Advances in Complex Systems, Vol. 13, No. 1 (2010) 3–17

DOI: 10.1142/S0219525910002463

World Scientific
www.worldscientific.com

A GENETIC ALGORITHM FOR DETECTING COMMUNITIES IN LARGE-SCALE COMPLEX NETWORKS

CHUAN SHI

Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China

BUPT-NOKIA Joint Research Laboratory
shichuan@bupt.edu.cn

ZHENYU YAN

Research Department, Fair Isaac Corporation (FICO), San Rafael, CA 94903, USA
yan_zhen_yu@hotmail.com

YI WANG, YANAN CAI and BIN WU

Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China

Received 24 May 2009
Revised 24 January 2010

Network model recently becomes a popular tool for studying complex systems. Detecting meaningful communities in complex networks, as an important task in network modeling and analysis, has attracted great interests in various research areas. This paper proposes a genetic algorithm with a special encoding schema for community detection in complex networks. The algorithm employs a metric, named modularity Q as the fitness function and applies a special locus-based adjacency encoding schema to represent the community partitions. The encoding schema enables the algorithm to determine the number of communities adaptively and automatically, which provides great flexibility to the detection process. In addition, the schema also significantly reduces the search space. Extensive experiments demonstrate the effectiveness of the proposed algorithm.

Keywords: Complex network; community detection; genetic algorithm; modularity.

1. Introduction

Recent researches (e.g. see Refs. 13, 22) indicate that complex systems in various domains may be modeled as complex networks. Examples of such systems and networks include the internet, WWW, social networks, citation networks, etc. Most of these networks are generally sparse in global yet dense in local. Scott [20] describes this characteristic as "the vertices within the groups have higher density of edges while vertices among groups have lower density of edges." Those "groups" are called

the communities, which are often the key elements to reveal many hidden features of a given network. Hence, community identification is a fundamental step not only for discovering what makes the entities come together, but also for understanding the overall structural and functional properties of large networks.

Because communities play crucial roles in complex networks, the identification of community structures has attracted a great deal of interest in physics and computer science societies. There have been many methods and algorithms proposed so far for revealing the underlying community structure in complex networks. Duch and Arenas [6] summarized that "these algorithms require the definition of community that imposes the limit up to which a group should be considered a community." A community detection algorithm's success in finding communities heavily depends on how it defines a community. A popular quantitative definition called network modularity Q, proposed by Girvan and Newman [9], is widely used as a quality metric for assessing the partitioning of a network into communities. Many recent algorithms employ the modularity as a quality metric, such as Newman's fast algorithm for very large networks [14] and the algorithm using extremal optimization [6]. The search for the largest modularity value is a NP-complete problem, because the space of all possible partitions grows faster than any power of the system size [2]. For this reason, many algorithms adopt various heuristic strategies to optimize this metric. However, these algorithms usually have high computational complexities, and are not suitable for large-scale complex networks. Thus, a highly efficient algorithm is desired.

This paper proposes a Genetic Algorithm for Community Detection (called GACD) to meet this need. The algorithm optimizes the modularity Q to obtain the community partition. A highly efficient locus-based adjacency encoding schema is applied to represent the community partition. The genetic representation not only reduces the search space distinctly, but also determines the number of communities adaptively and automatically. Based on the encoding schema, novel crossover and mutation operators are designed. Four experiments are carried out to test the performance of GACD. The first two experiments compare GACD with three popular methods through both artificial and real networks. Experimental results show that GACD can discover more accurate community structures for most problems. The third experiment tests the scalability of GACD. The results demonstrate that GACD can deal with large problems with smaller increase rates of time compared with other methods. The fourth experiment tests the stability of GACD through an e-mail network and a random network.

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 presents the details of the algorithm. The experimental results are illustrated in Sec. 4. Finally, we conclude the paper in Sec. 5.

2. Related Works

Many different approaches and algorithms have been proposed to analyze the community structures in complex networks. Those algorithms employ methods and

principles in physics, artificial intelligence, graph theory, and even electrical circuits. The spectral bisection methods [18] and the Kernighan–Lin [10] algorithm are early solutions to this problem. The spectral approach bisects graph iteratively, which is unsuitable to general networks. The Kernighan–Lin algorithm requires *a priori* knowledge of the size of the initial divisions, which is usually hard to obtain. In Social Network Analysis (SNA), a group of algorithms focus on the discovery of the so-called cohesive sub-structure [20], including the cliques [3], and quasi-cliques [15, 23], etc. These dense sub-structures often impose extra restrictions on the community definitions. Meanwhile, the average size of these sub-structures is always small, so people may get a great number of them, which actually hides the global organization of the network. Another widely used technique in SNA is the hierarchical clustering which groups similar vertices into larger communities.

One of the most known algorithms proposed as far, the Girvan–Newman (GN) algorithm, is a divisive method by iteratively cutting the edge with the greatest betweenness value [8, 9]. The algorithm can generate an optimized division of the network with $O(m^3)$ time complexity. Radicchi *et al.* [8] have proposed a similar methodology based on GN by using the edge-clustering coefficient as a new metric with a smaller time complexity $O(m^2)$. To further improve the efficiency, Clauset, Newman and Moore [4] have also proposed a fast clustering algorithm based on GN (GN Fast) with $O(n \log^2 n)$ time complexity on the sparse graph. GN Fast combines pairs of nodes to generate the maximum $\triangle Q$ iteratively until it becomes negative. Here m and n are the number of edges and nodes, respectively.

An important issue in community detection is how to quantitatively measure the quality of the community partitions. A quantitative definition, modularity, proposed by Grivan and Newman [8, 9] has been widely used in recent studies as the quality metric for the assessment of partitioning a network into communities. Modularity is defined as $Q = \sum_i (e_{ii} - a_i^2)$, where i is the index of the communities, e_{ii} is the fraction of edges, that connects two nodes inside the community i, to the total number of edges in the network and a_i is the faction of all the edges with at least one node in the community i to the total number of edges in the network. This measure essentially compares the number of links inside a given module with the expected values for a randomized graph of the same size and same degree sequences. Some other quantitative measures have also been proposed. The Hamiltonian-based method introduced by Reichardt and Bornholdt (RB) is based on considering the community indices of nodes as spins in a q-state Potts model [19]. Recently, Arenas, Fernandez, and Gomez (AFG) proposed a multiple resolution procedure that allows the optimization of modularity process to go deep into the structure [1]. The modularity Q is a special case of these two criteria. Once the modularity is chosen as the relevant quality function, the problem of community detection becomes equivalent to modularity optimization. The optimization problem is not trivial and it is indeed NP-complete [2]. Many heuristic search algorithms have been applied to solve the optimization problem. Duch and Arenas use the extremal optimization

method to optimize the modularity [5]. The simulated annealing method in Ref. 7 can obtain appropriate solutions, whereas it is very computationally expensive. Genetic Algorithm (GA), as an effective optimization technique, has also been used for community detection. The GA proposed by Tasgin and Bingol [21] (GATB) optimizes the modularity Q with the centroid-based genetic representation (i.e. a gene describes the index of the cluster a node belongs to). Due to the inefficient genetic representation, the algorithm is unsuitable for large-scale problems in fact. Pizzuti proposes another GA to optimize the "community score" criteria [16, 17]. Experiments illustrate that Pizzuti's algorithm is better than GN. However, the efficiency and effectiveness of the algorithm have not been validated on large-scale networks.

In all, most of the algorithms, although can successfully discover community structures in both artificial and social networks, have the high time and space complexities, which makes them unsuitable for large-scale networks. In addition, many algorithms also need the *a priori* knowledge about the community structure, such as the number of communities, etc. However, it is usually hard or even impossible to know these priori values in real networks. All these factors make it desirable to design a simple but highly effective algorithm that is suitable for large-scale networks.

3. The Genetic Algorithm for Community Detection

This section describes the GACD in detail, including the framework of the algorithm, the crucial genetic representation and the corresponding operators.

3.1. *Framework of the algorithm*

A GA is a search technique inspired by evolutionary biology to find exact or approximate solutions to optimization and search problems [25]. GAs are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals or phenotypes) to an optimization problem evolves toward better solutions. Based on the standard GA framework, the algorithm proposed in the paper is described in Algorithm 1.

To effectively apply GA to the community detection problem, there is much work to be done. A special genetic representation should be designed to encode community partitions, and the corresponding genetic variation operators need to be designed. These choices are nontrivial and are crucial for the performance and particularly for the scalability of the algorithm. Some encoding schemas may work well for networks with a few tens or hundreds of nodes, but their performance breaks down rapidly for larger scale networks. The design of an effective GA for community detection requires a close harmonization of the encoding and the operators, so that the search space can be reduced and the search can be guided effectively.

Algorithm 1. Main framework of GACD

procedure GACD($size, gens, p_c \in [0,1], p_m \in [0,1]$)
  //$size$ is the population size. $gens$ is the running generation.
  //p_c and p_m are the ratio of crossover and mutation, respectively, and $p_c + p_m = 1$.
  $P = \Phi$
  for each i in 1 to $size$ **do**
    $s_i = initialize(i)$
    $evaluate(s_i)$; $fillin(P, S_i)$
  end for
  for each g in 1 to $gens$ **do**
    $i = 0; P' = \Phi$
    while $i < size$ **do**
      randomly select two individuals s_i, s_{i+1} from P
      if random deviate $R(0,1) < p_c$ **then**
        $s'_i, s'_{i+1} = crossover(s_i, s_{i+1})$
      else $s'_i = mutate(s_i); s'_{i+1} = mutate(s_{i+1})$
      end if
      $i = i + 2$
      $evaluate(s'_i)$; $fillin(P', s'_i)$
      $evaluate(s'_{i+1})$; $fillin(P', s'_{i+1})$
    end while
    $update(P, P \bigcup P')$
  end for
  return $P[0]$
end procedure

$initialize(i)$ //initialize the individual i according to the genetic representation.

$evaluate(s)$ //evaluate the fitness of s according to modularity Q.

$fillin(P, s)$ //add individual s into population P, and sort P in the decreasing order of their fitness.

$update(P, P \bigcup P')$ //select first $size$ individuals with maximum fitness from $P \bigcup P'$ and fill in P in order.

$crossover(s_i, s_{i+1}), mutate(s_i)$ //crossover and mutation genetic operator, respectively.

3.2. *Genetic representation*

The biological and social complex networks are usually represented as graphs consisting of nodes and links, and then the communities to be detected are groups of nodes. When GA is applied to community detection, a community partition should usually be encoded in a character string (i.e. genotype) based on a genetic representation schema, and inversely the genotype (i.e. a solution of the problem

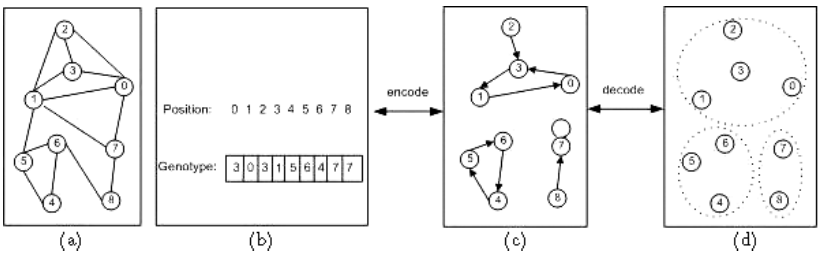

Fig. 1. Illustration of the locus-based adjacency representation. (a) shows the topology of a complex network. (b) shows one possible genotype. (c) shows how the genotype in (b) is translated into a graph structure, for example node 0 links to node 3, because the value of gene g_0 is 3. (d) shows the partition result.

or an individual in the population) can be decoded into a community partition. We design our genetic representation schema based on the locus-based adjacency representation [11] and illustrate it in Fig. 1. In this graph-based representation, each individual g consists of n genes $g_1, g_2, \ldots, g_n$ and each g_i can take one of the adjacent nodes of node i. Thus, a value of j assigned to the ith gene, is then interpreted as a link between nodes i and j; in the resulted partition solution, the two nodes will be in the same community. The decoding of this representation requires the identification of all connected components. All nodes belonging to the same connected component are then assigned to one community. Note that, using a simple backtracking schema, this decoding step can be done in linear time and the pseudo-code is illustrated in Algorithm 2.

The encoding approach cannot represent all possible partitions of nodes. For example, it cannot combine two disconnected subgraphs into one community. Someone may argue that the solutions with good community partitions may not fall in the solution space constructed by the genetic representation and encoding approach. Fortunately, Brandes *et al.* [2] have analyzed the basic structural properties of the clustering with maximum modularity and proposed the following theoretical results.

Property 1. There is always a clustering with maximum modularity, in which each cluster consists of a connected subgraph.

Property 2. A clustering of maximum modularity does not include disconnected clusters.

Because the genetic representation can accommodate all possibilities of connected subgraphs, these properties guarantee that a community partition with a favorable structure can be represented by the genetic representation. Moreover, the representation is well-suited for standard crossover operators, such as uniform, one-point or two-point crossover.

3.3. *Genetic operators*

The crossover operator based on the proposed genetic representation is illustrated in Fig. 2. For simplicity, the two chromosomes selected in crossover are named source

Algorithm 2. Decode of genotype

```
1: procedure DECODE
2:     current_cluster = 1
3:     for each i in 1 to N do
4:         cluster_assign = −1
5:     end for
6:     for each i in 1 to N do
7:         ctr = 1
8:         if cluster_assign_i = −1 then
9:             cluster_assign_i = current_cluster
10:            neighbor = g_i
11:            previous_ctr = i
12:            ctr = ctr + 1
13:            while cluster_assign_neighbor == −1 do
14:                previous_ctr =neighbor
15:                cluster_assign_neighbor = current_cluster
16:                neighbor = g_neighbor
17:                ctr = ctr + 1
18:            end while
19:            if cluster_assign_neighbor ≠ current_cluster then
20:                ctr = ctr − 1
21:                while ctr >= 1 do
22:                    cluster_assign_previous_ctr = cluster_assign_neighbor
23:                    ctr = ctr − 1
24:                end while
25:            else
26:                current_cluster=current_cluster + 1
27:            end if
28:        end if
29:    end for
30:    number_of_clusters=current_cluster
31: end procedure
```

and destination, respectively. First, we randomly select a gene from the source chromosome, and recursively search for a segment of genes (i.e. a community) that the selected gene links to (directly and indirectly). Then we substitute the counterpart genes in the destination chromosome with the segment of genes from the source chromosome. Counterpart here means that two genes are in the same position (or have the same index) in the source and destination chromosomes. Note that the crossover operation is bidirectional: we need to exchange the source and destination chromosomes and do the same action. An example of the crossover operation is shown in Fig. 2. The crossover operator has the following advantages: (1) it is prone to preserve the good structures generated in the evolution to the new individuals;

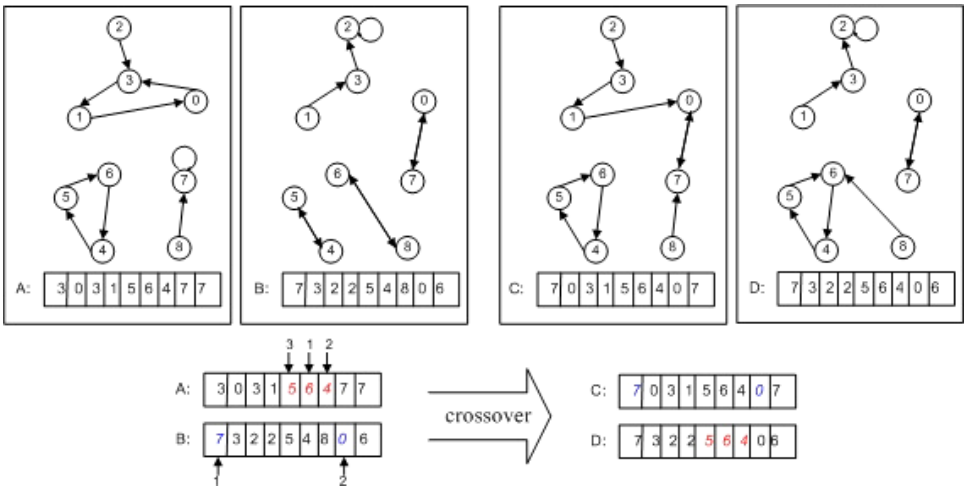

Fig. 2. Illustrate an example of the crossover operation. The position of genes selected from the source chromosomes are g_5 in A and g_0 in B. The selected gene segments are labeled with italic and different colors.

(2) it is efficient to generate individuals with diverse structures. The computational complexity is $O(l)$, where l is the length of the gene segment, namely the size of the selected community. l is usually smaller than n. In the mutation operation, we randomly select some genes and assign them with randomly selected adjacent nodes. The initial population is randomly generated. For each individual in the initial population, a gene g_i randomly takes one of the adjacent nodes of node i.

3.4. *Discussion*

When GA is applied to a real problem, one of the most important issues is to design an appropriate encoding schema according to the characteristics of the problem. A good encoding schema not only reduces the size of the search space, but also facilitates to design effective genetic operators. As a consequence, the encoding schema usually greatly influences the performance of GA. The locus-based adjacency encoding schema has several major advantages for community detection. Most importantly, there is no need to fix the number of communities in advance, as it is automatically determined during the evolution. Many methods require some *a priori* knowledge like the number of communities or threshold settled, whereas GACD does not need those setups beforehand. Another important advantage is that the search space constructed by the genetic representation is reduced significantly compared with other methods. In the existing GA (i.e. GATB), Tasgin and Bingol [21] use a number ranging from 1 to n to represent the community that a node belongs to, which results in the search space with the complexity n^n. Brandes *et al.* cast the problem of maximizing modularity into an integer linear program [2],

whose search space has a complexity 2^{n^2}. In GACD, because each node only links to its adjacent node, the complexity of the search space constructed by locus-based adjacency representation is $\prod_{i=1}^{n} d_i$ (n is the number of nodes, d_i is the degree of node i). For most real problems, d_i is much smaller than n. The reduced searching space enables GACD to find the more accurate solutions with less time.

The fitness evaluation function (i.e. the function that calculates the Q value) is the most time-consuming process in the algorithm. Calculating the objective value has the complexity $O(m)$, and the decoding process has the complexity $O(n)$. As a consequence, the fitness evaluation of an individual has the complexity $O(m + n)$. The whole complexity of GACD is $O(gs(m + n))$ (g is the running generation and s is the population size).

4. Experiments

In order to test the performance of GACD, this paper performs four sets of experiments. The experiments are carried out on a 3.4GHz and 2G RAM Pentium computer running Linux.

4.1. *Artificial network experiment*

To test the performance of the algorithm we first use the artificial networks with known community structures designed in Ref. 8. These networks have 128 vertices grouped in four communities of 32 vertices. Each vertex has on average z_{in} edges to vertices in the same community and z_{out} edges to vertices in other communities, keeping an average degree $z_{\text{in}} + z_{\text{out}} = 16$. The experiments generate 11 networks with z_{out} ranging from 0 to 10. As z_{out} increases, the community structure becomes less obvious, and thus they are more difficult to be identified. Three other existing algorithms are also tested: GN, GN Fast, and GATB (their descriptions can be seen in Sec. 2). The same parameters are used in GACD and GATB: *size* is 100, *gens* is 100, p_c is 0.8, and p_m is 0.2. The evaluation of partition results is not straightforward due to the difficulties in mapping clusters into communities and handling the mixed clusters. Some effective criteria have been proposed, such as variation of information [12] which measures the amount of information lost and gained between two compared partitions. Besides the modularity Q, the experiments use the Fraction of Vertices Identified Correctly (FVIC) as a measure criterion. FVIC has been used in many researches [5–7]. The larger the FVIC is the better the partition is.

Figure 3 shows the experimental results. We can observe that GACD always discovers the community structure with the highest FVIC, especially when z_{out} is larger than 6. Correspondingly, GACD also obtains the maximum Q values in most cases. GATB obtains the worst performances. GN seems to be more effective when the networks have the obvious community structures (i.e. z_{out} is smaller). It should be noted that the Confidence Intervals (CIs) are all very tight, which reveals that the solution distributions of all algorithms have very small variances and the

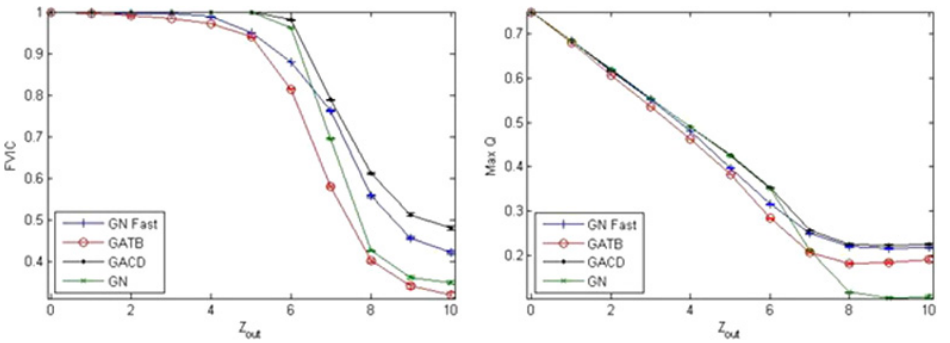

Fig. 3. Left: Fractions of Vertices Identified Correctly (FVICs) in the artificial networks. Both the means and 95% confidence intervals are calculated based on the results of 100 different networks. Right: means and 95% confidence intervals of maximum Q in the 100 different networks.

algorithms are all stable in this example. In addition, the tight CIs also indicate the statistical significance of the results.

4.2. *Real network experiment*

To further test the performance of GACD, the experiments compare the four algorithms on seven real social networks. As shown in Table 1, these popular networks [4, 8, 9, 21] have different scales with the number of vertices ranging from 34 to 22963. GATB and GACD have the same parameters: p_c is 0.8 and p_m is 0.2. According to the scale of the problems, the population sizes are 50, 50, 50, 50, 500, 500, 500, respectively; and the running generations are 20, 70, 100, 100, 500, 500, 2000, respectively. The experimental results are the average values of 30 runs. Figure 4 shows the means and the 95% CIs of the Q values of the four algorithms for the seven problems. It is obvious that GACD achieves the maximum Q for most problems and GATB generates the worst solutions. Although GACD is less stable for the problems with large size, it is stable for the middle-scaled problems. Similar to the artificial networks, the tight CIs indicate the statistical significance of the

Table 1. A comparison of the results of four different algorithms.

Problem	Vertice number	Edge number	GN		GN fast		GATB		GACD	
			N	T	N	T	N	T	N	T
Karate (P1)	34	78	5	1	3	1	5	1	4	1
Football (P2)	115	616	10	2	7	1	10	1	11	1
Enron (P3)	120	576	7	1	5	1	15	1	5	1
Celegansneural (P4)	297	1179	33	201	4	1	79	1	6	1
Tomcat (P5)	1607	6235	27	13233	28	4	234	657	44	86
Internet (P6)	8712	23305	—	—	29	34	3631	6937	151	552
as-22july06 (P7)	22963	48436	—	—	39	114	—	—	193	2139

Note: N represents the number of communities; and T represents the running time (the unit is second). "–" indicates that no results can be obtained in 10 h.

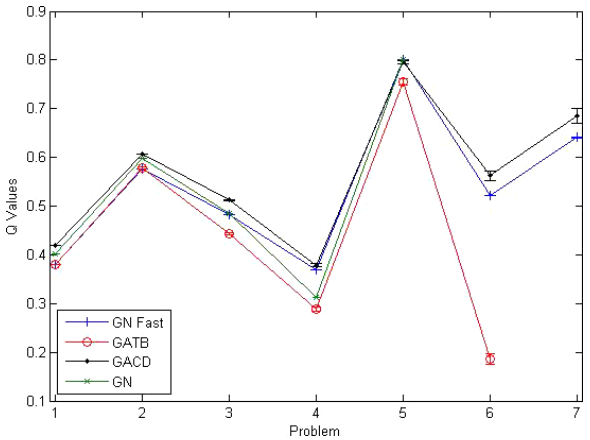

Fig. 4. Means and 95% of confidence intervals of modularity Q values for seven real networks. GN has no results in P6 and P7 and GATB has no result in P7.

results. As shown in Table 1, considering the running time, GN Fast is the fastest, and GACD is faster than the other two algorithms. It can also be found that GACD finds more communities than GN Fast, which indicates GACD can reveal communities with small size. For the large-scale problems (e.g. P6, P7), GN is not able to obtain a result in the reasonable time, because the algorithm needs huge memory. GATB has the same problem. Although GATB and GACD both are based on GA, their different genetic representations result in the different performances.

4.3. *Scalability experiment*

In this experiment, we test the scalability of GACD with network $K-m, n$ [7]. The network $K-m, n$ is made of m identical complete graphs with n nodes, and a link connects two adjacent complete graphs. It is clear that there are m communities in the network. In the experiments, n is 20 and m ranges from 50 to 250 with an interval of 20. Both GACD and GATB are tested in the experiments. The same parameters are settled for these algorithms: the population size is fixed at 200, p_c and p_m are 0.8 and 0.2, respectively. The stopping criterion of both GACD and GATB is the convergence of algorithms (i.e. the best and the worst individuals have the same fitness).

The result shown in Fig. 5 is based on the average of 30 runs. Figure 5(a) shows that the running time and generations of GACD increase stably as the scale increases, which indicates that, for the large-scale problems, GACD requires more generations and running time. Although the running generations and time of GATB increase for small-scale problems, GATB converges prematurely for large-scale problems. Figure 5(b) clearly demonstrates that the running time per generation of GACD (fixed the population size) increases linearly with the scale of problem, whereas that of GATB increases much faster. Figures 5(c) and (d) illustrate the quality of solutions obtained by GACD and GATB. With the increase of the scale,

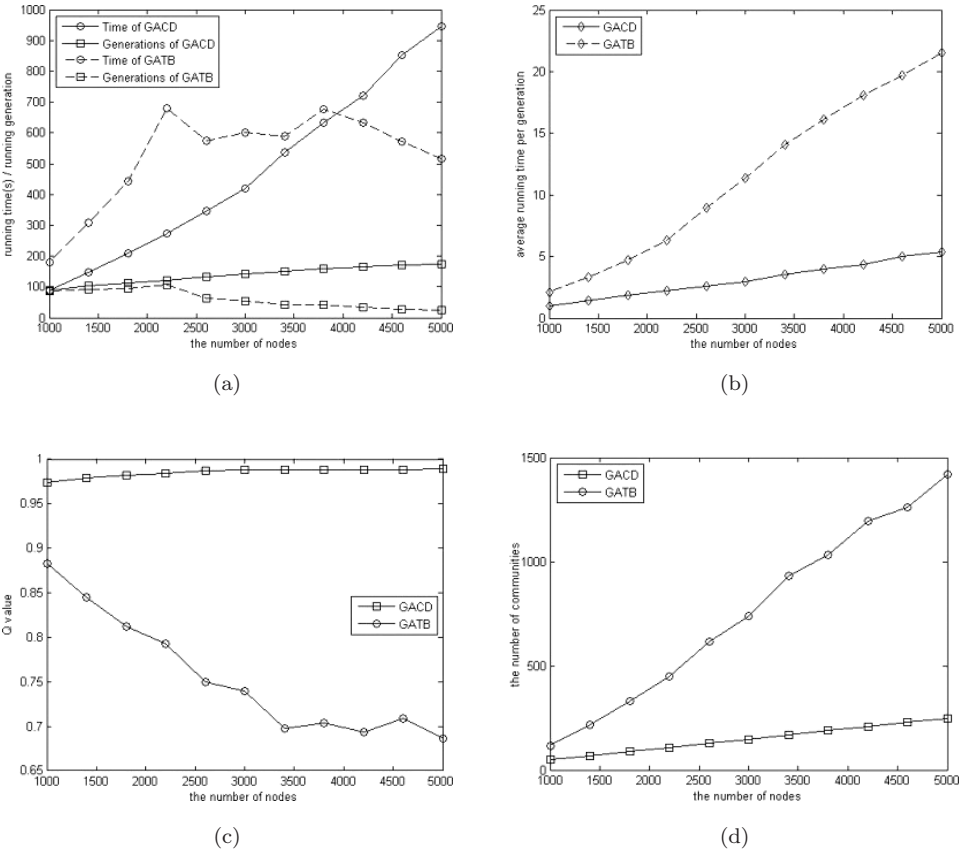

Fig. 5. Results of the scalability test. (a) is the relationship between the running time (generation) and the problem scale. (b) is the relationship between the average running time per generation and the scale. (c) is the relationship between the Q values obtained and the scale. (d) is the relationship of the number of communities found and the scale.

it becomes more difficult for GATB to reveal the correct community structures, because its Q value decreases (see Fig. 5(c)) and the number of communities booms incorrectly (see Fig. 5(d)). In contrast, GACD finds almost all the correct numbers of communities with high Q values. This experiment indicates that GACD is an effective algorithm for large-scale problems, because it remains the high accuracy with a small time cost as the problem scale increases.

4.4. *Stability experiment*

Note that as a stochastic algorithm, different runs of GACD in principle may yield different partitions. To check the consistency and stability of the proposed method, we test GACD on the e-mail network [6] with an obvious community structure and a random network with no community patterns, each with 50 runs. Figure 6 presents the results in a matrix, which shows the fractions out of the 50 runs

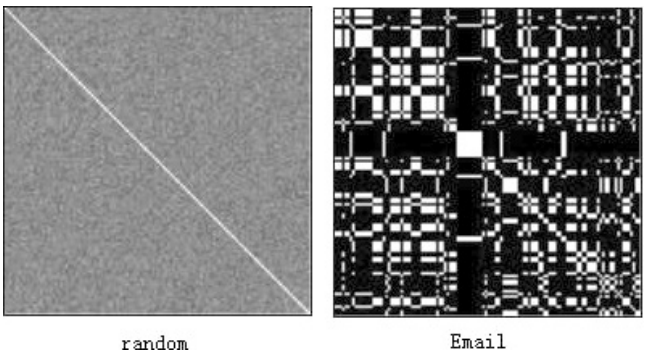

Fig. 6. Fraction of nodes classified in the same partition over 50 runs of the algorithm. The color of cell(i, j) corresponds to the fraction that node i and j belong to the same partition. A pure black dot indicates that the fraction is equal to 1 and a pure white one indicates that the fraction is equal to 0.

that pairs of nodes are classified in the same partition. A pure black dot indicates that the fraction is equal to 1 and a pure white one indicates that the fraction is equal to 0. We can see that the dots in the email network matrix are approximately either black or white. This demonstrates that GACD, as a stochastic algorithm, is able to generate approximately the same solution in each run in the case when a community pattern exists in the network. On the other hand, the matrix of the random network is uniformly grey, which indicates no community structure exists in this network and GACD returns random results in each run.

5. Conclusion

This paper proposes a GACD in large-scale networks through optimizing the modularity Q. The locus-based adjacency encoding schema is used to represent a community partition. The encoding schema is suitable for the community detection problem, and has the following advantages: (1) the search space can be distinctly reduced; (2) the number of communities can be automatically determined; (3) the crossover and mutation operation can be easily designed. Following the encoding schema, effective crossover and mutation operators are designed. Extensive experiments are carried out to test GACD. Compared to GN, GN Fast, and GATB, GACD can discover the most correct community partitions and the maximum Q for the most of the artificial and real networks with less running time. GACD is also stable although it is a stochastic algorithm. Most importantly, the experiments show that, with the fixed and reasonable population sizes, GACD can obtain good solutions with small time costs for the large-scale complex networks.

Recently, Fortunato and Barthelemy have mathematically showed that the optimization of modularity has a resolution limit, that is, modularity optimization fails to find small communities in large networks [7], and thus the modularity may not be a good metric for community detection. Although using modularity as the objective

function, the GA we proposed (especially the genetic representation) is flexible and can easily adapt to other optimization objectives.

Acknowledgments

This work is supported by the National Science Foundation of China (No. 60905025, 90924029). It is also supported the National High-tech R&D Program of China (No.2009AA04Z136) and the National Key Technology R&D Program of China (No.2006BAH03B05).

References

[1] Arenas, A., Fernandez, A. and Gomez, S., Multiple resolution of modular structure of complex networks, arXiv:physics/0703218v1 (2007).
[2] Brandes, U., Delling, D., Gaetler, M. *et al.*, On modularity clustering, *IEEE Trans. Knowl. Data Eng.* **20**(2) (2008) 172–188.
[3] Bron, C. and Kerbosch, J., Finding all cliques of an undirected graph, *Commun. ACM* **16** (1973) 575–577.
[4] Clauset, A., Newman, M. E. J. and Moore, C., Finding community structure in very large networks, *Phys. Rev. E* **70** (2004) 066111.
[5] Danon, L., Diaz-Guilera A., Duch, J. and Arenas, A., Comparing community structure identification, *J. Stat. Mech.: Theory Exp.* **9** (2005).
[6] Duch, J. and Arenas, A., Community detection in complex networks using extremal optimization, arXiv:cond-mat/0501368 (2005).
[7] Fortunato, S. and Barthelemy, M., Resolution limit in community detection, *PNAS* **104**(1) (2007).
[8] Girvan, M. and Newman, M. E. J., Community structure in social and biological networks, *PNAS* **99** (2002) 7821–7826.
[9] Girvan, M. and Newman, M. E. J., Finding and evaluating community structure in networks, *Phys. Rev. E* **69** (2004) 026113.
[10] Kernigham, B. W. and Lin, S., An efficient heuristic procedure for partitioning graphs, *Bell Syst. Tech. J.* **49** (1970) 291–307.
[11] Handle, J. and Knowles, J., An evolutionary approach to multiobjective clustering, *Trans. Evol. Comput.* **11** (2007) 56–76.
[12] Marina, M., Comparing clusterings: An information based distance, *J. Multivar. Anal.* **98** (2007) 873–895.
[13] Newman, M. E. J., The structure and function of complex network, *SIAM Rev.* **45** (2003) 167–256.
[14] Newman, M. E. J., Fast algorithm for detecting community structure in networks, *Phys. Rev. E* **69** (2004) 066133.
[15] Pei, J., Jiang, D. X., Zhang, A. D. *et al.*, On mining cross-graph quasi-cliques, in *The 12th ACM SIGKDD* (2006), pp. 228–237.
[16] Pizzuti, C., GA-Net: A genetic algorithm for community detection in social networks, in *PPSN* (2008), pp. 1081–1090.
[17] Pizzuti, C., Community detection in social networks with genetic algorithms, in *GECCO'08*.
[18] Pothen, A., Sinmon, H. and Liou, K-P., Partitioning sparse matrices with eigenvectors of graphs, *SIAM J. Matrix Anal. App.* **11** (1990) 430–452.
[19] Reichardt, J. and Bornholdt, S., Statistical mechanics of community detection, *Phys. Rev. E* **74**(1) (2006) 016110.

[20] Scott, J., *Social Network Analysis: A Handbook* (Sage Publications, London, 2002).
[21] Tasgin, M. and Bingol, H., Community detection in complex networks using genetic algorithm, arXiv:cond-mat/0604419 (2006).
[22] Watts, D. J. and Strogatz, S. H., Collective dynamics of 'small world' networks, *Nature* **393** (1998) 440–442.
[23] Zeng, Z., Wang, J., Karypis, G. *et al.*, Coherent closed quasi-clique discovery from large dense graph database, in *The 12th ACM SIGKDD* (2006), pp. 792–802.
[24] http://www-personal.umich.edu/ mejn/netdata/.
[25] http://en.wikipedia.org/wiki/Genetic_algorithm.