A New Genetic Algorithm for Community Detection

Chuan Shi, Yi Wang, Bin Wu, and Cha Zhong

Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing, China, 100876
shichuan@bupt.edu.cn

Abstract. With the rapidly grown evidence that various systems in nature and society can be modeled as complex networks, community detection in networks becomes a hot research topic in many research fields. This paper proposes a new genetic algorithm for community detection. The algorithm uses the fundamental measure criterion modularity Q as the fitness function. A special locus-based adjacency encoding scheme is applied to represent the community partition. The encoding scheme is suitable for the community detection based on the reason that it determines the community number automatically and reduces the search space distinctly. In addition, the corresponding crossover and mutation operators are designed. The experiments in three aspects show that the algorithm is effective, efficient and steady.

Keywords: complex network, community detection, genetic algorithm, modularity.

1 Introduction

Recent researches indicate that a large body of diverse systems in many different domains can be represented as complex networks [1, 2]. Examples include the internet, WWW, social networks, citation networks and etc. Most of these networks are generally sparse in global yet dense in local. They have vertices in a group structure that the vertices within the groups have higher density of edges while vertices among groups have lower density of edges [3]. This kind of structure is called the community which is an important network property and can reveal many hidden features of the given networks. Hence, community identification is a fundamental step for discovering what makes entities come together, but also for understanding the overall structural and functional properties of large network [4].

Since the community has played a crucial rule in complex network, community structure identification has created a great interest among physics and computer society. There has been many methods and algorithms proposed so far to reveal the underlying community structure in complex networks. The algorithms require the definition of community that imposes the limit up to which a group should be considered a community [23]. 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 [16], is widely used as a quality metric for assessing the partitioning of a network into communities. Most of the recent algorithms use the network modularity as quality metric like Newman's fast algorithm for very large networks [9] and the algorithm using extremal optimization [23]. The search for the largest modularity value is a NP-hard problem, since the space of all possible partitions grows faster than any power of the system size [21]. For this reason, recent many algorithms adopt various heuristic strategies to optimize this metric. However, many algorithms have a high computational complexity, and thus they are not suitable for a complex network with a large size. As a consequence, it is desire to design a high efficient algorithm.

This paper proposes a new genetic algorithm for community detection (GACD). The algorithm optimizes the modularity Q to obtain the community partition. A high efficient locus-based adjacency encoding scheme is applied to represent the community partition. The genetic representation not only reduces the search space distinctly but also determines the community number automatically. Based on the encoding scheme, the novel crossover and mutation operators are designed. These designs make the complex of GACD linear growth with the edges and vertices number of the network. Three experiments are done to validate the performance of GACD in three aspects. The first experiments use seven real social networks and compare GACD with three popular methods including GN [7], GN fast [18] and GA proposed by Tasgin and Bingol [22] (GATB). The results show that GACD obtains the maximum Q for most problems and it is much faster than GN and GATB. The second experiment on the random network and the email network shows that GACD is steady. In addition, the third running time experiment on a scalable network confirms that GACD has a linear complexity with the size of the network.

2 Related Works

There have been many different approaches and algorithms to analyze the community structure in complex networks. The algorithms use methods and principles of physics, artificial intelligence, graph theory and even electrical circuits. The spectral bisection methods [10] and the Kernighan-Lin [11] algorithm are early solutions to this problem in computer society. The spectral approach bisects graph iteratively, which is unsuitable to general networks. For the Kernighan-Lin algorithm, it requires a priori knowledge about the sizes of the initial divisions. In social network analysis (SNA), a group of algorithms focus on the discovery of the so-called cohesive sub-structure [3], including the cliques [12], and quasi-cliques [13, 14] ect. 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 [8] which groups similar vertices into larger communities.

One of the most known algorithms proposed as far, Girvan-Newman (GN) algorithm, is a divisive method by iteratively cutting the edge with the greatest betweenness value [7, 16]. The algorithm can generate an optimized division of the network with $O(m^3)$ time complexity according to the optimized network. Radicchi et al have proposed a similar methodology with GN [17] 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 have also proposed a fast clustering algorithm [18] with $O(n\log^2 n)$ time complexity on sparse graph which combine pairs of nodes to generate the maximal ΔQ iteratively until it becomes negative.(where m is the edge number and n is the node number)

An important issue in community detection is how to quantitatively measure the quality of the community partitions. A quantitative definition, network modularity, proposed by Grivan and Newman [7, 16] has been widely used in recent studies as the quality metric for assessment of partitioning a network into communities: $\mathrm{Q} = \sum_i (e_{ii} - a_i{}^2)$ 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_{ii} is the faction all the edges whose 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 [20]. The modularity Q is the special case of these two criterions. 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, since the decision version of modularity maximization is indeed *NP*-complete [21].

Many heuristic search algorithms have been applied to solve the optimization problem. The extremal optimization method, applied by Duch and Arenas, uses the artificial intelligence method in a recursive divisive manner [5]. The simulated annealing is used to obtain more results, but this method is computationally very expensive [24]. In addition, the genetic algorithm, as an effective optimization technique, has also been used to optimize Q [22]. However, the inefficient genetic representation makes the algorithm unsuitable for large scale problem in fact. Arenas, Fernadez and Gomez introduce the tabu heuristic to optimization the modularity, which also obtained a good performance [25].

The current algorithms are successful approaches in community detection. However, most algorithms have large time complexities that make them unsuitable for very large networks. In addition, some algorithms have data structures like matrices etc, which are hard to implement and use in very large networks. Most of the algorithms also need some priori knowledge about community structure like number of communities etc. However, it is impossible to know these values in real-life networks. All these factors make it desire to design a simple but high effective algorithm.

3 Genetic Algorithm for Community Detection

This section introduces the new genetic algorithm for community detection in detail, including the main components of the genetic algorithm: the framework of the algorithm, the crucial genetic representation and the corresponding operators etc.

3.1 Framework of the Algorithm

Our algorithm is based on optimization of network modularity using genetic algorithm. A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics which is inspired by evolutionary biology. Genetic algorithms 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. The evolution usually starts from a population of randomly generated individuals and happens in generations. 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 (crossover and mutation) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. The framework of the algorithm used in the paper can be seen in Fig.1.

To apply GA to the community detection problem effectively, there are much work to be done. As for the community detection problem, a special genetic representation should be designed to encode a community partition, 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 encodings may work well for networks with a few tens or hundreds of data points, but their performance breaks down rapidly for larger scale. The design of effective GA for community detection requires a close harmonization of the encoding, and the operators, so that the effective search space is reduced and the search can be guided effectively.

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 must be encoded in a character string (i.e. genotype) with the genetic representation, and inversely the genotype (i.e. a solution of the problem or an individual in the population) also can be decoded into a community partition. We employ the locus-based adjacency representation and illustrate it in Fig.2. In this graph-based representation, each individual g consists of n genes $g_1, g_2, \cdots, g_n$ 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 node i and j: in the resulting partition solution, they

```
Procedure GACD($size, gens, p_c \in [0,1], p_m \in [0,1]$)
  //$size$ is the size of the population size.  $gens$ is the running generation.
  // $p_c, p_m$ are the crossover and mutation ratio respectively, and $p_c + p_m = 1$.
  $P := \Phi$
  for each $i$ in 1 to $size$ do
    //initialization
    $s_i := initialize\_solution(i)$
    $evaluate(s_i)$
    $fillin(P, s_i)$
  end for
  for $g$ in 1 to $gens$ do
  //main loop
    $i := 0;\ P' := \Phi$
    while $i < size$ do
      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
      $i := i + 2$
      $evaluate(s'_i); evaluate(s'_{i+1})$
      $fillin(P', s'_i); fillin(P', s'_{i+1})$
    end while
    $update(P, P \cup P')$
  end for
  return $P[0]$
end procedure
```

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

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

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

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

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

Fig. 1. Main framework of GACD

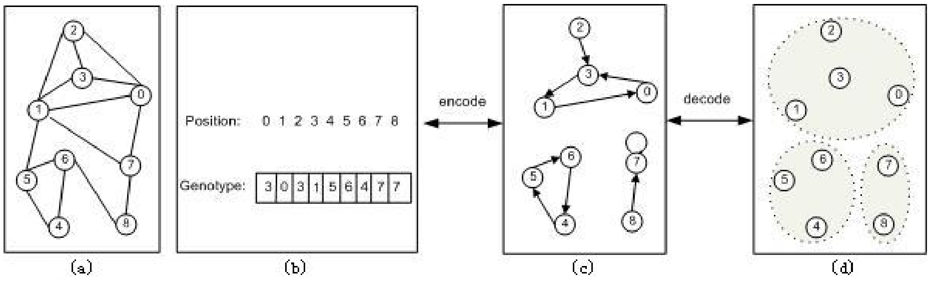

Fig. 2. Illustration of the locus-based adjacency representation. (a) shows the topologic of the graph representing a complex network. (b) shows one possible genotypes. (c) shows how (b) is translated into the graph structure, for example node 0 links to node 3, since gene g_0 is 3. (d) shows the partition result.

```
procedure decode
    $current\_cluster := 1$
    for each $i$ in 1 to $N$ do
        $cluster\_assign := -1$
    end for
    for each $i$ in 1 to $N$ do
        $ctr := 1$
        if $cluster\_assign_i = -1$ then
            $cluster\_assign_i := current\_cluster$
            $neighbour := g_i$
            $previous_{ctr} := i$
            $ctr := ctr + 1$
            while $cluster\_assign_{neighbour} = -1$
            do
                $previous_{ctr} := neighbour$
                $cluster\_assign_{neighbour} :=$
                        $current\_cluster$
                $neighbour := g_{neighbour}$
                $ctr := ctr + 1$
            end while
            if $cluster\_assign_{previous_{ctr}} \neq$ then
                    $current\_cluster$
                $ctr := ctr - 1$
                while $ctr >= 1$ do
                    $cluster\_assign_{previous_{ctr}} :=$
                            $cluster\_assign_{neighbour}$
                    $ctr := ctr - 1$
                end while
            else
                $current\_cluster :=$
                        $current\_cluster + 1$
            end if
        end if
    end for
    $number\_of\_clusters := current\_cluste$
end procedure
```

Fig. 3. Efficient decoding of the locus-based adjacency representation

will be in the same community. The decoding of this representation requires the identification of all connected components. All nodes belong to the same connected component are then assigned to one community. Note that, using a simple backtracking scheme, this decoding step can be done in linear time and the pseudo-code is illustrated in Fig.3.

According to the genetic structure, we find that the encoding approach can not represent all partition of nodes. For example, the genetic representation could not combine two disconnected subgraphs into one community. Someone could argue that the solutions with a good community structure may be not in the solutions space constructed by the genetic representation. Fortunately, Brandes etc. have analyzed the basic structural properties of the clustering with maximum modularity and proposed the following theoretical results [21].

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.

Although the modularity optimization has resolution limit [24], the community partition with a large modularity usually is a good solution. Since the genetic representation contains all possibility of connected subgraphs, these properties

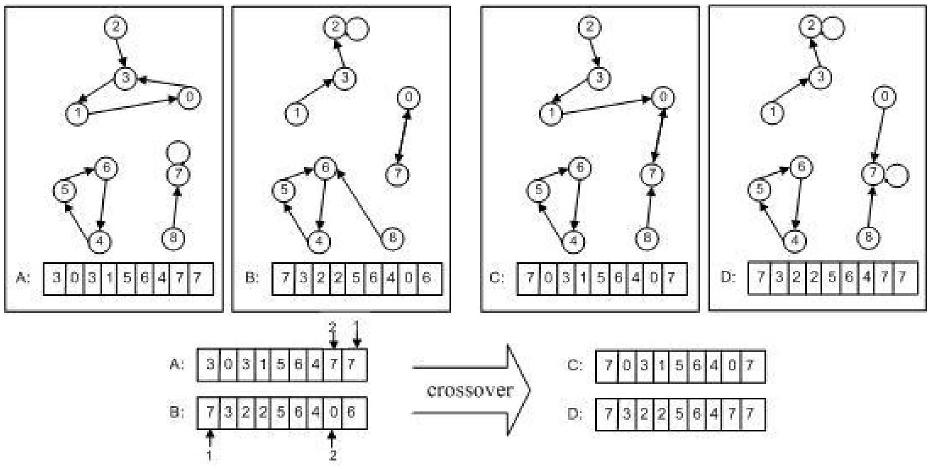

Fig. 4. Illustrate an example of the crossover operator. The position of genes selected from the source chromosomes are g_8 in A and g_0 in B.

promise that the community with a good structure can be represented with the genetic representation. Moreover, the representation is well-suited for the use with standard crossover operators such as uniform, one-point or two-point crossover.

3.3 Operators

Based the locus-based adjacency representation, the crossover operation in GACD is done by selecting two chromosomes randomly as illustrated in Fig.4. For simplicity, the chromosomes taking place in crossover are called source and destination respectively. First, a gene is selected randomly from the source chromosome, and then we iteratively search the gene values that the gene link to and transfer these values in source chromosome to the corresponding genes in the destination chromosome. An example of the operation of crossover on the encoding employed is shown in Fig.4. The exchange of gene segments is bidirectional. The crossover operator has the following advantages: (1) it is prone to replicate the good structures generated by evolution to the new individual; (2) it is efficient to generate the individual with different structure. The computational complexity is $O(l)$ (where l is the length of the gene segment, namely the size of the community selected.) l is usually smaller than n.

In the mutation operation, we randomly select some genes and assign them with other randomly selected adjacent nodes.

In the initialization process, we randomly generate some individuals. For each individual, each gene g_i randomly takes one of its adjacent nodes of node i.

3.4 Discussion

When GA is applied to a real problem, the most important issue is to design a suitable encoding scheme according to the characteristic of the problem. A good encoding scheme not only deduces the scope of the search space, but also facilitates to design the operators. As a consequence, the encoding scheme decides the performance of GA to some extent. The locus-based adjacency encoding scheme used 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 in the decoding step. Many methods need some priori knowledge like community number or threshold settled, whereas GACD need not any additional knowledge beforehand. Another important advantage is that the search space constructed by the representation is reduced distinctly. In the former genetic algorithm, Tasgin and Bingol [22] use a number ranging from 1 to n to represent the community a node belonging to, which cause the search space with the complexity $O(n^n)$. Brandes etc. cast the problem of maximizing modularity into an integer linear program (ILP) [21] with the complex $O(2^{n^2})$ in the search space. Since each node only links to its adjacent node, the complex of the search space constructed by locus-based adjacency representation is $O(d^n)$ (d is the degree of nodes). For most real problems, d is much smaller than n. The reduced searching space makes GACD search the more accurate solution with less time-consuming.

The fitness evaluation function (i.e., calculating 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 based on an individual has the complexity $O(m+n)$. The whole complexity of GACD is $O(gs(m+n))$ which is linear with the scale of the network. (g is the running generation, and s is the population size.) The running generation and population size directly affect the performance of the algorithm. However, increasing the population size or running generation does not yield better results after some point. As the constant parameters, these values (i.e., g, s) do not increase the time-complexity of the algorithm. As we know, most community detection algorithms have a large time-complexity [5]. Compared with these algorithms, the complexity of our algorithm is small.

4 Experiments

In order to validate the performance of GACD, this paper does the experiments in three aspects. The first experiment compares GACD with other three well known algorithms by seven popular social networks to verify the efficiency and effectiveness of GACD. The second experiment validates the steady of the algorithm through the running result on random network and real network with the obvious community structure. The third experiment observes the relationship of the running time and the size of the network. The experiments are carried out on a 3.4GHz and 2G RAM Pentium IV computer running Linux.

To validate the efficiency and effectiveness of GACD, we compare GACD with other three algorithms including GN, GN fast and GATB. Seven real social networks coming from ref [15] are used in the paper. These test problems are used widely as benchmark in community detection [6, 7, 9, 16, 18, 22]. The networks have different scales with the number of vertices ranging from 34 to 22963. The parameters of GA

Table 1. Parameters setting in the experiment

	Vertice Number	Edge Number	Population Size	Running generation	p_c	p_m
Karate (P1)	34	78	50	20	0.8	0.2
Football (P2)	115	616	50	70	0.8	0.2
Enron (P3)	120	576	50	100	0.8	0.2
Celegansneural (P4)	297	1179	50	100	0.8	0.2
Tomcat (P5)	1607	6235	500	500	0.8	0.2
Internet (P6)	8712	23305	500	500	0.8	0.2
as-22july06 (P7)	22963	48436	500	2000	0.8	0.2

Table 2. Comparing results of four different algorithms. N represents the number of communities; Q is the modularity Q value; and T represents the running time (the unit is second). P1-P7 is the problems in Table 1.

	GN			GN Fast			GATB			GACD		
	N	Q	T	N	Q	T	N	Q	T	N	Q	T
P1	5	0.401	1	3	0.381	1	5	0.379	1	4	**0.419**	1
P2	10	0.597	2	7	0.577	1	10	0.575	1	11	**0.604**	1
P3	7	0.484	1	5	0.483	1	15	0.436	1	5	**0.508**	1
P4	33	0.312	200	4	**0.369**	1	79	0.274	1	6	**0.369**	1
P5	27	**0.8**	13230	27	**0.8**	4	234	0.734	659	43	0.797	85
P6	--	--	--	29	0.520	34	3630	0.151	6938	150	**0.530**	550
P7	--	--	--	39	0.637	114	--	--	--	192	**0.639**	2138

and GACD are same, and they are shown in Table 1. The experimental results are the average values of ten runs, and they are shown in Table 2. For most problems, GACD achieves the maximum Q, which shows that GACD is an effective algorithm for community detection. Considering the running time, GN Fast is fastest, and GACD is faster than other two algorithms. For the problem with large scale (e.g., P6, P7), GN is not able to obtain a result in the reasonable time, since the algorithm needs a huge memory. It is similar to GATB. Through GATB and GACD both are based on genetic algorithm, their different genetic representations cause the different performance. Since the complex of the search space constructed by GACD is smaller than that of GATB, GACD finds the larger Q value with less time. GATB uses a number ranging from 1 to n to represent the community a node belonging to, which causes that there exists so many clusters (up to n) in the beginning phrase. It costs huge memory to store the clusters, and it makes the algorithm very inefficient in fact.

Note that since the core of GACD is stochastic, different runs could yield in principle different partitions. We have performed 50 runs of the algorithm for the email network with the obvious community structure and for a random network with the same number of links and nodes to check the consistency of the proposed method. In Fig. 5, we present the results of the fraction of times a couple of nodes are classified in the same partition. The community structure is clearly revealed for the email network; whereas this structure is inexistent for the random network. The experiment demonstrates that GACD is a steady algorithm.

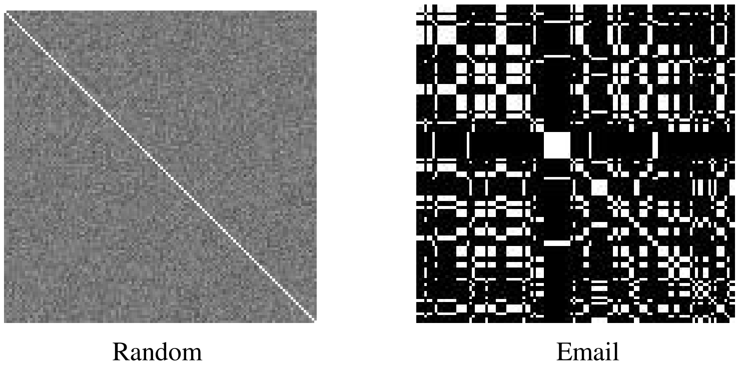

Fig. 5. Fraction of nodes classified in the same partition over 50 realization of the algorithm. The color of the position (i,j) corresponds to the fraction of times that node i and j belong to the same partition.

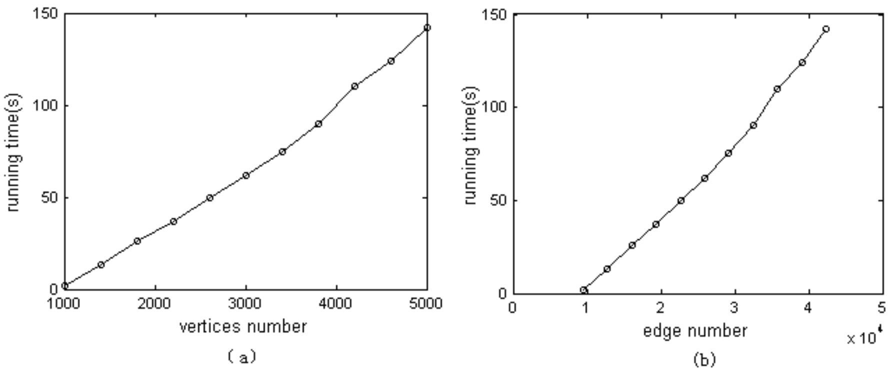

Fig. 6. Relationship of the running time and the size of network. (a) is the relationship of the running time and the vertices number. (b) is the relationship of the running time and the edge number.

In order to further validate the efficiency of GACD, we do experiments to observe the relationship of the running time and the size of the network. The example is the network used in ref [24]. The scalable network K-m,n is made of m identical complete graphs with size of n nodes, and a link connects two adjacent complete graphs. In the experiment, n is 20, m ranges from 50 to 250 with an interval of 20. To compare the running time fairly, the population size is 200, P_c and P_m are 0.8 and 0.2 respectively, and the stopping criterion of GACD is that the algorithm has converged (i.e. the best individual and worst individual have the same fitness). The result is the average of 10 runs as shown in Fig. 6. It is clear that the running time increases linearly with the vertices number and edge number, which confirm the complexity of GACD (see section 3.4).

5 Conclusion

This paper proposes a new genetic algorithm for community detection (GACD). GACD optimizes the modularity Q to detect community with a special genetic algorithm. The locus-based adjacency encoding scheme is used to represent a community partition. The encoding scheme is suitable for the community detection, and has the following advantages: (1) the search space can be reduced distinctly; (2) the community number can be determined automatically; (3) the decoding process is highly efficient. According to the encoding scheme, the effective crossover and mutation operators are designed. The theorem analysis shows that GACD has a linear complexity with the vertices and edge number of the network. Three experiments are done to verify GACD's performance. In first experiment, seven real social networks are used to validate the effectiveness and efficiency of GACD. Compared with GN, GN fast and GATB, GACD finds the maximum Q for most problems, and its running time is smaller than GN fast and GATB. The second experiment shows that GACD is steady for the network with the apparent community structure. Through observing the relationship of the running time and the size of network, the experiment further confirms that the running time of GACD is linear growth with the scope of the network.

Recently, Fortunato and Barthelemy have showed mathematically that the optimization of modularity has a resolution limit that is, modularity optimization fails to find small communities in large networks [24], and thus the modularity optimization may be not a good method to community detection. However, the genetic algorithm (especially the genetic representation) we proposed can be used as a general optimization technology in complex network.

Acknowledgments. The work is also supported by National Science Foundation of China (No. 60402011, 60805041) and National Eleven Five Year Scientific and Technical Support Plans of China under grant (2006BAH03B05).

References

1. Watts, D.J., Strogatz, S.H.: Collective Dynamics of Small World'Netwoks. Nature 393, 440–442
2. Newman, M.E.J.: The Structure and Function of Complex Network. SIAM Review 45, 167–256
3. Scott, J.: Social Network Analysis: A Handbook. Sage Publications, London (2002)
4. Milo, R., Itzkovitz, S., et al.: Network Motifs: Simple Building Blocks of Complex Networks. Science 298, 824–827
5. Danon, L., Diaaz-Guilera, A., Duch, J., Arenas, A.: Comparing Community Structure Identification. Journal of Statistical Mechanics: Theory and Experiments 9 (2005)
6. Newman, M.E.J.: Modularity and Community Structure in Networks. PNAS 103, 8577
7. Girvan, M., Newman, M.E.J.: Community Structure in Social and Biological Networks PNAS 99, 7821–7826
8. Han, J.W., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann publishers, San Francisco (2006)
9. Newman, M.E.J.: Fast Algorithm for Detecting Community Structure in Networks. Physical review E 69, 066133 (2004)

10. Pothen, A., Sinmon, H., Liou, K.-P.: Partitioning Sparse Matrices with Eigenvectors of Graphs. SIAM J. Matrix Anal. App. 11, 430–452
11. Kernigham, B.W., Lin, S.: A Efficient Heuristic Procedure for Partitioning Graphs. Bell System Technical Journal 49, 291–307
12. Bron, C., Kerbosch, J.: Finding all Cliques of an Undirected Graph. Communications of the ACM 16, 575–577
13. Pei, J., JIang, D.X., Zhang, A.D., et al.: On Mining Cross-graph Quasi-cliques. In: Proc. The 12th ACM SIGKDD, Philadephia, pp. 228–237 (2006)
14. Zeng, Z., Wang, J., Karypis, G., et al.: Coherent Closed Quasi-Clique Discovery from Large Dense Graph DataBase. In: Proc. The 12th ACM SIGKDD, Philadephia, pp. 228–237 (2006)
15. `http://www-personal.umich.edu/~mejn/netdata/`
16. Girvan, M., Newman, M.E.J.: Finding and Evaluating Community Structure in Networks. Physical Review E 69, 026113
17. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and Identifying Communities in Networks. PNAS 101, 2658
18. Clauset, A., Newman, M.E.J., Moore, C.: Finding Community Structure in Very Large Networks. Physical Review E 70, 066111
19. Reichardt, J., Bornholdt, S.: Statistical Mechanics of Community Detection. Physics Review E 74(1), 016110 (2006)
20. Arenas, A., Fernandez, A., Gomez, S.: Multiple Resolution of Modular Structure of Complex Networks. arXiv:physics/0703218v1 (2007)
21. Brandes, U., Delling, D., Gaetler, M., et al.: On Modularity Clustering. IEEE Transactions on Knowledge and Data Engineering 20(2), 172–188 (2008)
22. Tasgin, M., Bingol, H.: Community Detection in Complex Networks using Genetic Algorithm, arXiv:cond-mat/0604419 (2006)
23. Duch, J., Arenas, A.: Community Detection in Complex Networks using Extremal Optimization. arXiv:cond-mat/0501368 (2005)
24. Fortunato, S., Barthelemy, M.: Resolution Limit in Community Detection. Proceedings of the National Academy of Sciences 104(1) (January 2007)
25. Arenas, A., Fernandez, A., Gomez, S.: Analysis of the Structure of Complex Networks at Different Resolution Levels, physics 0703218v2, January 15 (2008)