Memetic algorithm for community detection in networks

Maoguo Gong,[1] Bao Fu,[1] Licheng Jiao,[1] and Haifeng Du[2]

[1]*Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, Shaanxi Province 710071, China*

[2]*Center for Administration and Complexity Science, Xi'an Jiaotong University, Xi'an, Shaanxi Province 710049, China*

(Received 26 August 2011; revised manuscript received 18 September 2011; published 3 November 2011)

Community structure is one of the most important properties in networks, and community detection has received an enormous amount of attention in recent years. Modularity is by far the most used and best known quality function for measuring the quality of a partition of a network, and many community detection algorithms are developed to optimize it. However, there is a resolution limit problem in modularity optimization methods. In this study, a memetic algorithm, named Meme-Net, is proposed to optimize another quality function, modularity density, which includes a tunable parameter that allows one to explore the network at different resolutions. Our proposed algorithm is a synergy of a genetic algorithm with a hill-climbing strategy as the local search procedure. Experiments on computer-generated and real-world networks show the effectiveness and the multiresolution ability of the proposed method.

DOI: 10.1103/PhysRevE.84.056101 PACS number(s): 89.75.Hc, 89.75.Fb, 87.23.Ge

I. INTRODUCTION

Many real-world complex systems can be represented as networks. Collaboration networks, the World-Wide-Web, power grids, biological networks, and social networks are some examples. Networks could be modeled as graphs, where nodes (or vertices) represent the objects and edges represent the interactions among these objects. The area of complex networks has attracted many researchers from different fields such as physics, mathematics, biology, and sociology. Besides a number of distinctive properties such as the small world effect, the right-skewed degree distributions, and network transitivity that many networks seem to share, community structure is another important property in a complex network [1]. Qualitatively, a community is defined as a subset of the graph nodes which have a pattern of interconnections which is denser than that observed with the rest of the network nodes not in that community [2,3]. Community detection in complex networks is potentially very useful. Nodes belonging to the same community are more likely to have properties in common. For instance, in the World-Wide-Web, community analysis has uncovered thematic clusters [4,5].

Modularity, originally introduced by Newman and Girvan [3] to define a stopping criterion for their algorithm (the GN algorithm, which is a divisive hierarchical clustering algorithm, one of the most well-known community detection methods), has rapidly become an essential element of many community detection algorithm. By a modeling assumption, high values of modularity are associated with subjectively good partitions. As a consequence, the partition that has the maximum modularity is expected to be the subjectively best partition, or at least a very good one. This is the main motivation for modularity maximization, by far the most popular class of methods to detect communities in graphs [6]. Many algorithms have been employed to optimize modularity, including the greedy algorithm [7,8], simulated annealing [9], extremal optimization [10], etc. Genetic algorithms (GAs) have also been used to optimize modularity. In standard GAs, a population of strings (called chromosomes), which encode candidate solutions (called individuals) to an optimization problem, evolves toward better solutions. The evolution usually starts from a random set of individuals. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population based on their fitness, and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. After several generations, only solutions with large fitness survive. In a work by Tasgin *et al.* [11], modularity is the fitness function and partitions are the chromosomes. The algorithm does not require the number of communities present in a graph. The number of communities comes as an emergent result as the modularity value is optimized. In another work, by Liu *et al.* [12], the maximum modularity partition is obtained via successive bipartitions of the the graph, where each bipartition is determined by applying a GA to each subgraph (starting from the original graph itself), which is considered isolated from the rest of the graph. A bipartition is accepted only if it increases the total modularity of the graph.

However, Fortunato and Barthélemy [13] have found that modularity optimization may fail to identify modules smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the modules, even in cases where modules are unambiguously defined. This is the resolution limit of modularity optimization. Li *et al.* [14] have introduced a quality function called *modularity density*. The authors have demonstrated that this quantitative function is superior to the widely used modularity. Also, a general version of modularity density including a tunable parameter is proposed which allows one to explore the network at different resolutions.

Evolutionary algorithms that interspersed the recombination of high quality solutions with periods of intensive individual optimization were named memetic algorithms (MAs) in [15]. These methods are inspired by models of natural systems that combine the evolutionary adaptation of a population with individual learning within the lifetimes of its members. The denomination "memetic" for this type of algorithms was inspired by Dawkin's concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement [16]. In MAs, a meme is generally considered as an individual learning procedure capable of performing local refinements. In the literature, MAs have also been named hybrid genetic algorithms, genetic local searchers, Lamarckian genetic algorithms, etc. From an optimization point of view, MAs have been shown to be more efficient and more effective than traditional GAs for some problem domains. Over the past decade, MAs have been a hot topic in the fields of both computer science and operational research [17–25] (see [26] for more information about MAs). An exhaustive optimization of modularity density (or other quality functions) is impossible, due to the huge number of ways in which it is possible to partition a graph, even when the latter is small. MAs have proved to be of practical success in this kind of non-deterministic polynomial-time complete (NPC) problem. MAs have also been used for problems in physics. For instance, in [27], the authors developed a matching-based recombination algorithm to enhance the performance of an MA for solving the Min Number Partition problem, which is essentially equivalent to find the ground-state of an infinite-range Ising spin-glass system with antiferromagnetic couplings. In a study by Daolio *et al.* [28], the use of this type of method can in turn reveal the structure of low-lying minima in a disordered system.

It should be noted that two decades ago, Moscato and Fontanari [29] had already described the phenomenon that the stochasticity of the Metropolis updating in the simulated annealing algorithm does not play a major role in the search for near-optimal minima. This gives us the inspiration that for solving a combinatorial optimization problem such as community detection, the combination of some deterministic individual optimization methods with stochastic algorithms may produce better results. In a recent work by Rizzi [30], the authors proposed a new objective function, the *arithmetic-harmonic cut*, for graph bipartitioning. The objective function seeks to minimize the intracluster distances and at the same time it seeks to maximize the intercluster distances. The authors posed the hierarchical clustering problem as a finite number of instances of the arithmetic-harmonic cut problem. The authors also implemented a MA for the problem and demonstrated the effectiveness of the arithmetic-harmonic cut on a number of data sets. Hierarchical clustering is a popular class of methods for finding clusters in a set of data points. Actually, hierarchical clustering can be also used in community detection. However, it does not provide a way to discriminate between the many partitions obtained by the procedure, and to choose that or those that better represent the community structure of the graph [6]. The results of the method also depend on the specific distance (or similarity) measure adopted. In our method, only information about the vertices of the graph and their connections is needed.

In this paper, we propose a community detection algorithm which tries to optimize the network modularity density (the general version including a tunable parameter) by employing MAs. The proposed algorithm, named Meme-Net, combines GAs and a hill-climbing strategy as the local search procedure. Experiments on computer-generated and real-world networks show the effectiveness of our algorithm. The algorithm does not require the number of communities present in the graph in advance. In addition, by tuning the parameter in the quality function, we are able to explore the network at different resolutions and may reveal the hierarchical structure of the network.

II. CONCEPTION OF MODULARITY DENSITY

Before describing our algorithm in detail, in this section we shall follow the work by Li *et al.* [14] to give a brief introduction to the quantitative function modularity density.

We consider an undirected graph $G = (V,E)$ with $|V| = n$ vertices and $|E| = e$ edges. The adjacent matrix of the graph is A. If V_1 and V_2 are two disjoint subsets of V, 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 $L(V_1,\overline{V_1}) = \sum_{i\in V_1, j\in \overline{V_1}} A_{ij}$, where $\overline{V_1} = V - V_1$. Given a partition $\Omega = \{V_1, V_2, \ldots, V_m\}$ of the graph, where V_i is the vertex set of subgraph G_i for $i = 1,\ldots,m$, the modularity density (also called the D value) is then defined as

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

In this equation, each summand means the ratio between the difference of the internal and external degrees of the subgraph G_i and the size of the subgraph. The larger the value of D, the more accurate a partition is. So the community detection problem can be viewed as a problem of finding a partition of a network such that its modularity density D is maximized.

Li *et al.* also proved the equivalence of modularity density and kernel k means, and proposed a more general modularity density measure:

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

When $\lambda = 1$, D_λ is equivalent to the ratio association; when $\lambda = 0$, D_λ is equivalent to the ratio cut; when $\lambda = 0.5$, D_λ is equivalent to the modularity density D. So the general modularity density D_λ can be viewed as a combination of the ratio association and the ratio cut. Generally, optimization of the ratio association algorithm often divides a network into small communities, while optimization of the ratio cut often divides a network into large communities. This general modularity density D_λ, which is a convex combination of these two indexes, can avoid the resolution limits. In other words, by varying the λ value, we can use this general function to analyze the topological structure and uncover more detailed and hierarchical organization of the complex network [14].

III. PROPOSED MEMETIC ALGORITHM FOR COMMUNITY DETECTION

In this section, we will give a detailed description of the proposed algorithm Meme-Net. Our goal is to maximize the modularity density, so the objective function (also the fitness function) is defined in Eq. (2). The framework of Meme-Net is given as Algorithm 1.

Algorithm 1 Algorithm framework of Meme-Net

```
1: Input: Maximum number of generations: $G_{\max}$ ; Population size: $S_{\text{pop}}$ ; Size of mating pool: $S_{\text{pool}}$ ; Tournament size: $S_{\text{tour}}$ ; Crossover probability: $P_c$; Mutation probability: $P_m$.
2: $\mathbf{P} \leftarrow$GenerateInitialPopulation($S_{pop}$);
3: repeat
4:     $\mathbf{P}_{\text{parent}} \leftarrow$Selection($\mathbf{P}$,$S_{\text{pool}}$ ,$S_{\text{tour}}$ );
5:     $\mathbf{P}_{\text{child}} \leftarrow$GeneticOperation($\mathbf{P}_{\text{parent}}$ ,$P_c$,$P_m$);
6:     $\mathbf{P}_{\text{new}} \leftarrow$LocalSearch($\mathbf{P}_{\text{child}}$ );
7:     $\mathbf{P} \leftarrow$UpdatePopulation($\mathbf{P}$,$\mathbf{P}_{\text{new}}$ );
8: until TerminationCriterion($G_{\max}$ )
9: Output: Convert the fittest chromosome in $\mathbf{P}$ into a partition solution and output.
```

This framework requires some explanation. First of all, the GenerateInitialPopulation() procedure is responsible for creating the initial population. The Selection() function is used to select parental population for mating in GA. Here we use deterministic tournament selection other than widely used roulette wheel selection because the value of evaluation function (2) may be negative. The GeneticOperation() function is used to perform crossover and mutation operation. The UpdatePopulation() procedure is used to reconstruct the current population using the population **P** and $\mathbf{P}_{\text{new}}$. Here, the current population is constructed taking the best S_{pop} individuals from $\mathbf{P} \cup \mathbf{P}_{\text{new}}$. As for the TerminationCriterion() function, it can be defined as setting a limit of the total number of iterations, reaching a maximum number of iterations without improvement, etc.

In the following, we will give a more careful description of the initialization procedure, genetic operation, local search strategy, and the parameter setting.

A. Representation and initialization

A partition Ω of the network G is encoded as an integer string

$$\mathbf{x} = \{x^1\ x^2 \cdots x^n\}.$$

Here, n is the number of the vertices in the graph, and x^i is the integer cluster identifier of vertex v_i, which can be any integer number between 1 and n. The vertices having the same cluster identifier are considered in the same community. This representation does not need the number of clusters present in the graph, which actually is a result of our algorithm. A graph of n vertices can be partitioned into n clusters at most, and in this case, each cluster contains only one vertex, which can be denoted as $\{1\ 2 \cdots n\}$. However, it should be noted that there are many different representations corresponding to the same partition. For instance, given a graph of four vertices, {3 1 2 3} and {1 2 3 1} represent the same partition {{1,4},{2},{3}} of the graph.

The population initialization procedure is given as Algorithm 2. Initially each vertex is put in a different cluster for all chromosomes, that is, each chromosome in the population is $\{1\ 2 \cdots n\}$. However, this initial population has a lack of diversity and each solution is of low quality. In GAs it is common to initialize a higher quality population to speed up the convergence. Here we employ a simple heuristic as in [11]. For each chromosome, we randomly select a vertex and assign its cluster identifier to all of its neighbors. We repeat this operation $\alpha \cdot n$ times for each chromosome in the initial population where α is a parameter and $\alpha = 0.2$ is used for the experiments reported in this paper. This operation is very fast and results in local small communities, but the resulting clusterings are still far from being optimal.

Algorithm 2 Population initialization procedure

```
1: Input: Population size: $S_{\text{pop}}$ .
2: Generate a population $\mathbf{P}$, each chromosome $\mathbf{x}_k$ of which is set to $\{1\ 2 \cdots n\}$, where $k = 1, 2, \cdots , S_{\text{pop}}$ .
3: for each chromosome $\mathbf{x}_k$ do
4:     $t_{\text{counter}} \leftarrow 0$;
5:     repeat
6:         randomly select a vertex $v_i$;
7:         $x_k^j \leftarrow x_k^i$ whenever $(v_i, v_j \in E)$;
8:         $t_{\text{counter}} \leftarrow t_{\text{counter}} + 1$;
9:     until $t_{\text{counter}} = \alpha \cdot n$
10: end for
11: Output: Population $\mathbf{P}$.
```

B. Genetic operators

Crossover. Traditionally, a one-point crossover takes two chromosomes, randomly selects a single crossing over point, exchanges all the elements of the chromosomes after that selection point between the two chromosomes, and returns two new chromosomes. However, this straightforward crossing over operation is not suitable for our algorithm, because for each chromosome, the clusters to which the vertices are assigned are represented by arbitrary integers. In our algorithm, based on one-way crossing over introduced in [11], we employ a *two-way crossing over* operation. The crossing over procedure is defined as follows. The two selected chromosomes are called $\mathbf{x}_a$ and $\mathbf{x}_b$, respectively. We pick a vertex v_i at random, determine its cluster (i.e., x_a^i) in the chromosome $\mathbf{x}_a$ and make sure that all the vertices in this cluster of $\mathbf{x}_a$ are also assigned to the same cluster in the chromosome $\mathbf{x}_b$ (i.e., $x_b^k \leftarrow x_a^i$, $\forall\ k \in \{k \mid x_a^k = x_a^i\}$). Simultaneously, we also determine the cluster of the vertex v_i in $\mathbf{x}_b$, and make sure that all the vertices in this cluster of $\mathbf{x}_b$ are also assigned to the same cluster in $\mathbf{x}_a$ (i.e., $x_a^k \leftarrow x_b^i$, $\forall\ k \in \{k \mid x_b^k = x_b^i\}$). This procedure returns two new chromosomes $\mathbf{x}_c$ and $\mathbf{x}_d$. An example of two-way crossing over is given in Table I. This two-way crossing over operation can generate descendants carrying features common to the parents, which represents the exploitative side of the crossover operator; on the other hand, the crossing over operation is exploratory, which means it can generate descendants carrying combinations of features taken from the parents. For instance, as shown in one of the descendants, $\mathbf{x}_c$ in Table I, v_4 becomes in the same community with v_3. These properties make the two-way crossing over operation suitable for our algorithm.

Mutation. In this process, we randomly pick a chromosome to be mutated. Then we employ one point mutation on this chromosome: a vertex is picked randomly on the chromosome, then the cluster of the vertex is randomly changed to the cluster

TABLE I. Two-way crossing over when v_3 is selected.

v		$\mathbf{x}_a$		$\mathbf{x}_b$		$\mathbf{x}_c$	$\mathbf{x}_d$		$\mathbf{x}_a$		$\mathbf{x}_b$		v
1		⑤	→	2	→	⑤	5		5		2		1
2		3		6		6	⑥	←	3	←	⑥		2
③	→	⑤	→	6	→	⑤	⑥	←	5	←	⑥	←	③
4		7		5		5	7		7		5		4
5		2		6		6	⑥	←	2	←	⑥		5
6		⑤	→	3	→	⑤	5		5		3		6
7		3		2		2	3		3		2		7

of one of its neighbors. This operation is repeated n times on the chromosome. The specialized mutation operator that only considers the neighbors of the vertex can decrease useless exploration and reduce the search space.

C. Local search procedure

We first define the neighbors of a partition. Given a partition $\Omega = \{V_1, V_2, \ldots, V_m\}$ $(2 \leqslant m \leqslant n)$ of a graph G, where m is the number of clusters of this partition and n is the number of vertices in the graph, a single vertex, chose from a cluster V_i $(i \in 1, \ldots, m)$, is reassigned into another cluster V_j $(j \in 1, \ldots, m$ and $j \neq i)$. The new partition Ω_{nbr} after this reassignment is called a neighbor of the partition Ω. In particular, when $m = 1$, $\Omega = \{V_1\}$, a neighbor of Ω is defined as $\Omega_{\text{nbr}} = \{V_1 - v,\ \{v\}\}$, where v is a vertex from V_1, and $\{v\}$ is a cluster including the single vertex v (a single-vertex cluster).

Then we can figure out N_Ω, the number of all possible neighbors of the partition Ω. If $m = 1$, $N_\Omega = n$; if $2 \leqslant m \leqslant n$, $N_\Omega = n(m-1) - \frac{1}{2}s(s-1)$, where s is the number of single-vertex clusters in this partition. For example, when $m = n$, the value of s must be n, so $N_\Omega = n(n-1) - \frac{1}{2}n(n-1) = \frac{1}{2}n(n-1)$.

Algorithm 3 Local search procedure

```
1: Input: $\mathbf{P}_{child}$.
2: $\mathbf{n}_{\text{current}}$ ←FindBest($\mathbf{P}_{\text{child}}$);
3: $y_{\text{islocal}}$ ← FALSE;
4: repeat
5:     L ←FindNeighbors($\mathbf{n}_{\text{current}}$);
6:     $\mathbf{n}_{\text{next}}$ ←FindBest(L);
7:     if Eval($\mathbf{n}_{\text{next}}$) >Eval($\mathbf{n}_{\text{current}}$) then
8:         $\mathbf{n}_{\text{current}}$ ← $\mathbf{n}_{\text{next}}$ ;
9:     else
10:        $y_{\text{islocal}}$ ← TRUE;
11:    end if
12: until $y_{\text{islocal}}$ is TRUE
13: Output: $\mathbf{P}_{\text{child}}$ .
```

The local search procedure used in Meme-Net is a hill-climbing strategy, and the implementation is given as Algorithm 3. In computer science, hill-climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found. Here we apply this optimization technique to a partition of the network. In Algorithm 3, the FindBest() function is responsible for evaluating the fitness of each chromosome in the input population, and return the chromosome having maximum fitness, on which the local search procedure will be performed. The Eval() function is used to evaluate the fitness of a solution. The FindNeighbors() function is responsible for finding the neighbors of a partition, which can be done easily according to the definition of the neighbors of a partition.

It is vital to note that the design of MAs raises a number of important issues which must be addressed [19]: Where, and when, should local search be applied within the evolutionary cycle? Which individuals in the population should be improved by local search, and how should they be chosen? How much computational effort should be allocated to each local search? In our algorithm, the local search is applied on $\mathbf{P}_{\text{child}}$, which is the population after crossover and mutation. Not all of the chromosomes in this population are undergoing the local search procedure. We only find the fittest chromosome in $\mathbf{P}_{\text{child}}$ and perform local search on it, until no improvement can be made.

It should be noted that the hill-climbing algorithm is sensitive to the starting point. In our algorithm, the larger the number of clusters included in a partition, the more effort the local search on this partition needs. This is because a partition consisting of a large number of clusters usually has a large number of neighbors, as denoted in N_Ω. From this point of view, the initialization procedure introduced in Sec. III A also speeds up the local search procedure.

D. Parameters

The proposed algorithm Meme-Net is not parameter free. Some parameters such as population size, number of generations, and crossover probability should be set to some values by us in advance. Since our purpose in this study is to show that the proposed Meme-Net is effective for community detection, some of the parameters are set to the values we found by trial and error. A thorough analysis of the effect of different parameters is out of the scope of this paper. Some parameters we have mentioned and their values we used in the experiments are summarized in Table II.

IV. EXPERIMENTAL RESULTS

In this section, we will test Meme-Net on 11 computer-generated networks and four real-world networks for which the partitions in communities are known. The results obtained by the fast modularity algorithm [8] are also given for comparison.

TABLE II. Some parameters in the algorithm.

Parameter	Meaning	Value
$G_{\max}$	The number of iterations	50
S_{pop}	Population size	450
S_{pool}	Size of the mating pool	$\frac{S_{\text{pop}}}{2}$
S_{tour}	Tournament size	2
P_c	Crossover probability	0.8
P_m	Mutation probability	0.2

The experiments on these networks show the effectiveness and the multiresolution ability of our method on community detection.

First, we will introduce the similarity measure called normalized mutual information (NMI) as described in [31], which is used in Meme-Net for estimating the similarity between the true 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_{ij} is the number of nodes of community i of the partition A that are also in the community j of the partition B. The normalized mutual information $I(A,B)$ is defined as

$$I(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} C_{ij}\log(C_{ij}N/C_{i\cdot}C_{\cdot j})}{\sum_{i=1}^{c_A} C_{i\cdot}\log(C_{i\cdot}/N) + \sum_{j=1}^{c_B} C_{\cdot j}\log(C_{\cdot j}/N)},$$

where c_A (c_B) is the number of groups in the partition A (B), $C_{i\cdot}$ ($C_{\cdot j}$) is the sum of elements of C in row i (column j), and N is the number of nodes (note that, some denominations here are different from the ones in previous sections just for convenience). If $A = B$, then $I(A,B) = 1$; if A and B are completely different, then $I(A,B) = 0$.

A. Computer-generated network

The network we used here is the benchmark network proposed by Lancichinetti *et al.* [32], which is an extension of the classic benchmark network proposed by Girvan and Newman in [1]. The network consists of 128 nodes divided into four communities of 32 nodes each. Every node has an average degree of 16 and shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction μ with the other nodes of the network; μ is the mixing parameter. When $\mu < 0.5$ the neighbors of a node inside its group are more than the neighbors belonging to the other three groups, thus a good algorithm should discover them. We use this computer-generated network to test if Meme-Net effectively detects the community structure inside the network.

We generated 11 different networks for the value of mixing parameter μ ranging from 0 to 0.5 and used the NMI to measure the similarity between the true partitions and the detected ones. For each network, we computed the average NMI over ten independent runs. Figure 1 shows the average NMI for different values of λ (the parameter in the objective function D_λ), when the mixing parameter increases from 0 to 0.5. As shown in Fig. 1, for $\lambda = 0.5$, when the value of mixing parameter μ is small ($\mu \leqslant 0.25$), which means the fuzziness of the community in the network is low, our algorithm find the true partition correctly (NMI equals 1). When the mixing parameter increases, it is more difficult to detect the true partition, but the detected partition is also close to the true one (NMI is about 0.9 when $\mu = 0.35$). Only when $\mu \geqslant 0.4$ is no community structure detected. However, the higher value of λ could help discover smaller communities. For instance, for $\lambda = 0.7$, when the mixing parameter $\mu = 0.45$, Meme-Net is able to detect about 80% of the community structure information. On the other hand, lower values of λ could help discover larger communities. For example, for $\lambda = 0.3$, when $\mu \geqslant 0.3$, Meme-Net tends to consider the whole network as a large single community (NMI equals 0). Notice that when $\mu = 0.5$, each node has half of the links inside the community and the other half with the rest the network. This means that the community structure is very fuzzy, and any algorithm can hardly find the true partition of the network.

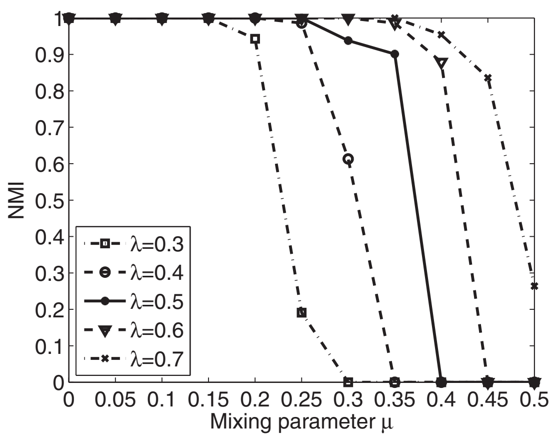


FIG. 1. Average NMI vs mixing parameter μ for different values of λ.

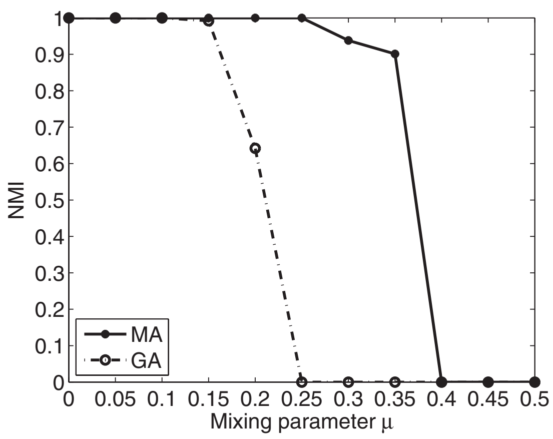
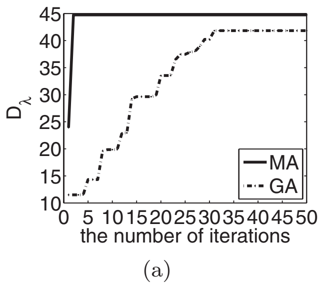


FIG. 2. Average NMI vs mixing parameter μ for $\lambda = 0.5$.

The local search procedure plays a very important role in Meme-Net. We simply developed a GA version of this algorithm by removing the LocalSearch function in Algorithm 1. Then we tested this GA version algorithm on the computer-generated networks with the same parameters as in Meme-Net. Figure 2 shows the average NMI obtained by Meme-Net and this GA version algorithm for $\lambda = 0.5$, when the mixing parameter increases from 0 to 0.5. It clearly shows that with the local search procedure, when $\mu \leqslant 0.25$, Meme-Net is able to detect the true partition of the network; however, without the local search procedure, it becomes harder for the algorithm to detect the true partition when $\mu > 0.15$.

The local search procedure also speeds up the convergence of Meme-Net. Figure 3 displays the values of D_λ and the corresponding NMI obtained by Meme-Net and the GA version algorithm in one run, for $\mu = 0.15$ and $\lambda = 0.5$, when the number of generations increases from 1 to 50. The figure shows that with the local search procedure, Meme-Net finds a maximum objective function value of 44.75 in just two generations, and the corresponding NMI is 1, which means the true partition is found. However, without the local search



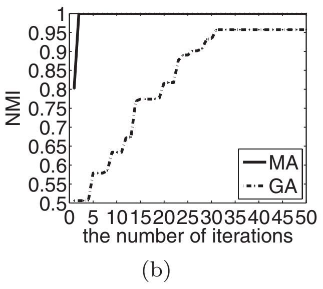


FIG. 3. (a) D_λ and (b) corresponding NMI vs the number of iterations for $\mu = 0.15$ and $\lambda = 0.5$.

procedure, after 50 generations, the algorithm does not find the true partition of the network.

B. Real-world networks

We now show the application of Meme-Net on four real-world networks: the Zachary's karate club, the Dolphin social network, the American college football, and the Books about US politics. The results obtained by the fast modularity algorithm [33] are also given at the end of the section for comparison.

Zachary's karate club. The karate club network was constructed by Zachary, who observed 34 members of a karate club over a period of 2 years [34]. During the course of the study, a disagreement developed between the administrator of the club and the club's instructor, which ultimately resulted in the instructor's leaving and starting a new club, taking about half of the original club's members with him.

TABLE III. Results of 30 runs of Meme-Net on four real-world networks for different values of λ.

Network	λ	I_{av}	I_{std}	I_{maxD}	$N_{cluster}$
Zachary's karate club	0.2	0	0	0	1
	0.3	1	0	1	2
	0.4	0.740	0.104	0.699	3
	0.5	0.699	0	0.699	3
	0.6	0.690	0.007	0.687	4
	0.7	0.688	0.004	0.687	4
	0.8	0.651	0.038	0.628	5
Dolphin social network	0.2	0.889	0	0.889	2
	0.3	1	0	1	2
	0.4	0.787	0.073	0.756	3
	0.5	0.569	0.035	0.586	5
	0.6	0.467	0.048	0.477	9
	0.7	0.400	0.034	0.385	11
	0.8	0.346	0.017	0.334	14
American college football	0.2	0.595	0.118	0.359	2
	0.3	0.723	0.078	0.523	3
	0.4	0.851	0.032	0.824	8
	0.5	0.890	0.033	0.911	11
	0.6	0.904	0.022	0.924	12
	0.7	0.912	0.011	0.911	13
	0.8	0.911	0.010	0.911	13
Books about US politics	0.2	0.583	0.015	0.598	2
	0.3	0.576	0.008	0.574	3
	0.4	0.588	0.011	0.590	4
	0.5	0.481	0.029	0.455	7
	0.6	0.449	0.018	0.434	9
	0.7	0.423	0.013	0.402	11
	0.8	0.394	0.014	0.378	14

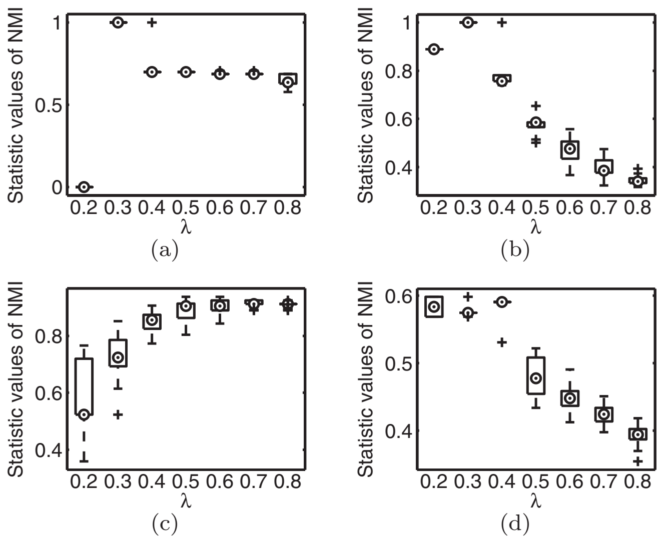


FIG. 4. Statistic values of NMI over the 30 runs on (a) Zachary's karate club, (b) Dolphin social network, (c) American college football, (d) Books about US politics for different values of λ. Here, box plots are used to illustrate the distribution of these samples. On each box, the central mark ⊙ is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points the algorithm considers to be not outliers, and the outliers are plotted individually. Symbol + denotes outliers.

Dolphin social network. The network of 62 bottlenose dolphins, living in Doubtful Sound, New Zealand, was compiled by Lusseau from the observation of dolphin behavior for 7 years [35]. A tie between two dolphins was established by their statistically significant frequent association. The network split naturally into two large groups, the number of ties being 159.

American college football. This network represents American football games between Division IA colleges during the regular fall season in 2000, as compiled by Girvan and Newman [1]. Nodes in the graph represent teams, and edges represent the regular season games between the two teams they connect. The teams are divided into conferences. The teams on average played four interconference matches and seven intraconference matches, thus teams tended to play between members of the same conference. The network consists of 115 nodes and 616 edges grouped into 12 teams.

Books about US politics. This is a network of 105 books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com. Edges between books represent frequent copurchasing of books by the same buyers [36]. Books were divided separately by Newman [37] based on a reading of the descriptions and reviews of the books posted on Amazon.

We set $\lambda = 0.5$ as the default value for λ in the objective function D_λ. For each network, we ran our algorithm 30 times, computed the average value and standard deviation of NMI (I_{av} and I_{std}) over the 30 runs, and recoded the value of NMI and the number of clusters corresponding to the maximum value of D_λ(I_{maxD} and $N_{cluster}$) in the 30 runs. Then we repeated this for $\lambda = 0.2$ to 0.8. The results are reported in Table III. We

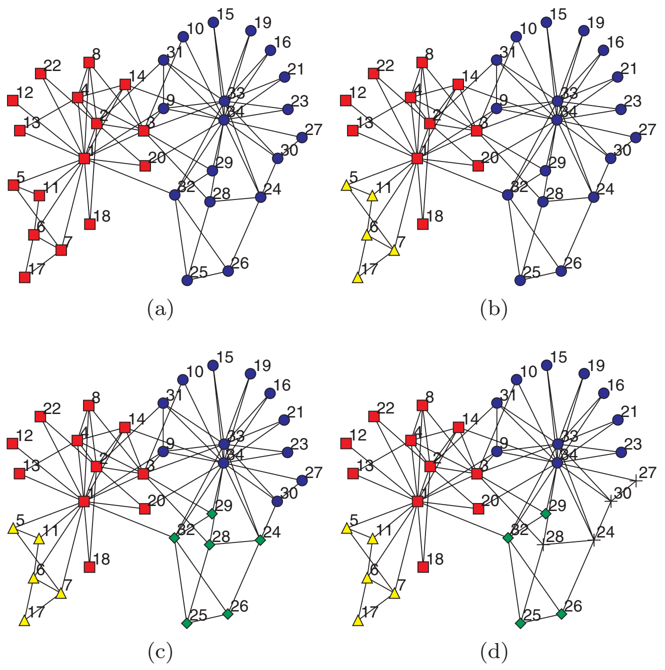


FIG. 5. (Color online) Detected partitions on Zachary's karate club network for (a) $\lambda = 0.3$, (b) $\lambda = 0.4$ or 0.5, (c) $\lambda = 0.6$ or 0.7, and (d) $\lambda = 0.8$.

also showed the statistic values of NMI over the 30 runs on the four real-world networks for different values of λ in terms of box plots in Fig. 4. As we can see from Fig. 4, on each of the four networks, the variability of NMI values obtained over the 30 runs is relatively small, especially for some values of λ. For instance, on Zachary's karate club network, this is true for almost all the λ values. In the following, we will give a more careful analysis about the experimental results.

On Zachary's karate club network, for $\lambda = 0.5$, in fact in all 30 runs, Meme-Net found a partition which consisted of three clusters, and the corresponding NMI was 0.699. Additionally, the algorithm converges very fast, which just needs a few iterations (usually less than 10 generations). Although this detected partition is different from the true one, it is very meaningful. In fact, this solution splits one of the two large groups into two smaller ones and never misplaces any node. Figure 5 displays the detected partitions corresponding to $I_{\max D}$ on this network for different values of λ. As we can see from the table, for $\lambda = 0.2$, the whole network is grouped into one cluster; for $\lambda = 0.3$, the network is grouped into two clusters [Fig. 5(a)], which is exactly the true partition, and the corresponding NMI is 1; for $\lambda = 0.4$ or 0.5, the network is grouped into three clusters, which splits the left part of the network into two smaller ones [Fig. 5(b)]; for $\lambda = 0.6$ or 0.7, four clusters are found, and this solution splits each of the two large groups into two smaller ones [Fig. 5(c)]; for $\lambda = 0.8$, the network is grouped to five clusters, which further splits the right part into three clusters [Fig. 5(d)]. If λ is set to 0.9 or larger, many small clusters containing only two or three vertices are detected. We did not display this network partition in Fig. 5.

On the Dolphin social network, for $\lambda = 0.3$, two clusters are found and the corresponding NMI is 1. This means that the detected partition is exactly the true one, which is shown in Fig. 6. For $\lambda = 0.4$, the network is grouped into three clusters and the corresponding NMI is 0.756. We find that this partition splits the lower right group of the network into two smaller ones and never misplaces a vertex. For $\lambda = 0.5$, five clusters are found and the corresponding NMI is 0.586, which splits the upper left group of the network into two smaller ones, and the lower right group into three smaller ones. For $\lambda = 0.6$ or larger, more smaller clusters are found. The experiments show that by tuning the parameter λ, we could explore the network at different resolutions. In general, the larger the λ value is, the smaller the communities Meme-Net tends to find.

Because of the complexity of the networks themselves, we did not find the "true" partition on the American college football network [Fig. 7(a)] and the Books about US politics network [Fig. 7(b)]. However, the detected ones are very close to the true partitions. For instance, on the American college football network, for $\lambda = 0.5$, 11 clusters are found and the corresponding NMI is 0.911, only a few vertices are misplaced. On the Books about US politics network, the results are also competitive with other popular community detection algorithms.

The results obtained by the fast modularity algorithm are given in Table IV. Now we consider the results obtained by Meme-Net for $\lambda = 0.5$. We can see that on Zachary's karate club network, the fast modularity algorithm found a solution with a NMI value of 0.693, while for all 30 runs Meme-Net found a solution with a NMI value of 0.699. On the American college football network, the average NMI value found by

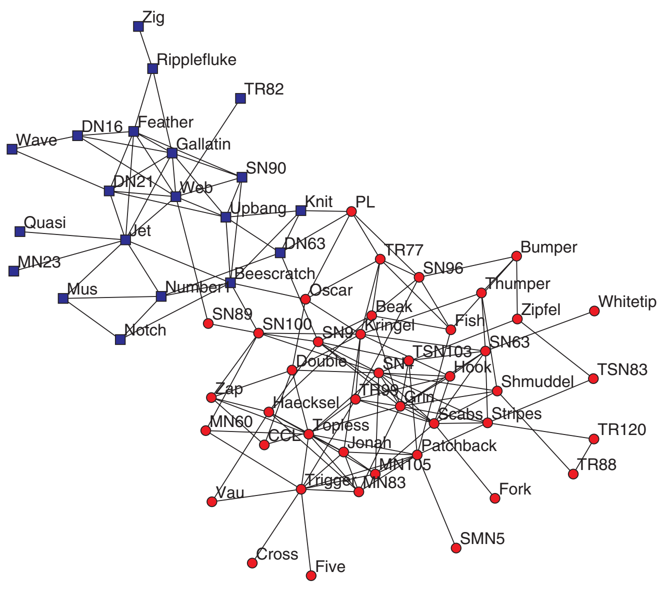


FIG. 6. (Color online) Dolphin social network.

TABLE IV. Results obtained by the fast modularity algorithm.

Network	Number of clusters	NMI
Zachary's karate club	3	0.693
Dolphin social network	4	0.573
American college football	7	0.762
Books about US politics	4	0.531

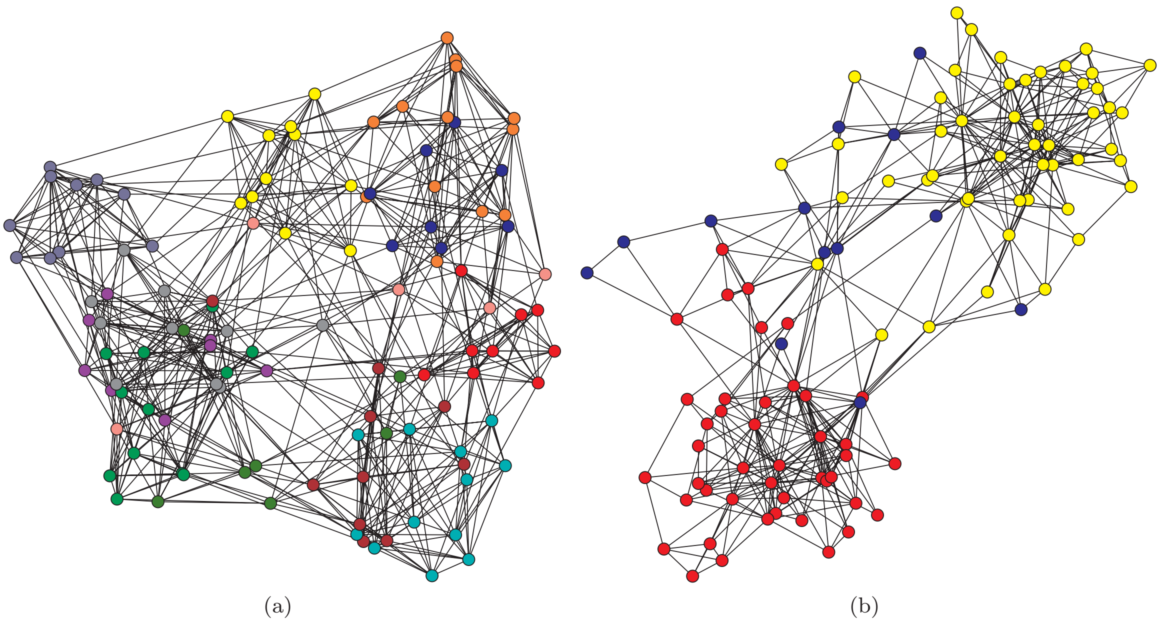


FIG. 7. (Color online) (a) American college football network and (b) Books about US politics network.

Meme-Net is 0.890, while the fast modularity algorithm found the NMI value of 0.762. On these two networks the results obtained by Meme-Net are better than the fast modularity algorithm. On the Dolphin social network and Books about US politics network, the average values of NMI found by Meme-Net are 0.569 and 0.481, respectively, while the fast modularity algorithm found the NMI values of 0.573 and 0.531, which are slightly better than Meme-Net. However, by tuning the parameter λ, we can also get better results. For example, on the Dolphin social network, when $\lambda = 0.3$, the solution found by Meme-Net is exactly the true partition. On the Books about US politics network, when $\lambda = 0.4$, the solutions with the average NMI value of 0.588 found by Meme-Net are more closer to the true partition. This comparison clearly shows the very good performance of Meme-Net with respect to the fast modularity algorithm.

In practice, we set $\lambda = 0.5$ as the default value for λ in the objective function D_λ, because this value indeed produces good results as we have seen in the experiments. Then if we want to analyze the network in a higher resolution, we can tune the value of λ larger. On the other hand if we want to analyze the network in a lower resolution, we can set the value of λ smaller.

V. CONCLUSION

In this paper, we propose the algorithm Meme-Net to optimize the modularity density for community detection. The proposed algorithm is a synergy of a genetic algorithm with a hill-climbing strategy as the local search procedure. Experiments show that combined with the local search procedure, Meme-Net performs better than traditional GAs on this subject and is competitive with state-of-the-art methods. In addition, experiments have shown that beyond the traditional quality function modularity, the modularity density is very suitable for community detection. By tuning the parameter λ in modularity density, we can analyze networks at different resolutions, and uncover topological structure and more detailed hierarchical organization of networks. Future work will aim at converting this single modularity density optimization problem into a multiobjective optimization problem, which can avoid tuning the parameter λ manually and automatically produce a set of meaningful candidate solutions in one run.

ACKNOWLEDGMENTS

The authors thank Dr. Andrea Lancichinetti for providing the benchmark testing networks and assistance with the visualization of networks. The authors also thank the editors and anonymous reviewers for their valuable comments and helpful suggestions which greatly improved the paper's quality. This work was supported by the Program for New Century Excellent Talents in University (Grant No. NCET-08-0811), the Program for New Scientific and Technological Star of Shaanxi Province (Grant No. 2010KJXX-03), and the Fundamental Research Funds for the Central Universities (Grant No. K50510020001).

[1] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).
[2] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. USA **101**, 2658 (2004).
[3] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[4] J.-P. Eckmann and E. Moses, Proc. Natl. Acad. Sci. USA **99**, 5825 (2002).

[5] G. Flake, S. Lawrence, C. Giles, and F. Coetzee, Computer **35**, 66 (2002).
[6] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[7] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[8] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[9] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, Phys. Rev. E **70**, 025101 (2004).
[10] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[11] M. Tasgin, A. Herdagdelen, and H. Bingol, e-print arXiv:0711.0491.
[12] X. Liu, D. Li, S. Wang, and Z. Tao, in *Computational Science-ICCS 2007*, Lecture Notes in Computer Science, edited by Y. Shi, G. van Albada, J. Dongarra, and P. Sloot, Vol. 4488 (Springer, Berlin, Heidelberg, 2007), pp. 657–664.
[13] S. Fortunato and M. Barthélemy, Proc. Natl. Acad. Sci. USA **104**, 36 (2007).
[14] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, and L. Chen, Phys. Rev. E **77**, 036109 (2008).
[15] P. Moscato C3P Report 826. California Institute of Technology (1989).
[16] R. Dawkins, *The Selfish Gene* (Oxford University, Oxford, 1989).
[17] H. Ishibuchi, T. Yoshida, and T. Murata, IEEE Trans. Evol. Comput. **7**, 204 (2003).
[18] Y. S. Ong and A. Keane, IEEE Trans. Electron. Comput. **8**, 99 (2004).
[19] N. Krasnogor and J. Smith, IEEE Trans. Electron. Comput. **9**, 474 (2005).
[20] M. Tang and X. Yao, IEEE Trans. Syst. Man Cybern. B Cybern. **37**, 62 (2007).
[21] Z. Zhu, Y.-S. Ong, and M. Dash, IEEE Trans. Syst. Man Cybern. B Cybern. **37**, 70 (2007).
[22] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, IEEE Trans. Evol. Comput. **13**, 604 (2009).
[23] K. Tan, S. Chiam, A. Mamun, and C. Goh, Eur. J. Oper. Res. **197**, 701 (2009).
[24] L. Jiao, M. Gong, S. Wang, B. Hou, Z. Zheng, and Q. Wu, IEEE Comput. Intell. Mag. **5**, 78 (2010).
[25] P. Moscato, A. Mendes, and R. Berretta, Biosystems **88**, 56 (2007).
[26] Y.-S. Ong, M. Lim, and X. Chen, IEEE Comput. Intell. Mag. **5**, 24 (2010).
[27] R. Berretta, C. Cotta, and P. Moscato, in *Metaheuristics: Computer-Decision Making* (Kluwer Academic, New York, 2003), pp. 65–90.
[28] F. Daolio, M. Tomassini, S. Vérel, and G. Ochoa, Phys. Stat. Mech. Appl. **390**, 1684 (2011).
[29] P. Moscato and J. Fontanari, Phys. Lett. A **146**, 204 (1990).
[30] R. Rizzi, P. Mahata, L. Mathieson, and P. Moscato, PLoS ONE **5**, e14067 (2010).
[31] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech. Theory Exp. (2005) P09008.
[32] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[33] [http://www.cs.unm.edu/∼aaron/research/fastmodularity.htm]
[34] W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[35] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behavior. Ecol. Sociobiol. **54**, 396 (2003).
[36] V. Krebs (unpublished), [http://www.orgnet.com].
[37] M. E. J. Newman, Proc. Natl. Acad. Sci. USA **103**, 8577 (2006).