and the real-world social and biological networks in comparison with GACD and GN algorithms. Apart from the modularity value, for the real-world networks, we have also employed accuracy based on true community structure as a performance metric [28], which provided reliable performance information. The experimental results have demonstrated that DECD is very effective for community detection in complex networks, including those with very vague community structures, e.g., the artificial networks with larger values of z_{out}. In addition to its excellent performance, another merit of DECD is that it does not require any prior knowledge about the community structure when detecting communities in networks.

The limitation of this work is that we only used modularity as the objective function to find the optimal community structure of a network. However, it has been recently pointed out that such approach might suffer from the so-called resolution limit problem, that is, some modules smaller than a specific scale will not be detected by the algorithms that only optimize modularity [8]. Although we have achieved better community detection results than many other algorithms on 4 complex networks, we do not anticipate DECD can avoid this resolution limit problem. We will further investigate this problem in our future work.

Acknowledgment. This work is sponsored by ESPRC (EP/J501414/1). We acknowledge the financial support of the Systems Science for Health initiative of the University of Birmingham.

References

1. Albert, R., Jeong, H., Barabasi, A.: Error and attack tolerance of complex networks. Nature 406, 378–382 (2000)
2. Bader, G.D., Hogue, C.W.V.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4 (2003)

3. Chen, G., Wang, Y., Yang, Y.: Community detection in complex networks using immune clone selection algorithm. International Journal of Digital Content Technology and its Applications 5, 182–189 (2011)
4. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70, 066111 (2004)
5. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of networks. Adv. Phys. 51, 1079 (2001)
6. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical Review E 72, 027104 (2005)
7. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. ACM SIGCOMM Computer Communications Review 29, 251–262 (1999)
8. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences 104, 36–41 (2007)
9. Freeman, L.: A set of measures of centrality based upon betweenness. Sociometry 40, 35–41 (1977)
10. Gavin, A.C., et al.: Proteome survey reveals modularity of the yeast cell machinery. Na 440, 631–636 (2006)
11. Gavin, A.C., et al.: Proteome survey reveals modularity of the yeast cell machinery. Nature 440, 631–636 (2006)
12. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99, 7821–7826 (2002)
13. Guimera, R., Amaral, L.: Functional cartography of complex metabolic networks. Nature 433, 895–900 (2005)
14. Kukkonen, S., Lampinen, J.: Constrained real-parameter optimization with generalized differential evolution. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006). IEEE Press, Sheraton Vancouver Wall Centre Hotel, Vancouver (2006)
15. Li, Z., Zhang, S., Wang, R., Zhang, X., Chen, L.: Quantitative function for community detection. Physical Review E 77, 036109 (2008)
16. Liu, Y., Slotine, J., Barabasi, A.: Controllability of complex networks. Nature 473, 167–173 (2011)
17. Mezura-Montes, E., Miranda-Varela, M., Gómez-Ramón, R.: Differential evolution in constrained numerical optimization: An empirical study. Information Sciences 180, 4223–4262 (2010)
18. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106 (2010)
19. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review E 69, 026113 (2004)
20. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69, 026113 (2004)
21. Pizzuti, C.: GA-Net: A Genetic Algorithm for Community Detection in Social Networks. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 1081–1090. Springer, Heidelberg (2008)
22. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. of Graph Alg. and App. Bf 10, 284–293 (2004)
23. Pu, S., Wong, J., Turner, B., Cho, E., Wodak, S.J.: Up-to-date catalogues of yeast protein complexes. Nucleic Acids Res. 37, 825–831 (2009)
24. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences 101, 2658–2663 (2004)
25. Rosvall, M., Bergstrom, C.: An information-theoretic framework for resolving community structure in complex networks. Proceedings of the National Academy of Sciences 104, 7327–7331 (2007)

26. Scott, J.: Social network analysis: A Handbook. Sage Publications, London (2000)
27. Sohaee, N., Forst, C.V.: Modular clustering of protein-protein interaction networks. In: 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB (2010)
28. Steinhaeuser, K., Chawla, N.V.: Identifying and evaluating community structure in complex networks. Pattern Recognition Letters 31, 413–421 (2009)
29. Storn, R., Price, K.: Differential evolution a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
30. Strogatz, S.H.: Exploring complex networks. Nature 410, 268–276 (2001)
31. Tasgin, M., Bingol, H.: Community detection in complex networks using genetic algorithm. In: Proceedings of the European Conference on Complex Systems (2006)
32. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
33. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation 15, 55–66 (2011)
34. Yang, Y., Sun, Y., Pandit, S., Chawla, N.V., Han, J.: Is objective function the silver bullet? a case study of community detection algorithms on social networks. In: International Conference on Advances in Social Network Analysis and Mining, pp. 394–397 (2011)
35. Yeu, Y., Ahn, J., Yoon, Y., Park, S.: Protein complex discovery from protein interaction network with high false-positive rate. In: Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics 2011, EvoBio 2011 (2011)
36. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977)