information

MDPI

Article

Community Detection Based on Differential Evolution Using Modularity Density

Caihong Liu [1,*] and Qiang Liu [2]

[1] College of Software, Dalian University of Foreign Languages, Dalian 116041, China
[2] School of Computer Science, National University of Defense Technology, Changsha 410073, China; liuqiang1981@nudt.edu.cn
* Correspondence: lchjsj@126.com

Received: 25 June 2018; Accepted: 23 August 2018; Published: 30 August 2018

Abstract: Currently, many community detection methods are proposed in the network science field. However, most contemporary methods only employ modularity to detect communities, which may not be adequate to represent the real community structure of networks for its resolution limit problem. In order to resolve this problem, we put forward a new community detection approach based on a differential evolution algorithm (CDDEA), taking into account modularity density as an optimized function. In the CDDEA, a new tuning parameter is used to recognize different communities. The experimental results on synthetic and real-world networks show that the proposed algorithm provides an effective method in discovering community structure in complex networks.

Keywords: complex networks; community detection; differential evolution; modularity density

1. Introduction

Recently, an increasing number of researchers are paying attention to the community structure identification [1–3] of networks, including social networks [4–7], biology networks [6,8], worldwide web networks [9,10], and so on, in order to have a profound understanding of their structures and functions. A complex network is a representation of a complex system in real life in the form of nodes and edges, where the nodes represent objects and the edges represent their mutual interactions. Understanding the relationship between these nodes and edges plays an important role in shedding light on the overall organization of complex systems and their functional properties.

Complex networks possess some important properties after an enormous amount of theoretical and practical investigations [1–10], of which community structure became a dominant research direction for the purpose of getting valuable information from the networks. For instance, the division of a network into groups has dense intra-connections, and sparse inter-connections [11,12]. Then, many methods for community detection were proposed by maximizing internal links and minimizing external links. One of most famous methods, called network modularity Q [13], is used as a quality metric for measuring the partition of networks. According to this measurement, many successful algorithms were proposed by optimizing the modularity Q function, such as those described in References [2,5,14–16]. However, the search for the largest modularity value is an NP-hard problem, since the space of all possible partitions increases quickly compared to the complex system scale. That is to say, modularity is not a scale-invariant measure, and hence, finding communities smaller than a specialized size is impossible. This serious problem is famously known as the resolution limit problem of modularity-based algorithms [17]. In order to resolve this disadvantage, Li et al. [18] proposed a new quality function named modularity density, and demonstrated its superiority in detecting communities compared to traditional modularity-based methods. The main difference between modularity and modularity density lies in the fact that they have different referential objects

in their functions. Modularity measures the strength of a community partition by taking into account the degree distribution, while modularity density adopts the internal node distribution of a community. Since modularity density was proven to resolve the resolution limit problem of the modularity function, we introduce it in our paper and propose a community detection algorithm based on a differential evolution algorithm using modularity density.

The differential evolution (DE) algorithm, proposed by Storn and Price [19], is a new population-based stochastic search algorithm in the form of parallel ways. In contrast to traditional evolutionary methods such as genetic algorithms (GAs), the DE algorithm has at least three merits: finding the true global optimized value of a multimodal search space regardless of initial parameter values, fast convergence, and using few control parameters [19]. Additionally, the DE algorithm uses the genetic information of several individuals in a mutation scheme, which can greatly utilize the distributed characteristics of a population and effectively improve the search ability. Therefore, the DE shows remarkable performance in a wide variety of fields [20,21]. In particular, it exhibited some great merits in optimization problems [22–24]. The performance of the DE algorithm lies in two important steps. The first is the mutation scheme and the second is the setting of control parameters, such as population size, crossover size, and the scale factor. In order to solve a problem efficiently, proper parameters should be set.

As a new research direction in complex networks, the application of evolutionary methods in complex networks attracted a large number of researchers, such as the authors of References [2,3,16,21,23,25,26]. In the field of community detection, modularity-based optimization methods are now widely used. However, their optimization is often a typical NP-hard problem, whereby the solutions obtained using traditional methods can only find local optimal solutions, which cannot reflect the global characteristic of the problem. However, optimization methods based on biological evolution and swarm intelligence can easily get global and local search capabilities and yield high-quality globally optimal solutions. For instance, GA, an example of a conventional representative method of evolutionary computation, was successfully applied to community detection. Tasgin et al. [25] and Shang et al. [16] proposed GA-based methods using modularity, Pizzuti proposed a GA-Net algorithm based on community score, Jia et al. [22] proposed a differential evolution for community detection adopting modularity, and Gong et al. [2,27] proposed a memetic algorithm (Meme-Net) and an improved Meme-Net algorithm for community detection in networks based on GA. However, recent research [26] shows that DE is a faster, and more efficient and robust metaheuristic optimization technique than GA.

In this paper, we put forward a community detection method based on the differential evolution algorithm (CDDEA), which adopts modularity density as an optimization function and can explore the network in different resolutions. It is worth mentioning that CDDEA does not need any prior knowledge of community structure, such as the number of communities, in the course of searching better community partitions, and this point plays an important role in real-world problems in the case of an absence of related prior information.

The remainder of this paper is organized as follows: the community detection problem and related concepts are reviewed in Section 2. The community detection algorithm based on the differential evolution algorithm using modularity density is proposed and explained in Section 3. In Section 4, the proposed method is tested on computer-generated and real-world networks, and the experimental results of CDDEA are in comparison with other benchmark algorithms. Section 5 concludes this paper.

2. Related Concepts

This section first presents a comparison between modularity and modularity density in order to show the superiority of adopting modularity density. Next, an introduction to normalized mutual information (NMI) is established so as to lay a solid foundation for evaluating the effectiveness of our proposed algorithm.

Community detection in complex networks attracted a great many researchers all over the world, and an enormous number of effective methods were proposed in recent years [28], including hierarchical clustering [12,29], spectral clustering [30–32], and modularity optimization [5,14], which are currently the widely used community detection methods [28].

Hierarchical clustering methods: In order to examine communities in a different granularity, Newman [12] proposed a method to find weak ties based on edge betweenness, and then to remove those with highest betweenness in order to detect communities in a hierarchical fashion. In this way, we must recompute all elements of edge betweenness in the network after removing an edge, which leads to high computational cost. Blonder et al. [29] put forward a Louvain algorithm, merging two communities into one if this combination could improve the modularity of the network.

Spectral clustering methods: By introducing a graph partition method to community detection, this method repeatedly separates the whole network into two subnetworks until the number of subnetworks is equal to the initially designated value. Corneil et al. [33] proposed a Kernighan–Lin algorithm, which adopted a greedy search manner in order to find better network partitions according to maximizing gain-function value. Pothen et al. [31] put forward a spectral bisection algorithm based on a Laplacian matrix eigenvalue of the network. The limits of these two spectral clustering methods and their extension lie in whether we can get proper partition results using the bisection method. Moreover, it is hard to know the exact size of two subnetworks in initial conditions, which leads to a poor performance in acquiring correct partition results. Nascimento [32] proposed a spectral heuristic method, based on a measure known as the clustering coefficient, to detect communities in networks, which can also overcome the scale deficiency of the modularity maximization problem.

Modularity optimization methods: With an increasing number of community detection methods being proposed, it is often difficult to give a fair evaluation of them. Newman and Grivan [14] proposed a modularity measure function in 2004, which is widely used in current community detection methods. The higher the modularity value of a kind of community partition, the better this kind of partition is. Therefore, community detection problems can be converted into optimization problems. However, it was proven that modularity optimization is a complete NP problem. Most algorithms based on modularity optimization show a high computational complexity, such as SA [34], EO [35], MIQP [36], GA [25] and so on. In the research of Fortunate and Barthelemy [17], the resolution limit problem of the modularity in community partition was highlighted. They attributed the essence of this problem to the improper hypothesis that the modularity uses a reference of a random network with properties of the same degree distribution. Compared to traditional modularity-based algorithms, this paper uses modularity density as a quality function to guide the selection of better communities in CDDEA.

2.1. Modularity and Modularity Density

Modularity [14] can be calculated as the function described below. Consider an unsigned network denoted as $G = (V, E)$, where V is the vertex set with a number n, and E is the edge set with a number e. The adjacent matrix of G is A. If V_1 and V_2 are two disjoint subsets of V, then we define $L(V_1, V_2) = \sum_{i \in V_1, j \in V_2} A_{ij}$, $L(V_1, V_1) = \sum_{i \in V_1, j \in V_1} A_{ij}$, and $L(V_1, \overline{V_1}) = \sum_{i \in V_1, j \notin V_1} A_{ij}$, where $\overline{V_1} = V - V_1$. Meanwhile, we also define a partition of a network G, $G_1(V_1, E_1)$, $G_2(V_2, E_2)$, ... , $G_m(V_m, E_m)$, where V_i and E_i are the aggregation of vertices and edges of G_i for $i = 1, 2, \ldots, m$, and the modularity Q can be defined as follows:

$$Q = \sum_{i=1}^{m} \left[\frac{L(V_i, V_i)}{L(V, V)} - \left(\frac{L(V_i, V)}{L(V, V)} \right)^2 \right]. \quad (1)$$

According to the above function Q, we can see that the main idea of modularity comes from a comparison between real community partitions and ones allocated without any regard for the underlying structure. Then, all differences in the partitions of these two kinds of network structure are summed. However, Fortunato and Barthelemy [17] pointed out that modularity is not competent in

finding valid communities, and regarded that the size of a detected community relies on the size of the whole network. The real reason is that the modularity does not include any information on the number of nodes in a community, and the detected community is highly sensitive to the total number of edges in the network. Therefore, it is impossible to find a community partition smaller than a certain size.

To resolve the limit of modularity detection problems, Li et al. [18] put forward a quantitative function named modularity density for community identification. They demonstrated that modularity density is superior to the widely used modularity. The modularity density function D is defined as follows:

$$D = \sum_{i=1}^{m} \frac{L(V_i, V_i) - L(V_i, \overline{V_i})}{|V_i|}. \quad (2)$$

By summing the ratio between the difference of the internal and external degrees of the network G_i and the size of the subnetwork, this method provides a way of testing the reasonableness of a community partition. The larger the value of D, the more accurate the partition is. Therefore, the community detection problem can be converted into a problem of finding partitions of a network by maximizing its modularity density D. After a series of theoretical and experimental tests, Li et al. [18] also proved the equivalence of the modularity density function and the kernel k means, and proposed a more general modularity density measure as follows:

$$D_\lambda = \sum_{i=1}^{m} \frac{2\lambda L(V_i, V_i) - 2(1-\lambda)L(V_i, \overline{V_i})}{|V_i|}; 0 \leq \lambda \leq 1, \quad (3)$$

if $\lambda = 1$, D_λ is equivalent to the ratio association, if $\lambda = 0$, D_λ is equivalent to the ratio cut, and if $\lambda = 0.5$, D_λ is equivalent to the normalized modularity density D. In short, the general modularity density D_λ can be viewed as a combination of ratio association and ratio cut. In general, the optimization of ratio association always divides a network into small communities, while the optimization of ratio cut always divides a network into large communities. In other words, according to this general function, we can analyze the topological structure and reveal a more detailed and hierarchical organization of the network by tuning the parameter λ. Eventually, the resolution limit of modularity can be resolved. Moreover, the modularity density is independent of the size of the graph, which means that the balance between intraconnectivity and interconnectivity is constant regardless of the size of the community. A detailed discussion and proof can be found in Reference [18]. Furthermore, Conde-Cespedes [37] designed an approach to easily identify the resolution limit through the comparison of linear modularization criteria, and Campigotto [38] proposed a generic version of the Louvain method for community detection with a number of quality functions. All these methods can also demonstrate that the modularity density function has no resolution limit problem.

Furthermore, current researches [39–42] show that not all community detection methods can find community partitions effectively due to the existence of a so-called detectability threshold problem. In particular, it was recently illustrated [40,41] that there exists a phase transition in the detectability of communities such that, below the transition, no algorithm can detect the real community partitions better than chance. Chen [42] found a universal phase transition phenomenon on community detectability, while Ghasemian [43] studied the detectability thresholds for community structure in dynamic networks. In this paper, we test the proposed method in well-known real-world and synthetic networks, which are widely used to illustrate the effectivity of community detection methods, and the communities of these networks are well defined.

2.2. Normalized Mutual Information (NMI)

At present, a great many methods were proposed for community detection, but it is not clear which method is reliable. In other words, when community partitions are found by an algorithm, a reasonable evaluation criterion should be used to evaluate how accurately the detection algorithm performed. The normalized mutual information measure, proven by Danon et al. [44], is a reliable

criterion in evaluating community partitions. Therefore, we used normalized mutual information (NMI) in order to evaluate the similarity between the real partitions and the detected ones. Given two partitions A and B of a network in communities, let C be the confusion matrix whose element $C_{i,j}$ is the number of nodes of a community i of the partition A, which is also in the community j of the partition B. The normalized mutual information $I(A, B)$ is defined as follows:

$$I(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} C_{ij}\log\left(\frac{C_{ij}N}{C_{i.}C_{.j}}\right)}{\sum_{i=1}^{C_A} C_{i.}\log\left(\frac{C_{i.}}{N}\right) + \sum_{j=1}^{C_B} C_{.j}\log\left(\frac{C_{.j}}{N}\right)}, \tag{4}$$

where C_A (C_B) is the number of groups in the partition A (B), $C_{i.}$ ($C_{.j}$) is the sum of the elements of C in row i (column j), and N is the number of nodes. If $A = B$, $I(A, B) = 1$; if A and B are completely different, then $I(A, B) = 0$.

3. The Proposed CDDEA Algorithm

In this section, we explain the principle of the proposed CDDEA algorithm based on modularity density. Firstly, the differential evolution algorithm is introduced in community detection. Then, CDDEA is described in detail.

3.1. Community Detection Algorithm Based on Differential Evolution (DE)

The differential evolution (DE) algorithm, proposed by Strorn and Price [19], is widely utilized as an effective method in handling nondifferentiable, nonlinear, and multimodal objective functions. Moreover, the differential evolution algorithm is also a kind of evolutionary method frequently applied in most optimization problems, and shows similarity with the genetic algorithm (GA). For instance, basic stages of each method comprise mutation, crossover, and selection. However, DE shows some remarkable advantages compared to GA. Take the mutation step in DE for example, DE adds a scaled vector difference from two random individuals to a third individual according to particular rules, which plays an important role in improving search ability by making use of more individuals, and then avoids the disadvantages of GA in the mutation step.

In DE, there are three important operators: mutation, crossover, and selection. Firstly, a population is generated randomly. Secondly, a proper mutation operation should be selected to apply in initial population according to different problems. Thirdly, the crossover operation is employed to each pair of generated mutant individuals and its original individuals. Lastly, a selection operation is used to select a better individual for the next population. A brief presentation of DE operators and their applications in community detection are described in the subsections below.

3.1.1. Individual Representation and Initialization

For a network $G = (V, E)$, where V is the vertex set with a number n, and E is the edge set with a number e, an individual can be represented as follows:

$$inividual_j = \{c\_id_1, \cdots, c\_id_{i,,} \cdots c\_id_n\} i \in [1, n], c\_id_i \in [1, n]$$

where $inividual_j$ represents the jth individual in the population, and c_id_i means that the vertex i belongs to the community c_id_i. For instance, if $c\_id_5 = 3$, this means that the fifth vertex is now in the third community. Therefore, the vertex with the same community identifier (ID) should be considered in the same community. In the course of population initialization, each vertex is put in a different community for all individuals, such as $\{1, 2, \cdots, n\}$. In other words, each vertex belongs to a different community and there are n communities. However, if we do so, we do not utilize any information of the network in the process of initializing population, and lose sight of the fact that any two neighboring vertices may belong to the same community. Here, we employ a simple heuristic proposed in Reference [25]. For each individual, we randomly choose some nodes and assign their

community IDs to all their neighbors. This method is beneficial for improving the performance of the algorithm and avoids unnecessary iterations. For instance, Figure 1a shows a social network with nine actors and 14 connections [45].

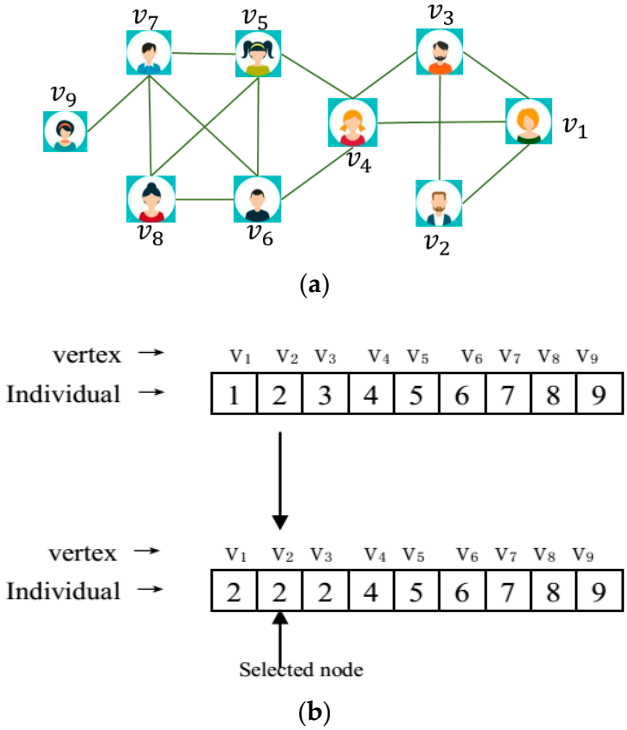

Figure 1. The social network and its representation of an individual with community ID. (**a**) (Color online) A social network of nine actors and 14 connections [45]. (**b**) Select a node randomly and assign its community identifier (ID) to its neighbor nodes.

According to Figure 1b, we can see clearly how to represent an individual in the form of a community ID. Meanwhile, we also can use network information [25] to initialize individuals in the course of the initial stage of CDDEA. For example, if vertex 2 is chosen, then we assign its community ID 2 to its neighboring nodes, vertex 1 and vertex 2; then, both their community IDs become 2.

3.1.2. Mutation

After an initialization of the population, DE makes a mutation operation and recombines the individuals to produce new individuals. Of course, there are different mutation schemes, and the most often used strategies are listed as follows [19]:

① DE/rand/1,

$$X^i(t+1) = X^i(t) + F \cdot (X^j(t) - X^k(t));$$

② DE/best/1,

$$X^i(t+1) = X^{best}(t) + F \cdot (X^j(t) - X^k(t));$$

③ DE/best/2,

$$X^i(t+1) = X^{best}(t) + F \cdot (X^j(t) + X^k(t) - X^m(t) - X^n(t));$$

④ DE/rand-to-best/1,

$$X^i(t+1) = X^i(t) + \omega \cdot (X^{best}(t) - X^j(t)) + F \cdot (X^m(t) - X^n(t)).$$

$X^i(t)$ is the current individual vector; $X^{best}(t)$ is the best individual in the current generation according to the fitness function; $X^j(t)$, $X^k(t)$, $X^m(t)$ and $X^n(t)$ are four different individuals randomly chosen from the current population; F is a scale factor, and its effective value is often smaller than 2.0 and bigger than 0; ω is also a scale factor named the greedy factor, which can accelerate the search speed for better individuals. In ①, the scaled difference between two randomly selected individuals, $F \cdot (X^j(t) - X^k(t))$, defines the direction and length of the search step; then, this difference is added to a thirdly randomly selected individual in order to create a mutation individual. ① is often named the classic mutation strategy in DE, while ②, ③, and ④ are variants of perturbations based on ① so as to find better individuals quickly. Therefore, in ②, the best individual in the current generation, $X^{best}(t)$, is chosen to replace the randomly selected individual in ①. On the other hand, ③ is very similar to ②; the main point of ③ is that it employs two difference vectors, $F \cdot (X^j(t) + X^k(t) - X^m(t) - X^n(t))$, for mutation. However, choosing the best individual, added with one or more difference vectors, often leads to only the best local individual. Hence, some important changes were made in ④ based on ①, ② and ③. In ④, the best individual and a random selected individual are chosen in the course of acquiring one difference vector with a scaled factor ω, and the other two random individuals are chosen in the process of getting another difference vector with a scaled factor F; then, the current individual is added with these two difference vectors in order to acquire a new mutation individual.

From the analysis above, we can see that ④ not only contains current individual information, but also utilizes information of the best individual and other three randomly selected individuals with corresponding scaled factors, which accelerates the search speed and avoids the disadvantage of being trapped in a local best individual. Therefore, ④ was naturally chosen for the CDDEA.

Here, we make a detailed specification concerning the process of mutation in the proposed method. Firstly, an individual with the highest modularity density value in the current population is selected. Then, the differential vector with a greedy factor ω is calculated between this individual and a random chosen one. Next, the other differential vector with a scaled factor F is acquired between two randomly selected individuals. Subsequently, the current individual is added with these two differential vectors. Last, but not the least, we must pay attention to the range of each vertex in each individual. For instance, the generated offspring after mutation is shown in Figure 2.

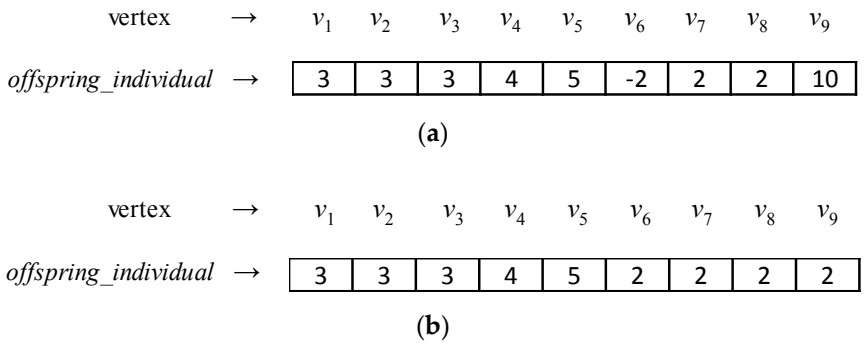

Figure 2. One of the offspring generated by mutation. (**a**) One of the offspring generated by mutation. (**b**) Adjusted community ID values.

According Figure 2, we can see that values of v_6 and v_9 are out of range of the community ID values. Here, we propose a method to handle this vertex using the information of its neighbor vertices, while the original DE algorithm uses a random number among valid values. If values of one or more vertices are beyond the range of community ID values, we substitute these values with their neighboring values, such as in Figure 2b. If there are multiple neighboring vertices, we can choose one randomly, and proper community ID values of each vertex are then generated in CDDEA.

3.1.3. Crossover

In order to enhance the variant diversity of individuals, a crossover operation is also employed in DE. The most common form of crossover is defined as follows [19]:

$$X_j^i(t+1) = \begin{cases} X_j^i(t+1), & rand_j^i \leq P_c \text{ or } j = J_{rand} \\ X_j^i(t), & rand_j^i > P_c \text{ or } j \neq J_{rand} \end{cases},$$

where $X_j^i(t)$ is the gene value in locus j of the individual i in current population, $X_j^i(t+1)$ is the new mutant individual, $i = 1, 2, \cdots, N$; j is the vector length of an individual, $j = 1, 2, \cdots, length(X^i(t))$; and the crossover probability, $P_c \in (0, 1]$, is a user-defined value that controls the fraction of parameter values copied from the mutant. If $rand_j^i \leq P_c$, the gene value of $X_j^i(t+1)$ is kept; otherwise, $X_j^i(t+1)$ is replaced with $X_j^i(t)$. In addition, the parameter J_{rand} is a random locus; if $j = J_{rand}$, the gene value in locus j should be inherited from the mutant individual of locus j, which ensures that the mutant individual at least provides a bit of useful information to the new offspring individual, thereby improving the diversity of population.

However, this kind of crossover is not fit for our proposed CDDEA algorithm, because we arbitrarily assign a random community ID to each vertex, and need to completely convey community ID information of a selected vertex and its neighbors to another individual. That is to say, the same community ID in different individuals may represent different communities, while the different community ID may also be the same community. For instance, in Figure 3, community ID 2 in $individual_1$ and $individual_2$ show different communities, while community ID 4 in $individual_1$ and community ID 2 in $individual_2$ represent the same community.

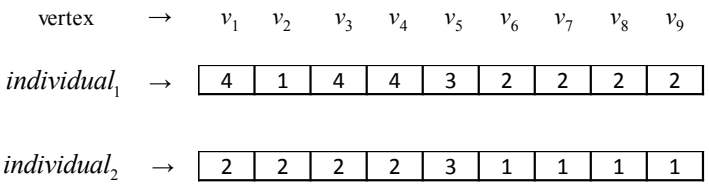

Figure 3. Two individuals with a random community IDs for each vertex.

In our algorithm, we employ a two-way crossover operation [2] inspired by a one-way crossover method [25]. For two individuals, $individual_a$ and $individual_b$, we randomly select a vertex v_i and find out other vertices with the same community ID in each individual. Then, we assign the community ID of v_i and other vertices with the same community ID to each individual and complete the process of crossover gradually. A case in point is given in Table 1, where new1 and new2 are two offspring after crossover between $individual_a$ and $individual_b$, and v_4 is the selected vertex ready to be crossed. Obviously, we can see that each individual conveyed its feature information of the selected vertex and others with the same feature to its offspring.

Table 1. Two-way crossover when v_4 is selected.

V	$Individual_a \rightarrow Individual_b$	New1	New2	$Individual_a \leftarrow Individual_b$	V
1	② → 3	②	③	2 ← ③	1
2	② → 1	②	2	2 1	2
3	1 1	1	1	1 1	3
④ →	② → 3	②	③	2 ← ③ ←	④
5	3 2	2	③	3 ← ③	5
6	4 2	2	③	4 ← ③	6
7	3 4	4	3	3 4	7
8	5 5	5	5	5 5	8
9	3 4	4	3	3 4	9

3.1.4. Selection

If the offspring individual $X^i(t+1)$ from the crossover has a bigger modularity density value than that of the original parent individual $X^i(t)$, it replaces $X^i(t)$ in the next generation; otherwise, $X^i(t)$ retains its place in the population at least one more generation. This operation is often named greedy selection. In other words, the selection of an individual is determined by its modularity density, and these individuals with high modularity density values have more chance of being chosen, passing their better community ID information to the next generation. Eventually, the modularity density makes the partition of the community evolve for the better.

3.2. The Framework of CDDEA

There is no doubt that the fitness function plays a key role in generating better individuals in evolutionary algorithms, and there is no exception in our algorithm. CDDEA employs modularity density as the fitness function for its advantages described in detail in Section 2.2.

The framework of CDDEA is described as follows:

Step 1: Initialize algorithm parameters; maximum number of generations = $G_{\max}$; population size = P_{size}; crossover probability = P_c; scale factor = F; greedy factor = ω.

Step 2: Initialize population according to Section 3.1.1; compute the modularity density of each individual in order to find the best individual in the current population.

Step 3: Perform the mutation scheme based on detailed instructions in Section 3.1.2.

Step 4: Execute the crossover operation between each mutant individual and each randomly selected individual from the current generation.

Step 5: Calculate the modularity density of each offspring from Step 4, and sort these offspring and the original individuals in the current population according to values of modularity density.

Step 6: Reconstruct the current individual population according to the sorted individuals from Step 5.

Step 7: Select P_{size} individuals from the newly constructed current population as parent individuals of the next generation.

Step 8: Check whether the termination criterion is satisfied: if a predefined number of iterations ($G_{\max}$) is achieved, stop and output the best individual generated in Step 5; otherwise, go to Step 3 for the next iteration.

Next, let us consider the time complexity of the CDDEA. Suppose V_n is the number of vertices in the network, $G_{\max}$ is the maximum number of generations, and P_{size} is the population size. Then, for the CDDEA, the main time-consuming components include the mutation and crossover operation. For P_{size} individuals, the complexity of the mutation operation is $O(P_{size}V_n)$. Furthermore, the complexity of the crossover operation is also $O(P_{size}V_n)$. Therefore, the whole time complexity of the CDDEA is $O(G_{\max}P_{size}V_n)$.

4. Experimental Results

In this section, we conduct experiments on both well-studied real networks and synthetic benchmark networks. We first compare the effectiveness of the proposed CDDEA on four real world networks used by Jia et al. (DECD) [46], Tasgin et al. (GA) [25], Pizzutiet al. (GA-Net) [47], Gong et al. (Meme-Net) [2], Blondel et al. [29], Newman et al. (Fast Newman) [14], and Danon et al. [48]. Then, we test the effectiveness of CDDEA and compare the results with community detection methods [2,25,46,47] based on the evolutionary algorithm in synthetic benchmark networks.

Our proposed algorithm CDDEA was implemented in MATLAB R2012b, and all experiments were conducted on Windows 7 with an Intel(R) Core (TM) i5-2520M processor, 2.5 GHz, 4 Gb RAM. The parameters in CDDEA were set as follows: number of population = 600, scaling factor $F = 1.0$, crossover parameter $P_c = 0.8$, and number of generations ranged from 50 to 250 according to the network being analyzed.

Since the results of community detection methods based on the evolutionary algorithm mainly rely on the effectiveness of the stochastic search process, we performed 10–30 iterations for each community detection algorithm using four real-world networks so as to objectively compare them.

4.1. Real-World Networks

We tested CDDEA using four real-world networks which received attention in many researches [2,12,16,18,23,25,26]. Meanwhile, we compared the effectiveness of community detection methods using normalized mutual information (NMI) to measure the similarity between the real community partitions and the detected ones (a detailed introduction is provided in Section 2.2).

Zachary's karate club: this network was generated by Zachary, who studied the friendship of 34 members of a karate club over a period of two years [49]. In the course of his research, he found a disagreement developed between the administrator and the instructor of the karate club. Eventually, the club divided into two groups of similar size. This network consists of 34 nodes and 78 edges.

Bottlenose dolphins network: this network is a description of 62 bottlenose dolphins living in the Doubtful Sound, New Zealand, compiled by Lusseau after studying their behavior for seven years [50]. A tie between two dolphins was established based on their statistically significant frequent association. The network splits naturally into two large groups, where the number of ties is 159.

American college football network: this network is from college football in the United States (US), which represents the schedule of Division I games in the course of the 2000 fall season [12]. Nodes in the graph represent teams, and edges represent the regular season games between the two teams they connect. The teams are separated into conferences. The teams, on average, played four inter-conference matches and seven intra-conference matches; thus, they play between members of the same conference more often. The network includes 115 nodes and 616 edges, grouped in 12 teams.

Political books about the US: this network involves books of US politics published around the time of 2004, complied by V. Krebs [51], which contains 105 nodes and 441 edges. These nodes represent 105 books on American politics bought from Amazon, while these edges show frequent co-purchasing of books by the same buyer. Books were divided separately by Newman [15] according to the descriptions and reviews of books posted on Amazon.

4.2. Synthetic Benchmark Networks

We also used the benchmark network proposed by Lancichinetti et al. [52] in order to evaluate the performance of CDDEA. The benchmark network is an extension of the classic benchmark network proposed by Girvan and Newman [12], which consists of 128 nodes separated into four communities of 32 nodes each. Each node has an average degree of 16, which shares a fraction $1-\mu$ of its links with other nodes within the same community, and a fraction μ with other nodes of the whole network. Here, μ is a mixing parameter playing an important role in controlling the network structure. When $\mu < 0.5$, the number of neighbors of a node inside its community is more than that of the neighbors belonging to other three communities; when $\mu \geq 0.5$, the number of neighbors of a node inside its community is equal or less than that of the neighbors belonging to other communities. In other words, a valid community detection method should be good at detecting these communities on the condition that $\mu < 0.5$. Therefore, we utilized this artificial network to check the performance of CDDEA. For this benchmark network, we ranged the mixing parameter μ from 0 to 0.5 in increments of 0.05, and generated 11 different networks. In general, when the value of μ increases, the detection of a community structure increases in difficulty.

4.3. Comparisons of Experimental Results

4.3.1. Experimental Results of Four Real-World Networks

Some related parameters in our experiments were set above; however, there was also an important parameter λ that needed to be set. As this tunable parameter plays a crucial role in finding better

community partitions in CDDEA, we needed to set a different parameter λ according to the studied network. After several tests on four real-world networks, we found a proper parameter λ for community detection methods in CDDEA and Meme-Net. In Table 2, we set $\lambda = 0.35$ in CDDEA and Meme-net, and the number of iterations was 50; in Table 3, we set $\lambda = 0.41$ in CDDEA, while $\lambda = 0.40$ in Meme-net, and the number of iterations was 100; in Table 4 , we set $\lambda = 0.81$ in CDDEA, while $\lambda = 0.80$ in Meme-net, and the number of iterations was 150; in Table 5, we set $\lambda = 0.41$ in CDDEA, while $\lambda = 0.40$ in Meme-net, and the number of iterations was 100. The greedy factor of CDDEA was equal to 1.8 for these four real-world networks.

Table 2. Normalized mutual information (NMI) result obtained using eight algorithms on Zachary's karate club network.

Algorithm	NMI			Standard Deviation
	Best	Worst	Average	
CDDEA	1.0000	1.0000	1.0000	0.0000
DECD	0.6956	0.6873	0.6889	0.0000
GA	0.6995	0.5866	0.6858	0.0192
GA-Net	0.7071	0.6280	0.6410	0.0181
Meme-Net	1.0000	1.0000	1.0000	0.0000
Fast unfolding	0.7071	0.5819	0.6194	0.0471
Fast Newman	0.6925	0.6925	0.6925	0.0000
Leon Danon	0.5485	0.5305	0.5377	0.0090

Table 3. NMI result obtained using eight algorithms on the bottlenose dolphins network.

Algorithm	NMI			Standard Deviation
	Best	Worst	Average	
CDDEA	1.0000	0.7563	0.9772	0.0698
DECD	0.5720	0.4039	0.4813	0.0518
GA	0.5901	0.5092	0.5578	0.0226
GA-Net	0.4703	0.3844	0.4186	0.0186
Meme-Net	1.0000	0.2260	0.9742	0.1413
Fast unfolding	0.5932	0.4618	0.5743	0.0333
Fast Newman	0.6208	0.5727	0.6031	0.0236
Leon Danon	0.5743	0.5743	0.5743	0.0000

Table 4. NMI result obtained using eight algorithms on the American college football network.

Algorithm	NMI			Standard Deviation
	Best	Worst	Average	
CDDEA	0.9269	0.8855	0.9129	0.0098
DECD	0.8301	0.7199	0.7912	0.0311
GA	0.8145	0.6418	0.7306	0.0410
GA-Net	0.8869	0.7106	0.8091	0.0079
Meme-Net	0.9242	0.8885	0.9129	0.0086
Fast unfolding	0.8903	0.8506	0.8872	0.0074
Fast Newman	0.7624	0.6977	0.7398	0.0243
Leon Danon	0.7725	0.7298	0.7517	0.0147

Table 5. NMI result obtained using eight algorithms on the American political books network.

Algorithm	NMI			Standard deviation
	Best	Worst	Average	
CDDEA	0.6085	0.4759	0.5706	0.0250
DECD	0.6025	0.4730	0.5385	0.0308
GA	0.5901	0.4834	0.5568	0.0253
GA-Net	0.4728	0.4085	0.4340	0.0151
Meme-Net	0.5901	0.5171	0.5832	0.0160
Fast unfolding	0.5812	0.2390	0.5326	0.0611
Fast Newman	0.5308	0.5225	0.5278	0.0041
Leon Danon	0.5737	0.5223	0.5377	0.0240

Tables 2–5 show the computational results using different detection methods on four real-world networks. The DECD and GA are modularity maximization methods based on evolutionary algorithms. The GA-Net finds community partitions by maximizing a fitness function named the community score. The Meme-Net is a community detection method which tries to optimize the network modularity density by employing a memetic algorithm. Our proposed algorithm is also a modularity density maximization method in the form of differential evolution. The other two algorithms, fast unfolding and Leon Danon, listed in Tables 2–5 for comparison, are extensions of the fast Newman algorithm proposed by Newman et al. [14].

From Tables 2–5, we can see that CDDEA shows a better performance in community detection on the four real-world networks. In Table 2, the results obtained by CDDEA show that it can converge to the global optimal NMI value with a value of 1. In other words, the detected communities by CDDEA are the same as the real ones. Of course, the Meme-net also presents better performance for Zachary's karate club network, and finds the optimal value of NMI. However, other algorithms do not detect the true partitions. Table 3 shows that the CDDEA method finds the optimal value of NMI with 0.9772 on the bottlenose dolphins network, while Meme-net obtains the best value of NMI with 0.9742. In the course of executing CDDEA and Meme-net, we recorded values of NMI for each method running 30 times. In CDDEA, we got the best value of $NMI = 1$ 27 times, $NMI = 0.8047$ once, and 0.7563 the remaining two times. In Meme-net, we obtained the optimal value of $NMI = 1$ 29 times, and $NMI = 0.2260$ once. According to this, we can see that methods based on the evolutionary algorithm plunge into local optimal values of NMI, and cannot escape easily. However, compared to other algorithms listed in Table 3, the higher average values of NMI obtained using CDDEA and Meme-net illustrate their better performance on the bottlenose dolphins network. In Table 4, most results obtained by the listed methods were very close to each other. We sorted their performance according to their average NMI values from large to small: CDDEA (0.9129), Meme-Net (0.9129), fast unfolding (0.8872), GA-Net (0.8091), DECD (0.7912), Leon Danon (0.7517), fast Newman (0.7398), and GA (0.7306). In Table 5, these methods could not find real partitions on the American political books network.

As far as the number of detected communities is concerned, the result will change according to the tunable parameter λ in D_λ. As can be seen in Table 6, for the proper λ set above, CDDEA can always find two real communities on Zachary's karate club network, and can detect two real communities on the bottlenose dolphins network under most conditions, while 12 or 13 communities were detected on the American college football network, and two to four communities were detected on the American political books network. Moreover, Table 6 shows that the parameter λ has an influence on the number of detected communities on these four real-world networks. For each network, we ran our algorithm 10 times, and recorded the average value, the maximum value, the standard deviation, and the number of detected communities. The range of parameter λ used in each real-world network was 0.25 to 0.90, with an interval of 0.05.

Table 6. Results of 10 runs of CDDEA on four real-world networks for different values of the parameter λ.

Network	NMI					Detected Communities
	λ	Best	Worst	Average	Standard Deviation	
Zachary's karate club network	0.25	0.0000	0.0000	0.0000	0.0000	1
	0.30	1.0000	0.2260	0.9226	0.2448	2
	0.35	1.0000	1.0000	1.0000	0.0000	2
	0.40	1.0000	0.6995	0.8197	0.1552	2, 3
	0.45	1.0000	0.6995	0.7295	0.0950	2, 3
	0.50	0.6995	0.6995	0.6995	0.0000	3
	0.55	0.7071	0.7071	0.7071	0.0000	4
	0.60	0.6873	0.6873	0.6873	0.0000	4
	0.65	0.6873	0.6873	0.6873	0.0000	4
	0.70	0.7071	0.6873	0.6893	0.0063	4
	0.75	0.7071	0.5451	0.6365	0.0542	5
	0.80	0.6280	0.5075	0.5609	0.0451	5, 6, 7
	0.85	0.5653	0.4480	0.4965	0.0313	7, 8
	0.90	0.5176	0.4359	0.4792	0.0290	9, 10
Bottlenose dolphins network	0.25	0.8888	0.0000	0.7395	0.2886	2
	0.30	1.0000	0.5797	0.9170	0.2060	2
	0.35	1.0000	0.5751	0.9351	0.1348	2
	0.40	1.0000	0.5115	0.9398	0.1546	2, 4
	0.45	1.0000	0.5797	0.8544	0.1710	2, 4
	0.50	0.8499	0.4567	0.6474	0.1253	3, 4, 5
	0.55	0.7185	0.4451	0.5568	0.0928	3, 4, 5
	0.60	0.7118	0.4325	0.5580	0.0940	4, 5, 6, 7
	0.65	0.5652	0.3565	0.4922	0.0578	6, 7, 9
	0.70	0.4874	0.3585	0.4341	0.0371	7, 8, 9
	0.75	0.4891	0.3573	0.4167	0.0416	8, 9, 10, 11, 12, 13, 14
	0.80	0.4313	0.3554	0.3856	0.0229	11, 12, 13, 14, 15
	0.85	0.3974	0.3447	0.3758	0.0172	13, 15, 16, 17, 18, 20
	0.90	0.3735	0.3139	0.3394	0.0195	17, 18, 19
American college football network	0.25	0.8013	0.0000	0.6208	0.2398	1, 3, 4, 5, 6, 7, 8
	0.30	0.8369	0.0000	0.6220	0.2562	1, 3, 4, 5, 6, 7, 8
	0.35	0.7981	0.0000	0.6176	0.2388	1, 4, 5, 7
	0.40	0.8113	0.4448	0.7239	0.1043	3, 6, 7, 8
	0.45	0.8418	0.7112	0.7763	0.0428	6, 7, 8, 9
	0.50	0.8726	0.7669	0.8252	0.0340	7, 8, 9, 10
	0.55	0.9361	0.7162	0.8298	0.0637	6, 7, 8, 10, 11
	0.60	0.9063	0.8386	0.8728	0.0218	9, 10, 11
	0.65	0.8940	0.8196	0.8673	0.0256	10, 11, 12
	0.70	0.9100	0.7992	0.8791	0.0348	9, 10, 11, 13
	0.75	0.9075	0.8025	0.8774	0.0297	10, 11, 12, 14
	0.80	0.9331	0.8668	0.8978	0.0182	12, 13
	0.85	0.9049	0.8014	0.8704	0.0339	13, 14, 15, 17
	0.90	0.9040	0.8395	0.8822	0.0209	13, 14, 15, 17, 18
American political books network	0.25	0.5979	0.0000	0.5120	0.1814	1, 2
	0.30	0.5979	0.5001	0.5731	0.0370	2, 3
	0.35	0.5979	0.0000	0.5246	0.1850	1, 2, 3
	0.40	0.5979	0.5357	0.5751	0.0197	2, 3, 5
	0.45	0.5744	0.5172	0.5589	0.0202	2, 3, 4
	0.50	0.5744	0.4674	0.5421	0.0294	3, 4
	0.55	0.5751	0.4625	0.5339	0.0318	3, 4, 5, 6, 7
	0.60	0.5312	0.4296	0.4915	0.0321	5, 6, 7, 8
	0.65	0.5235	0.4295	0.4672	0.0304	6, 7, 8, 9
	0.70	0.4764	0.4234	0.4437	0.0170	8, 9, 10
	0.75	0.4486	0.3946	0.4300	0.0142	8, 9, 10, 11
	0.80	0.4354	0.3841	0.4102	0.0155	10, 11, 12
	0.85	0.4119	0.3542	0.3887	0.0161	13, 14, 15, 16
	0.90	0.3923	0.3345	0.3614	0.0177	15, 16, 17, 18, 19, 21, 22

Figure 4 shows the box plots over 10 runs of CDDEA on four real-world networks, which records the distribution of NMI values with the change in values of λ. On each box, the median is represented as a red line, and the edges of the box are the upper quartile and lower quartile. Meanwhile, the whiskers extend to the most extreme data point which is no more than 1.5 times the length of the box away from the box, and the outliers are plotted with red crosses. It can be seen from Figure 4a that, when $\lambda = 0.35$, the statistic values of NMI on Zachary's karate club network are all equal to 1, and there were no outliers, which means that the detected communities were the same as the real ones. In Figure 4b, when $\lambda = 0.30$ and $\lambda = 0.35$, the distribution values of NMI on the bottlenose dolphins network

were very close to each other, and the main difference was the number of outliers between them. When $\lambda = 0.40$, a better distribution of values of NMI was acquired. Then, the values of NMI began decreasing with an increase in λ. Therefore, parameter λ should be set to 0.4 in order to find ideal community partitions. Similarly, in Figure 4c, when $\lambda = 0.80$, better community partitions concerning the American college football network were detected compared to other values of λ. In Figure 4d, we can also detect better communities on the American political books network if $\lambda = 0.40$.

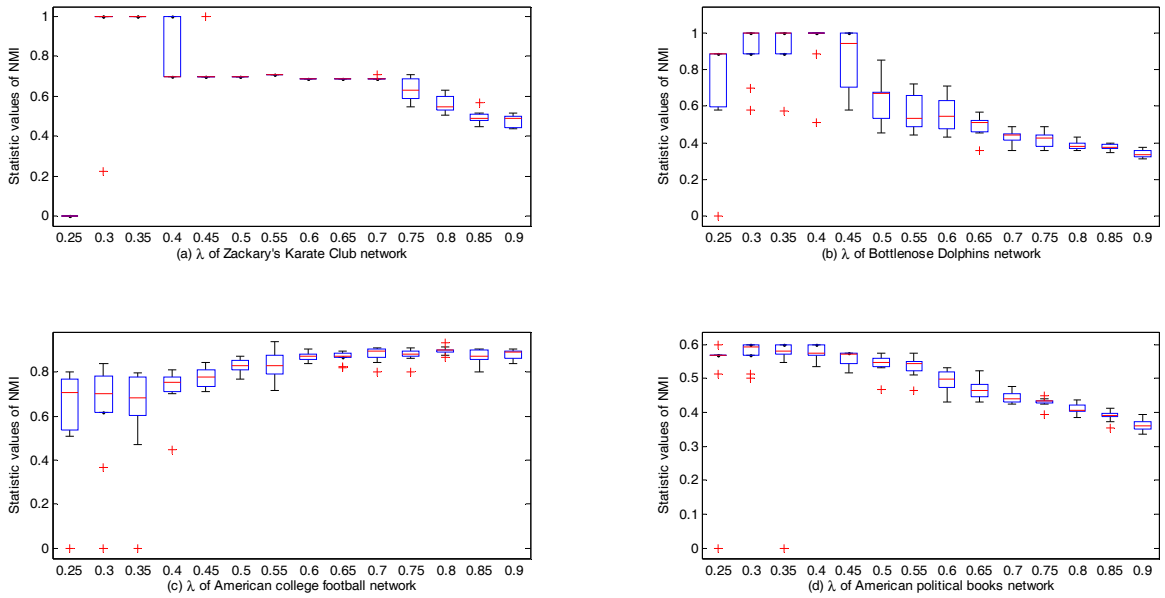

Figure 4. (Color online) Statistical values of normalized mutual information (NMI) over the 10 runs of the community detection approach based on a differential evolution algorithm (CDDEA) on four real-world networks.

Figure 5 illustrates that CDDEA can find different ideal community partitions with the change in parameter λ on Zachary's karate club network. When $\lambda = 0.35$, the CDDEA found two partitions, which were the same as the real ones. For $\lambda = 0.45$, three partitions were found. The yellow partition in Figure 5a was separated into two small partitions, such as Figure 5b, and node 10, originally belonging to the yellow partition, was allotted to the red partition. When $\lambda = 0.55$, this original red partition in Figure 5b was separated into two other small ones, and there were four detected communities on Zachary's karate club network. For $\lambda = 0.65$, we can see from Figure 5d that nodes 24 and 28 were removed out from the red partition and added to the pink partition found in Figure 5c. When $\lambda = 0.75$, nodes 24, 27, 28, and 30 constituted a new partition compared to Figure 5d, as seen in Figure 5e. From Figure 5a–e, we can see that the number of detected communities increased with λ increasing. However, not all detected communities were meaningful due to the fact that, when λ reached an idea value for a particular network, the NMI also approached the perfect value 1, and if λ continued increasing, the NMI decreased. As far as Zachary's Karate Club network is concerned, CDDEA could find meaningful communities when λ exceeded the ideal value of 0.35. Of course, when λ exceeded 0.75, CDDEA found worse partitions which consisted of only two or three nodes. Therefore, it is crucial to set a proper value of λ according to the different network being studied with CDDEA.

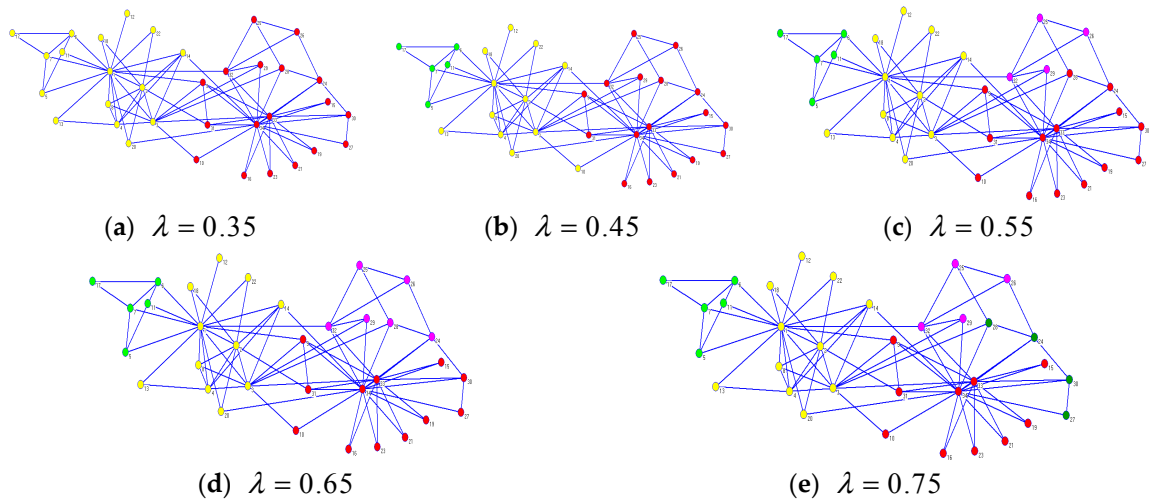

Figure 5. (Color online) Detected communities on Zachary's karate club network for different values of λ.

Concerning the dolphin social network, its real community partitions could also be detected using CDDEA in most conditions. For $\lambda = 0.41$, the corresponding average value of NMI was 0.9772, while the best NMI value of 1 occurred 27 times after running CDDEA 30 times, and the detected partitions are illustrated in Figure 6. Meanwhile, as can be seen from Table 6, the number of detected communities demonstrates a similar effect with the increase in λ.

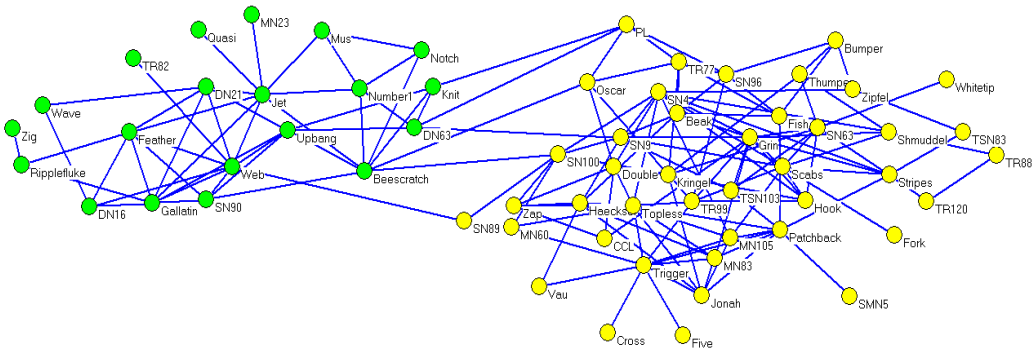

Figure 6. (Color online) Detected communities on the dolphin social network (when $\lambda = 0.41$, $NMI = 1$).

For the American college football network, we got a better result compared to the real partitions. When $\lambda = 0.81$, the average value of NMI value was 0.9129, which was very close to the real partitions. Figure 7 shows the detected communities generated by CDDEA. As can be seen from Figure 7, there were 13 partitions, which was the same as the number of real communities, while only a few nodes were misplaced.

On the American political books network, CDDEA did not acquire a better performance, and the best value of NMI was 0.6085 when $\lambda = 0.41$. Figure 8 shows the three detected partitions in one run of CDDEA when $\lambda = 0.41$. Furthermore, other community detection algorithms, such as fast Newman and Meme-Net, also could not find real partitions in this network due to its complexity. As can be seen from Table 5, the average value of NMI acquired by most algorithms was about 0.55, which means that the result of detected communities had some differences with the real ones.

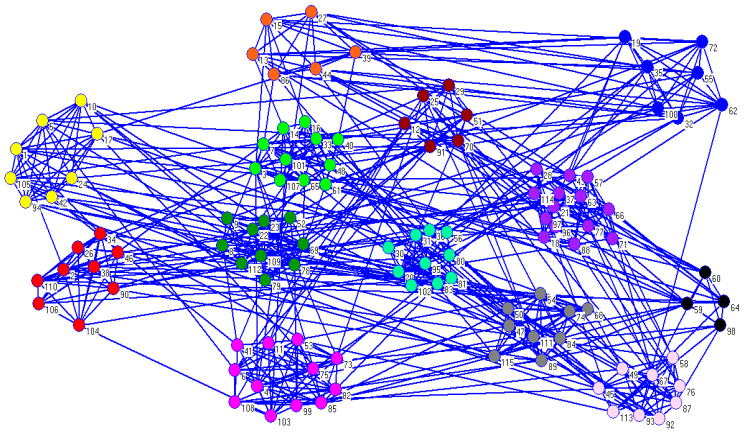

Figure 7. (Color online) Detected communities on the American college football network (when $\lambda = 0.81$,$NMI = 0.9126$).

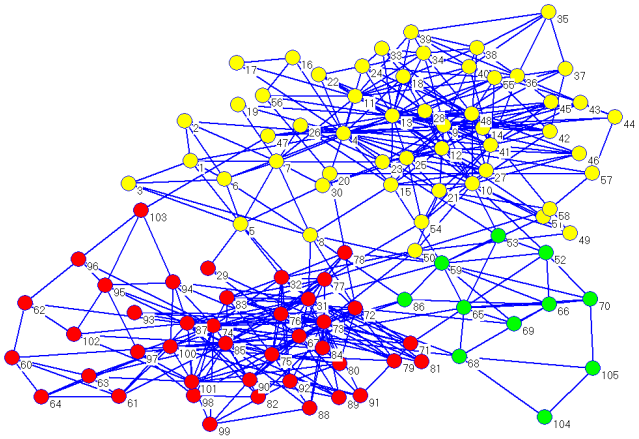

Figure 8. (Color online) Detected communities on the American political books network (when $\lambda = 0.41$, NMI = 0.5745).

4.3.2. Experimental Results of Synthetic Benchmark Networks

Firstly, we set the maximum generation of these five evolution-based algorithms to $G_{\max} = 250$; the mutation parameter is 0.2 in GA and Meme-Net, and some important parameters of CDDEA were set, as listed in Table 7, while other parameters were the same as the initial part of Section 4. There were 11 different networks, and with the increase in mix parameter μ range from 0 to 0.5, the complexity of the corresponding network increased, resulting in them being more difficult distinguish. Therefore, parameter λ needs to be tuned in the modularity density function in order to increase the ability of detection so as to find better community partitions. According to Reference [2], the default proper value for λ in Meme-Net is 0.5. However, this is not perfect with the increase in μ as NMI was equal to 0 when $\mu = 0.4$. In previous experiments on four real-world networks which have different complexities, we acquired better results by tuning the parameter λ. Accordingly, we can also tune λ in these 11 different networks. For each mix parameter μ which represents a different kind of network, we may set a corresponding λ to find better community partitions. After many tests on these benchmark networks, we found proper values for λ. In Meme-Net, we set $\lambda = 0.7$ when μ ranged from 0 to 0.5. In CDDEA, the proper values of λ were set as shown in Table 7, according to the complexity of corresponding network. Then, we ran each algorithm 10 times, and Figure 9 shows that CDDEA had a better performance in the benchmark network. When $\mu \leq 0.25$, CDDEA could basically find real community partitions except for the influence of plunging into local optimization values, which may also exist in other evolution-based algorithms. Moreover, the performance of CDDEA was

better than GA and DECD, and was close to Meme-net. When μ increased, the complexity of network also increased, and our algorithm could also find better community partitions and its performance was better than other methods. For instance, when $\mu = 0.30$, the average value of NMI was close to 1; when $\mu = 0.35$, it was close to 0.9, which shows that the detected communities were very close to the real ones. If μ continued increasing, the detection became a hard task for CDDEA, but the NMI was still close to 0.8, which was better than Meme-Net, GA, and DECD. When μ reached 0.45 or 0.5, none of these algorithms could detect the real communities due to the great complexity of the networks.

Table 7. Setting parameters in CDDEA for different networks.

Mix Parameter ¯	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
The greedy factor ω	1.80	1.80	1.80	1.80	1.80	1.85	1.85	1.85	1.85	1.85	1.85
The tuning parameter λ	0.800	0.800	0.800	0.800	0.800	0.800	0.843	0.870	0.875	0.915	0.915

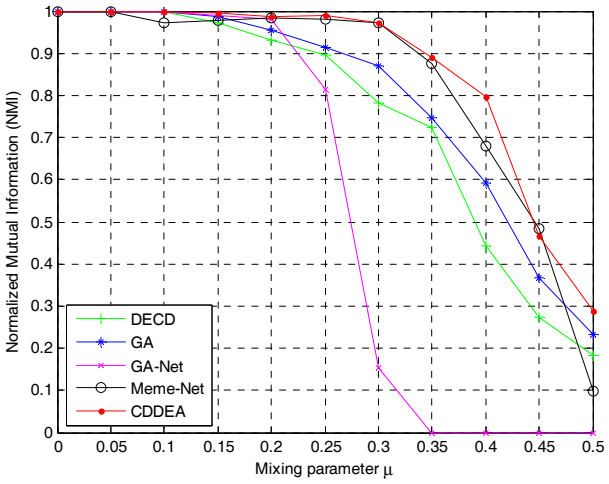

Figure 9. (Color online) Test of the accuracy of CDDEA on the synthetic benchmark network and a comparison with other evolution-based methods.

Meanwhile, we also found that proper parameters could effectively improve the detection ability of CDDEA in the course of finding better communities. By tuning λ, we could improve the resolution and detection ability of CDDEA in analyzing complicated networks; by tuning the greedy factor, we could improve the mutation ability and increase the diversity of individuals; by tuning the maximum number of iterations of CDDEA, we could acquire exact solutions which were close to real ones.

4.3.3. Scalability Analysis of CDDEA on Synthetic Benchmark Networks

In this section, we test the scalability of CDDEA on synthetic benchmark networks with the number of nodes varying from 128 to 10000. By tuning the parameters of the networks, different benchmark networks could be generated. These generated networks were defined as $LFR(N, k, \max k, mu, \min c, \max c)$, where N is the number of nodes in network, k is the average degree of nodes, max$k$ is the maximum degree of the nodes, mu is the mixing parameter, min$c$ is the minimum size of communities, and max$c$ is the maximum size of communities.

For different community detection methods, the evaluation of its efficiency is a big concern for users. Currently, many evaluation methods are proposed [14,18,37]. However, NMI and modularity are still two widely used methods. Therefore, we used these two criteria to test the scalability and efficiency of CDDEA. We generated six different LFR (Lancichinetti–Fortunato–Radicchi) benchmark networks according to Reference [52], and compared CDDEA to other evolutionary community

detection methods. Tables 8 and 9 separately show the NMI and Q value comparison of different algorithms on LFR Networks. From Table 8, we can see that when the number of nodes was small (128 or 256), almost all methods could find real community partitions. When the node size increased, the Meme-Net method could not detect communities in limited time due to its high time penalty in the local search operator. However, for CDDEA, its NMI value was around 0.8, and was close to the real community partitions. Although the NMI value of CDDEA was sometimes smaller than others (when the node size was 1000), CDDEA could get bigger Q values. These results demonstrate the efficiency and scalability of CDDEA.

Table 8. NMI value comparison of different algorithms on LFR Networks.

No	$LFR(N, k, \max k, mu, \min c, \max c)$	Nodes	Edges	NMI				
				CDDEA	DECD	GA	GA-Net	Meme-Net
1	(128,16,16,0.1,32,32)	128	1024	1.0000	1.0000	1.0000	1.0000	1.0000
2	(256,16,16,0.1,32,32)	256	2048	1.0000	1.0000	0.9889	0.9896	0.9625
3	(512,10,16,0.2,10,50)	512	2532	0.9270	0.7943	0.8477	0.8315	-
4	(1000,20,50,0.2,40,50)	1000	10067	0.8136	0.7161	0.7653	0.8351	-
5	(2000,20,50,0.2,50,60)	2000	20660	0.7848	0.6959	0.7150	0.7016	-
6	(10000,30,50,0.2,50,100)	10000	151655	0.8320	0.7044	0.7523	0.7223	-

Table 9. Q value comparison of different algorithms on LFR Networks.

No	$LFR(N, k, \max k, mu, \min c, \max c)$	Nodes	Edges	Q				
				CDDEA	DECD	GA	GA-Net	Meme-Net
1	(128,16,16,0.1,32,32)	128	1024	0.6504	0.6504	0.6504	0.6504	0.6504
2	(256,16,16,0.1,32,32)	256	2048	0.7744	0.7744	0.7125	0.7159	0.7076
3	(512,10,16,0.2,10,50)	512	2532	0.6593	0.5790	0.5608	0.5090	-
4	(1000,20,50,0.2,40,50)	1000	10067	0.6501	0.5162	0.5328	0.5447	-
5	(2000,20,50,0.2,50,60)	2000	20660	0.5937	0.4412	0.4812	0.5067	-
6	(10000,30,50,0.2,50,100)	10000	151655	0.6676	0.5648	0.6055	0.6117	-

5. Conclusions

In this paper, we put forward a community detection algorithm based on differential evolution using modularity density, and tested the effectiveness of CDDEA on four real-world networks, as well as artificial benchmark networks. Meanwhile, we compared it to other algorithms on these networks. The experimental results show that CDDEA is very effective in community detection problems, and modularity density is superior to traditional modularity in detecting better community partitions.

Author Contributions: C.L. conceived and designed the methodology, and the experiments. Q.L. performed the experiments and analyzed the data. C.L. wrote the paper. Both authors have read and approved the final manuscript.

Funding: This work was supported by the National Social Science Foundation of China (Grant No. 15BYY028), the Scientific Research Project of the Education Department of Liaoning Province, China (Grant No. 2016JYT01), and the Scientific Planning Project of the Education Department of Liaoning Province, China (Grant No. JG18DB109).

Acknowledgments: We thank Ruihua Qi for assistance in the revision of this manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Newman, M.E.J. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E* **2016**, *94*, 052315. [CrossRef] [PubMed]
2. Gong, M.; Fu, B.; Jiao, L.; Du, H. Memetic algorithm for community detection in networks. *Phys. Rev. E* **2011**, *84*, 056101. [CrossRef] [PubMed]
3. Gong, M.; Ma, L.; Zhang, Q.; Jiao, L. Community detection in networks by using multiobjective evolutionary algorithm with decomposition. *Phys. A Stat. Mech. Appl.* **2012**, *391*, 4050–4060. [CrossRef]

4. Atay, Y.; Koc, I.; Babaoglu, I.; Kodaz, H. Community detection from biological and social networks: A comparative analysis of metaheuristic algorithms. *Appl. Soft Comput.* **2017**, *50*, 194–211. [CrossRef]
5. Bu, Z.; Zhang, C.; Xia, Z.; Wang, J. A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network. *Knowl. Based Syst.* **2013**, *50*, 246–259. [CrossRef]
6. DasGupta, B.; Desai, D. On the complexity of Newman's community finding approach for biological and social networks. *J. Comput. Syst. Sci.* **2013**, *79*, 50–67. [CrossRef]
7. Chamberlain, B.P.; Levy-Kramer, J.; Humby, C.; Deisenroth, M.P. Real-time community detection in full social networks on a laptop. *PLoS ONE* **2018**, *13*, e0188702. [CrossRef] [PubMed]
8. Gavin, A.C.; Aloy, P.; Grandi, P. Proteome survey reveals modularity of the yeast cell machinery. *Nature* **2006**, *440*, 631–636. [CrossRef] [PubMed]
9. Eustace, J.; Wang, X.; Li, J. Approximating web communities using subspace decomposition. *Knowl. Based Syst.* **2014**, *70*, 118–127. [CrossRef]
10. Wang, K.; Ting, I.; Wu, H. Discovering interest groups for marketing in virtual communities: An integrated approach. *J. Bus. Res.* **2013**, *66*, 1360–1366. [CrossRef]
11. Newman, M.E.J. *Networks: An Introduction*; Oxford University Press, Inc.: Oxford, UK, 2010.
12. Girvan, M.; Newman, M.E.J. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef] [PubMed]
13. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113. [CrossRef] [PubMed]
14. Newman, M.E. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **2004**, *69*, 066133. [CrossRef] [PubMed]
15. Newman, M.E.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef] [PubMed]
16. Shang, R.; Bai, J.; Jiao, L.; Jin, C. Community detection based on modularity and an improved genetic algorithm. *Phys. A Stat. Mech. Appl.* **2013**, *392*, 1215–1231. [CrossRef]
17. Fortunato, S.; Barthelemy, M. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 36–41. [CrossRef] [PubMed]
18. Li, Z.; Zhang, S.; Wang, R.-S.; Zhang, X.-S.; Chen, L. Quantitative function for community detection. *Phys. Rev. E* **2008**, *77*, 036109. [CrossRef] [PubMed]
19. Storn, R.; Price, K. Differential Evolution—A simple and efficient heuristic for global optimization over continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
20. Neri, F.; Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [CrossRef]
21. Omidvar, M.N.; Li, X.; Mei, Y.; Yao, X. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Trans. Evolut. Comput.* **2014**, *18*, 378–393. [CrossRef]
22. Huang, T.W.Q.; Jia, G. Community detection using cooperative co-evolutionary differential evolution. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Taormina, Italy, 1–5 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 235–244.
23. Leal, T.P.; Goncalves, A.C.A.; Vieira, V.F.; Xavier, C.R. Decode-differential evolution algorithm for community detection. In Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Manchester, UK, 13–16 October 2013; pp. 4635–4640.
24. Liang, J.J.; Qu, B.Y.; Mao, X.B.; Niu, B.; Wang, D.Y. Differential evolution based on fitness Euclidean-distance ratio for multimodal optimization. *Neurocomputing* **2014**, *137*, 252–260. [CrossRef]
25. Tasgin, M.; Bingol, H. Community detection in complex networks using genetic algorithm. In Proceedings of the European Conference on Complex Systems, Oxford, UK, 25–29 September 2006.
26. Alam, M.N.; Das, B.; Pant, V. A comparative study of metaheuristic optimization approaches for directional overcurrent relays coordination. *Electr. Power Syst. Res.* **2015**, *128*, 39–52. [CrossRef]
27. Gong, M.; Cai, Q.; Li, Y.; Ma, J. An Improved Memetic Algorithm for Community. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, Australia, 10–15 June 2012; pp. 1–8.
28. Fortunato, S.; Hric, D. Community detection in networks: A user guide. *Phys. Rep.* **2016**, *659*, 1–44. [CrossRef]
29. Blondel, V.D.; Guillaume, J.-L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, P10008. [CrossRef]

30. Kernigan, R. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **1970**, *49*, 291–307. [CrossRef]
31. Pothen, A.; Simon, H.D.; Liou, K.-P. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* **1990**, *11*, 430–452. [CrossRef]
32. Nascimento, M.C.V. Community detection in networks via a spectral heuristic based on the clustering coefficient. *Discret. Appl. Math.* **2014**, *176*, 89–99. [CrossRef]
33. Corneil, D.G.; Gotlieb, C.C. An Efficient Algorithm for Graph Isomorphism. *J. ACM* **1970**, *17*, 51–64. [CrossRef]
34. Bellafiore, S.; Barneche, F.; Peltier, G.; Rochaix, J.-D. State transitions and light adaptation require chloroplast thylakoid protein kinase STN7. *Nature* **2005**, *433*, 892–895. [CrossRef] [PubMed]
35. Boettcher, S.; Percus, A.G. Optimization with extremal dynamics. *Phys. Rev. Lett.* **2001**, *86*, 5211–5214. [CrossRef] [PubMed]
36. Xu, G.; Tsoka, S.; Papageorgiou, L.G. Finding community structures in complex networks using mixed integer optimisation. *Eur. Phys. J. B* **2007**, *60*, 231–239. [CrossRef]
37. Conde-Céspedes, P.; Marcotorchino, J.F.; Viennet, E. Comparison of linear modularization criteria using the relational formalism, an approach to easily identify resolution limit. In *Advances in Knowledge Discovery and Management*; Springer: Cham, Switzerland, 2017; pp. 101–120.
38. Campigotto, R.; Céspedes, P.C.; Guillaume, J.L. A generalized and adaptive method for community detection. *arXiv*, 2014; arXiv:1406.2518.
39. Radicchi, F. A paradox in community detection. *Europhys. Lett.* **2014**, *106*, 38001. [CrossRef]
40. Radicchi, F. Detectability of communities in heterogeneous networks. *Phys. Rev. E* **2013**, *88*, 010801. [CrossRef] [PubMed]
41. Nadakuditi, R.R.; Newman, M.E.J. Graph spectra and the detectability of community structure in networks. *Phys. Rev. Lett.* **2012**, *108*, 188701. [CrossRef] [PubMed]
42. Chen, P.-Y.; Hero, A.O., III. Universal phase transition in community detectability under a stochastic block model. *Phys. Rev. E* **2015**, *91*, 032804. [CrossRef] [PubMed]
43. Ghasemian, A.; Zhang, P.; Clauset, A.; Moore, C.; Peel, L. Detectability thresholds and optimal algorithms for community structure in dynamic networks. *Phys. Rev. X* **2016**, *6*, 031005. [CrossRef]
44. Danon, A.D.-G.L.; Duch, J.; Arenas, A. Comparing community structure identification. *J. Stat. Mech. Theory Exp.* **2005**, *2005*, P09008. [CrossRef]
45. Tang, L.; Liu, H. *Community Detection and Mining in Social Media*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2010.
46. Jia, G.; Cai, Z.; Musolesi, M.; Wang, Y.; Tennant, D.A.; Weber, R.J.M.; Heath, J.K.; He, S. Community detection in social and biological networks using differential evolution. In *Learning and Intelligent Optimization*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 71–85.
47. Pizzuti, C. GA-Net: A genetic algorithm for community detection in social networks. In Proceedings of the 10th International Conference on Parallel Problem Solving from Nature: PPSN X, Dortmund, Germany, 13–17 September 2008; pp. 1081–1090.
48. Danon, L.; Díaz-Guilera, A.; Arenas, A. The effect of size heterogeneity on community identification in complex networks. *J. Stat. Mech. Theory Exp.* **2006**, *2006*, P11010. [CrossRef]
49. Zachary, W.W. An Information Flow Model for Conflict and Fission in Small Groups. *J. Anthropol. Res.* **1977**, *33*, 452–473. [CrossRef]
50. Lusseau, D.; Schneider, K.; Boisseau, O.J.; Haase, P.; Slooten, E.; Dawson, S.M. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behav. Ecol. Sociobiol.* **2003**, *54*, 396–405. [CrossRef]
51. Krebs, V. The Political Books about US Dataset. Available online: http://www.orgnet.com/ (accessed on 25 August 2018).
52. Lancichinetti, A.; Fortunato, S.; Radicchi, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **2008**, *78*, 046110. [CrossRef] [PubMed]