Effective Algorithm for Detecting Community Structure in Complex Networks Based on GA and Clustering

Xin Liu[1], Deyi Li[2], Shuliang Wang[1], and Zhiwei Tao[1]

[1] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
[2] China Institute of Electronic System Engineering, Beijing, 100039, China
tsinllew@gmail.com

Abstract. The study of networked systems has experienced a particular surge of interest in the last decade. One issue that has received a considerable amount of attention is the detection and characterization of community structure in networks, meaning the appearance of densely connected groups of vertices, with only sparser connections between groups. In this paper, we present an approach for the problem of community detection using genetic algorithm (GA) in conjunction with the method of clustering. We demonstrate that our algorithms are highly effective at discovering community structure in both computer-generated and real-world network data, and show how they can be used to shed light on the sometimes daunting complex real-world systems of scale-free network structure.

Keywords: complex network, community structure, modularity, clustering.

1 Introduction

There have been numerous and various complex networks with the development of science, technology and human society. A property that seems to be common to many networks is community structure [3], the division of network nodes into communities or modules within which connections are dense, but between which they are sparser. Community detection, which can help us simplify functional analysis, is potentially very useful. To evaluate the accuracy of the community structure Newman and Girvan devised a quantitative measure called modularity Q. It is defined in [4] as:

$$Q = \sum_i (e_{ii} - a_i^2) \tag{1}$$

Q is the addition of the modularity q of all the communities, Q=sum (q_i). For each community i, q_i is calculated as e_{ii}, the fraction of the links that fall within the communities, minus a_i^2, the expected value of the same number of links distributed randomly in spite of the community structure.

Y. Shi et al. (Eds.): ICCS 2007, Part II, LNCS 4488, pp. 657–664, 2007.

Adopting this successful measure as the evaluation function based on optimization, many attempts were proposed to tackle the module identification problem. Guimerà and Amaral put forward a simulated annealing approach [1]. Duch and Arenas suggested an algorithm based on extremal optimization [7]. Newman proposed in [5] a fast method by means of creating the hierarchy following an agglomerative strategy to address this issue. Pujol, Béjar and Delgado recently developed an algorithm [6] that is more efficient while maintaining its accuracy. In this paper, we introduce another approach based on the genetic algorithm (GA) and clustering, which is efficient, sensitive and able to analyze very large networks.

2 The Method of Extracting Communities Based on GA

2.1 Bi-partitioning Strategy

Community detection problem is an NP hard puzzle [5]. It is in practice infeasible to carry out an exhaustive search of all possible divisions for systems larger than 30 or 40 vertices. However, by performing GA, as we will shortly show, one can find out a fine partition that divides the network into two parts at ease. In order to detect more modules, we adopt a smart scheme. Our approach, with reference to [1], employs the repeated division into two: we first divide the network into two parts, then divide those parts, and so forth. When partition a subgraph, we first isolate it from the rest of the network, then perform a 'nested' GA. The result of the nested GA is a partition of the subgraph into two new modules, which we accept or exclude according to the global modularity. Namely if we find that the proposed split makes a negative contribution to the total modularity, we leave the corresponding subgraph undivided. When the entire network has been decomposed into indivisible subgraphs in this way, the algorithm ends.

2.2 Dividing a Network into Two Using GA

Given a network (or a subgraph that is separated from the entire network), we divide it into two parts on the criterion of modularity. So the goal is to:

$$\underset{q \in H}{Maximize} \quad Q(q) \tag{2}$$

where H is all possible ways of bi-partitions. The GA used to solve such an optimization problem consists of several steps and the implementation of them is explained as follows.

Problem Encoding
A divisive resolution that bi-partition a network containing k vertices is represented by a binary string:

$$(num_1, num_2, \ldots, num_k), \ num_i = 0, 1 \ (i=1,...,k),$$

where $num_i = 0$ denotes that vertex num_i is distributed to the first part, while $num_i = 1$ means the other assignment.

Evaluation Function
We choose modularity Q as the fitness value of the chromosome. Then, the evaluation function is defined in (1).

Mutation Operation
In this process, each chromosome in the population except the fittest one undergoes a random one-bit flip. If the mutation has improved the fitness value of the chromosome, we adopt it for certain. Otherwise, we accept or reject the mutated string according to a variable probability:

$$pro = 1 - t / maxgen , \tag{3}$$

where t, $maxgen$ denotes the current generation number and the maximal generation number respectively. This mechanism, analogous to simulated annealing, causes the acceptance probability of the deteriorated mutation being large initially, where we can search the space uniformly, and it being very small at later stage, where we can search the space very locally and escape the local optima.

Local Hill-climbing Operation
We first select the fittest chromosome u in the population, then mutate on one of its bit obtaining u'. Only if the random mutation has improved the fitness value did we replace u by u', otherwise nothing should be done. This operation is significant in that the algorithm depends highly on it to search the global optimization when the evolution reaches the later stage.

Termination Condition
The algorithm is terminated on condition that at least one of the following two terms is satisfied:

1. The fittest chromosome in the population has not been improved for a desired number of generations.
2. The number of the current generation is beyond limitation.

Pseudo-code
Finally, the pseudo-code of our genetic algorithm is described as follows:

```
program Bi-Partition a network (Output)
begin
  Step 1: current generation t:=0;
  Step 2: initiate pop(t) randomly;
  Step 3: while(termination condition is not satisfied)
      Step 3.1: mutation operation;
      Step 3.2: local hill-climbing operation;
      Step 3.3: eliminate a quarter of the inferior
chromosomes and replace them by the top 25% ones;
      Step 3.4: t := t + 1;
  Step 4: convert the fittest chromosome into a
bipartition solution;
end.
```

Complexity Analysis

Some of parameters used in GA are listed in Table 1. Concerning the computational time, the most consuming step is the fitness evaluation process. For a given chromosome with the string length of l, the running time of fitness evaluation operation is $O(l^2)$. However, if just flip a single bit of a string, we only need to modify the previous fitness value to fulfill the re-evaluation, whose cost would be as low as $O(l)$. This is also the reason of not involving the crossover operation in our algorithm, as re-evaluating the offspring produced by crossover entails unbearable complexity of $O(l^2)$. Thus, the complete fitness evaluation operation of $O(l^2)$ complexity is only performed when assessing the first randomized chromosomes. In addition, we do the fitness re-evaluation of $O(l)$ complexity *popsize* times per generation. Therefore, in the worst case, the complexity of the GA would be:

$$popsize \times O(l^2) + (maxgen \times popsize) \times O(l) . \tag{4}$$

If the network with n vertices is ultimately divided into s communities, we would perform the GA $2s - 1$ times. Yet, the length of the chromosome, which is n in the first time, decreases dramatically rather than unchangeable. Hence, a rough upper limit of the total computation cost could be estimated as:

$$2 \times popsize \times O(n^2) + (\log_2 s + 2) \times (maxgen \times popsize) \times O(n) . \tag{5}$$

3 Community Detection in Scale-Free Networks of Large Size

When confronted with the large real-world graphs that have thousands or millions of vertices, the GA becomes incompetent. To classify communities in such graphs, we should exploit their scale-free nature. First, the power law distribution tells us that most vertices are of low degree, but a few nodes have a very high degree. Second, vertices in scale-free networks are not created equally − some are more "important" than others. Based on these two views, we propose a method of clustering which could guide the search of dense modules. Our notion can be expressed vividly as follows. For each vertex with higher degree, we take it as a center and draw a circle around it. One can imagine intuitively that if the scope is small enough, all nodes in it would belong to the same community. In other words, if we allot the centered node to a certain community, the other vertices in the same circle will be assigned to the same module too. Thus, we could omit the massive trivial nodes in the same ring, and treat them as a unified cluster represented by the central vertex.

To realize the above idea, we first find out the most connected $r \times n$ vertices (hubs), supposing r is the ratio of the superior nodes and n denotes the dimension of the network. Next, we affiliate each vertex with its nearest hub. Upon that, the initial network G has been preparatively classified into $r \times n$ clusters. Then our consequent GA only needs to consider for each cluster to which community it belongs, with the chromosome coding on the typical hub and hence the efficiency of the algorithm can be enhanced enormously.

Generally speaking, our method is a joint strategy of GA and clustering, which allows us to analyze very large networks in reasonable time while maintaining

accuracy. To be noted, the efficiency and accuracy of the algorithm is affected by the proportion of hubs we mined. As a matter of fact, if the hubs are excessive, there would be too many clusters for the algorithm to be efficient. On the contrary, if the number of hubs is too small, although the algorithm is efficient, the partition would be ill-constructed. Therefore, the quality-efficiency trade-off of our algorithm is really subject to the value of r.

On a network with n vertices and m edges, the running time of clustering operation is dependent on r. Specifically, the complexity is between $O(n)$, where $r = 1$, i.e., all nodes are hubs, and $O(mn)$, where r is so small that only one hub exists. According to (5), the sequential divisional algorithm has $2 \times popsize \times O((r \times n)^2) + (\log_2 s + 2) \times (maxgen \times popsize) \times O(r \times n)$ complexity. If the values of all parameters are set as Table 1, the total computational cost would approximately be:

$$O(mn) + 2 \times 10^5 \times (\log_2 s + 2) \times O(n) + 8 \times O(n^2) . \quad (6)$$

This is the complexity of the worst case, considering every time the GA would converge much earlier before it evolving to the max generation. Therefore, generally speaking, our algorithm entails a sort of operation with cost $O(n(m+n))$ or $O(n^2)$ on a sparse network.

Table 1. Summary of parameters used in the algorithm

Parameter	Meaning	Value
popsize	size of the population	100
maxgen	maximal generation number	10000
ungen	max number of generations for unimproved fittest chromosome	100
r	fraction of mined hubs	0.2
s	number of identified communities	network dependent

4 Experiments and Results

4.1 Computer-Generated Networks

As a first example of the working of our algorithm, we employ a widely used computer-generated random graph with known community structure [4]. It consists of 128 vertices divided into four groups of 32. Each vertex has on average k_{in} edges connecting itself to other nodes of the same group and k_{out} edges to members of other groups, satisfying $k=k_{in}+k_{out}=16$. As k_{out} becomes larger and larger, the resulting community structure becomes weaker and weaker, and thus the graph poses greater and greater challenges to our algorithm. This time, we set $r=1$ to get the best partition result. We compare our algorithm to the GN algorithm [3], the most frequently referenced one, and SA algorithm [1], the most accurate one at present. As Fig.1 shows, the algorithm performs well, correctly identifying all of vertices for values of $k_{out}/k \leq 0.3$. When k_{out}/k approaches 0.4, more than 97.5% of vertices are still classified accurately. Only for $k_{out}/k \geq 0.45$ does the classification begin to deteriorate markedly.

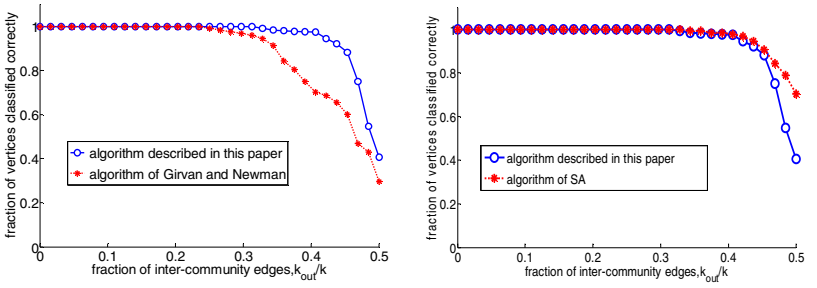

Fig. 1. The fraction of vertices correctly identified by our algorithm, GN algorithm and SA algorithm in the computer-generated graphs described in the text. Our algorithm, which significantly outperforms the GN algorithm and is comparable to the SA algorithm for most cases, could be able to reliably identify modules in a network whose nodes have as many as 40% of their connections outside their own module.

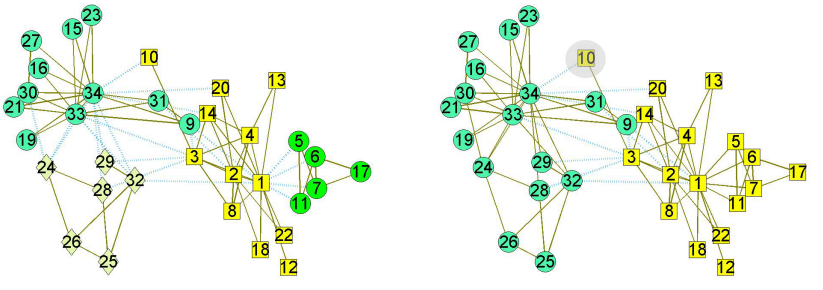

Fig. 2. The division results of the karate club network by our algorithm. On the left, we apply our algorithm directly to the network obtaining 4 communities with Q=0.418803, the biggest modularity at present. On the right, the result of improved algorithm by a threshold of Δ=0.02 is satisfying, only the number 10 vertex is classified wrongly according to the actual structure of the network.

4.2 Zachary's Karate Club Network

Now we turn to some applications of real-world network. The first one is Zachary's karate club network [4], which is taken from one of the classic studies in sociology. We apply our algorithm to it and the original network is segmented into 4 parts with Q=0.418803 (see Fig.2, left). In order to avoid the graph being decomposed into more parts, we set a threshold Δ=0.02. Only if the contribution to the total modularity of the proposed split of the isolated subgraph is larger than Δ, we do the division. In this way, we carry out our experiment for the second time. The algorithm works perfectly, finding most of the known split of the network into two groups with Q=0.371795, only one vertex, number 10, is classified wrongly (see Fig.2, right). In fact, if we distribute the number 10 vertex to the right group, the corresponding Q would fall to 0.371466. As can be seen form the picture, the number 10 vertex is connected to the two parts with only one link respectively. So both assignments could be accepted from the view of "community structure", whether we distribute it to the left part or the right one.

4.3 UML Class Diagram

This time we employ a UML class diagram with 224 vertices and 541 edges. Again we set r to 1 in the experiment. The program ends rapidly and segments the original graph into 10 parts with the modularity of 0.531982. Fig.3 shows the image of the network after performing our algorithm.

Fig. 3. The identified 10 communities of UML class diagram

4.4 Other Real-World Networks

In order to further verify the significance of the algorithm proposed in this paper, we apply it to several other networks of different sizes. In all the tests, we only work with the biggest connected component, removing all multiple relations and self-referenced edges. The results given in Table 2 provide a useful quantitative measure of the success of our algorithm when applied to real-world problems.

Table 2. The result of several other real-world networks by our algorithm. The applied networks, ranging from 1458 to 28502 vertices, are all of scale-free structure. The algorithm successfully identifies the communities in reasonable time (The runs are executed on a desktop computer of Pentium 4, 3.0GHz).

Networks	Num of Vertices	Num of Communities	Q	r	Time(s)
Protein interaction network	1458	12	0.738382	0.3	5.1
Word association graph	7192	22	0.434555	0.2	62.4
Co-authorship graph	28502	68	0.578198	0.1	1724.0

5 Conclusion

In this paper we describe an algorithm, based on repeated subdivisions, to extract community structure from complex networks. The running time of the algorithm is $O(n^2)$ on a sparse network, where n is the number of nodes. Experiments show that our method performs well in both efficiency and accuracy. The innovative aspects of

this paper are as follows. First, we employ the GA and devise some specific tricks to successfully solve the intractable problem. Second, we exploit the scale-free nature of real-world complex networks, and propose a method, which combines the idea of clustering, to manage the daunting real-world systems of medium and large sizes in reasonable time.

Acknowledgments. The authors thank Zhijian Wu, Zengyang Li for their wonderful suggestions and Baohua Cao, Bin Liu for providing data. This research is supported by the National Grand Fundamental Research 973 Program (2006CB701305), the State Key Laboratory of Software Engineering Fund (SKLSE05-15), and the Ministry Key GIS Laboratory Fund (wd200603).

References

1. Roger Guimerà and Luís A. Nunes Amaral.: Cartography of complex networks: modules and universal roles. Journal of Statistical Mechanics: Theory and Experiment. 1742-5468/05/P02001
2. L.Danon, J.Duch, A.Diaz-Guilera, and A. Arenas.: Comparing community structu-re identification. J. Stat. Mech. p. P09008 (2005)
3. Michelle Girvan and M. E. J. Newman.: Community structure in social and biological networks. 2002, Proc. Natl. Acad. Sci., 99, 7821
4. M. E. J. Newman and M. Girvan.: Finding and evaluating community structure in networks. Phys. Rev. E, 69, 026113
5. M. E. J. Newman.: Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 066113 (2004)
6. Josep M. Pujol, Javier Béjar, and Jordi Delgado.: Clustering algorithm for determining community structure in large networks. Phys. Rev. E 74, 016107 (2006)
7. Jordi Duch and Alex Arenas.: Community detection in complex networks using extremal optimization, Phys. Rev. E 72, 027104 (2005)
8. M. E. J. Newman.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 036104 (2006)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
10. M. E. J. Newman.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences of the United States of America 103:8577 2006