International Journal of Applied Science and Engineering
2020. 17, 2: 157-173

Improving Efficiency of Graph Clustering by Genetic Algorithm Using Multi-Objective Optimization

Yu-Ching Lu[a*] and Goutam Chakraborty[a, b]

[a]*Graduate School of Software Information Science, Iwate Prefectural University, Iwate, Japan*
[b]*Sendai Foundation of Applied Information Sciences, Japan*

Abstract: Information networks, like social networks on the internet, evolve naturally without constraint on degree or connectivity. The distribution of degree of such network is exponential, with a few nodes having a large degree, forming what is called a scale-free network. Scale-free networks form communities, or clusters of nodes. Discovering such clusters is the most important step for analysis of the information network. This is a NP-hard problem. Recently, genetic algorithm based works are reported, where the optimum number of clusters evolve automatically. The optimization criterion, the modularity index, is used as the fitness function of the chromosome. We observed that, if the diameter of the clusters are constrained to a lower value, during evolution of genetic search, a faster convergence is achieved. We used a multi-objective genetic search with two optimization criteria: the modularity index (the higher the better), and the largest diameter (the lower the better) of the communities defined by a chromosome. Simulation results, using network with known communities, show that using this multi-objective GA the result is stable, and achieves better modularity compared to the most often used Louvain algorithm.

Keywords: Graph clustering; social network analysis; multi-objective optimization; genetic algorithm.

1. Introduction

Clustering of information networks and social networks reveal important knowledge about relation between member nodes, which is exploited to many applications. In addition, clustering of a very high dimensional data, necessary for mining, is difficult. It is more efficient to analyze by first representing it as a graph, where data units are nodes and a link represents a defined relation between two data units. Analysis of such graph reveals interesting, useful and important knowledge hidden in the data, in an efficient way.

Information, which is evolved without a central control, when represented as graphs, takes the structure of scale-free network. It forms clusters or communities around important nodes. One example of scale-free network is scientific papers, where papers are nodes and relation like citation or co-authorship is used as links. In fact, many physical phenomenon occurring in nature like gene interaction, aftershocks from a big earthquake, neuron excitation in the brain etc., exhibit scale-free properties.

Definition of the nodes and the links of the graph depends on the application. For scientific papers, nodes could be author and links representing co-authorship. The central node of a cluster of such network will reveal an important author. On the other hand, nodes could be

*Corresponding author; e-mail: g236q203@s.iwate-pu.ac.jp
Received 23 May 2019
doi:10.6703/IJASE.202005_17(2).157 Revised 28 January 2020
 Accepted 26 March 2020

papers and directional links represent papers in citation. Here, the central nodes will reveal seminal works in the field. Similarly, representing web-sites as nodes and hyperlinks as directed links, we can rank websites and set the order according to their importance. This ranking algorithm, which does not consider the content of the page, is known as Page Ranking. This was proposed by Brin and Page [1]. Variations of page ranking is now used in all internet search engines.

Naturally evolved networks, like social network, have scale-free property. Scale-free network (SFN) is also known as Small-world network (SWN) [2-5]. In SFNs, distance between two randomly selected nodes is proportional to the logarithm of the total number of nodes in the network. Due to preferential connection, Small-world networks form communities, where nodes within a community are strongly interlinked, and between communities the connection is weak with a few links. Most of the works on community detection, for large networks, are heuristic. As the problem is multimodal, heuristic algorithms often converge in local minimum. In recent years, there are a few works proposing solutions using genetic algorithm (GA) [6-8]. But, GA approaches too often convergence in local minimum, and are slow to converge. In this work, we use GA. Our motivation is to find communities in an efficient way and achieve stable solutions. We used two optimization objectives, which are clumped into a fitness function. Various simulation results ensure stability and better quality of results. We compared the proposed algorithm with the most popular heuristic algorithm namely Louvain [9].

The rest of the paper is as follows. Section 2 is about related works. In section 3, we explain the proposed multi-objective genetic algorithm. In section 4, we describe the data used for our experiments and experimental results. For our dataset, we know the optimum clustering results. We compare our results with optimum solutions. We also compared efficiency with simple genetic algorithm, which was our previous work [10]. Finally, we experimented with a real-life data, for which the cluster memberships are known but clusters are not well defined, having overlapping communities. We compare performance with Louvain algorithm [9], which is the most popular heuristic method. It is shown that our algorithm gives better and more stable solutions. The conclusion and future works are summarized in section 5.

2. Problem Definition and Related Works

Research in network science was initiated by Erdos and Renyi [11]. They proposed different random network models and their properties. Around the end of 20th century, a special type of random network, called Small-world network [5, 12, 13] attracted much attention, because of its ubiquity in many naturally evolving phenome. These networks often form closely knitted communities. Finding those communities efficiently, for very large networks, became an attractive research topic for many practical applications. A good survey of graph clustering is available at [14].

Different community detection algorithms are proposed. Their respective efficacy depends on the structure of the network. Partitioning of graph is a NP-hard problem. Algorithms were proposed using Lapacian matrix [15], or hierarchical graph partitioning. The subject received wide interest after publication of 1998 paper by Duncan et al. [12] and Barabasi et.al. [13]. We will discuss a few prevalent algorithms, including our previous works based on genetic algorithm [10, 16], following which we will describe the special features of this work based on multi-objective genetic algorithm.

The membership of a node in a community is defined as follows: the member nodes of a community have more links to other nodes of the same community than with nodes outside the community. For example, for the 10-nodes network in Figure 1, we can divide it into three communities of subset of nodes, C1 = {1, 2, 3}; C2 = {4, 5, 6, 7}; C3 = {8, 9, 10}. The number of links of node 2 with other nodes of C1 is 2, and links going outside its community is 1. The same is true for nodes in other clusters, like node 4 and 6 in cluster 2.

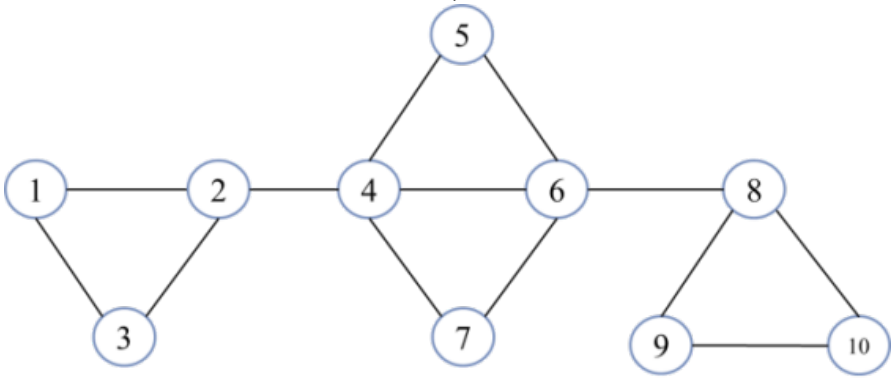

Figure 1. Clustering example of a ten nodes network.

Sometimes, overlapping communities are formed as shown in Figure 2. Here, the membership of node 13 and 14 are in two groups, as the number of links from 13 to members of both groups is 2, and the number of links from node 14 to members of both groups is 4. One way is to coalesce the overlapping clusters. But, depending on the motivation of clustering, one may be interested to identify overlapping clusters and common nodes of them. Palla et al. [17] used clique percolation technique to efficiently grouping the network.

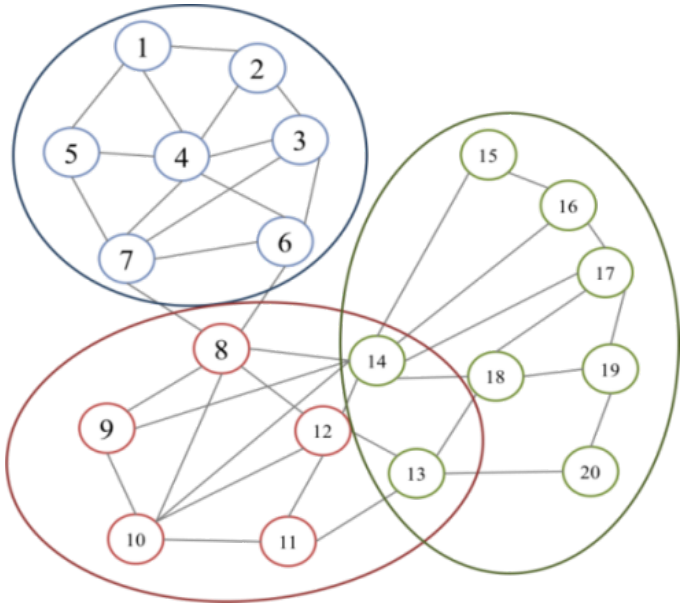

Figure 2. Example of overlapped communities.

An example of networks of hierarchical clusters is shown in Figure 3. As one zooms in, with higher resolution smaller clusters are revealed. This structure prevails in many real-life networks, like internet or wide area networks, power grid, etc..

The Louvain method [9] for community detection is an efficient way to find hierarchical clustering. The Modularity measure is the density of edges inside communities to edges outside communities. Optimizing this value is NP-hard, and heuristic algorithms are used. In the Louvain method, first small communities are found by optimizing modularity locally, following which small communities are grouped into one node and the algorithm is iterated. The method is similar to the earlier method by Clauset, Newman and Moore [18]. Louvain method is very efficient and could work even for networks with a billion nodes.

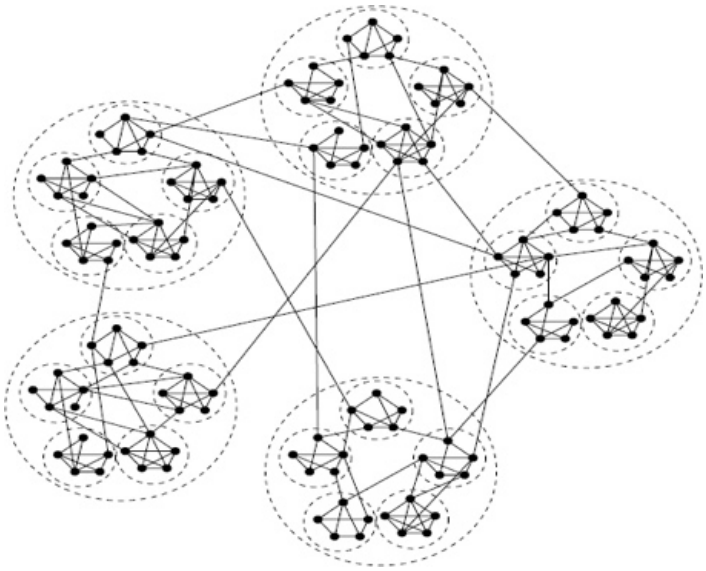

Figure 3. Example of hierarchical communities.

In recent years there are many works for community detection using genetic algorithm [10, 13, 16, 19, 20]. In our previous work in 2017 [10], the main contribution was that it could always find the optimum number of clusters and node assignments after sufficient generations. This was verified with networks of known communities. The convergence is faster when communities are well defined, i.e., modularity index is high. The proposed algorithm could also find a very near optimum solution, with correct number of clusters, in a small number of generations. A parallel implementation of the work, for quite large real world networks, is reported in [16]. In [16], it is also observed that, if the diameter of communities are constrained to a low value, the convergence of genetic search is faster. But, of course, the value of the diameter of communities vary depending on the characteristics of the network. The optimum maximum diameter of communities are different for different networks. A blind search, with different values of highest diameter, is computationally complex.

In this work, we proposed a multi-objective optimization approach, where in addition to maximizing the modularity by grouping the nodes in communities, we also give priority to minimize the diameter of large communities. The balance between these two optimization parameters is important. Detail of the algorithm is given in section 3.

To evaluate our algorithm with benchmark graphs [21], we created SWNs of different sizes with different numbers of clusters synthetically. In [18], a mathematical measure of the modularity of the community structure was proposed. It approaches the maximum value of 1, when the graph is very strongly clustered and zero when nodes are connected in random. One objective of the fitness function in genetic search is to search for the appropriate grouping of the nodes to maximize modularity index.

3. Proposed Multi-Objective Genetic Algorithm

The steps of the conventional genetic algorithm [22] are given below.

3.1 Algorithm Steps

The pseudocode of the algorithm is as follows:

(1) Generate initial population.
(2) Perform Cross-over on the selected chromosomes.
(3) Perform mutation on the selected genes.
(4) Fitness evaluation and tournament selection, from the new set of chromosomes to form population of the next generation.
(5) **if** (convergence == true).
(6) **end algorithm.**
(7) **Else.**
(8) Go to step 2.
(9) **End.**

First, we generate initial chromosomes. After uniform two-point crossover and mutation, finesses for the parent and crossed over chromosomes are evaluated. Tournament selection is used to form the next generation. Elite preservation is carried out to keep the best chromosome till convergence. We have two evaluation criteria. The result of clustering, as described by the best chromosome, is evaluated by checking the modularity index and the number of nodes erroneously clustered, i.e., nodes with more links outside its community compared to links to nodes of its own community.

3.2 Generation of Chromosomes for Initial Population

An example of a 10-nodes network is shown in Figure 1. Two possible chromosomes and the corresponding clustering of network nodes are shown in Figure 4 and Figure 5.

Node	1	2	3	4	5	6	7	8	9	10
Chromosome i	3	4	2	7	6	7	4	9	10	9

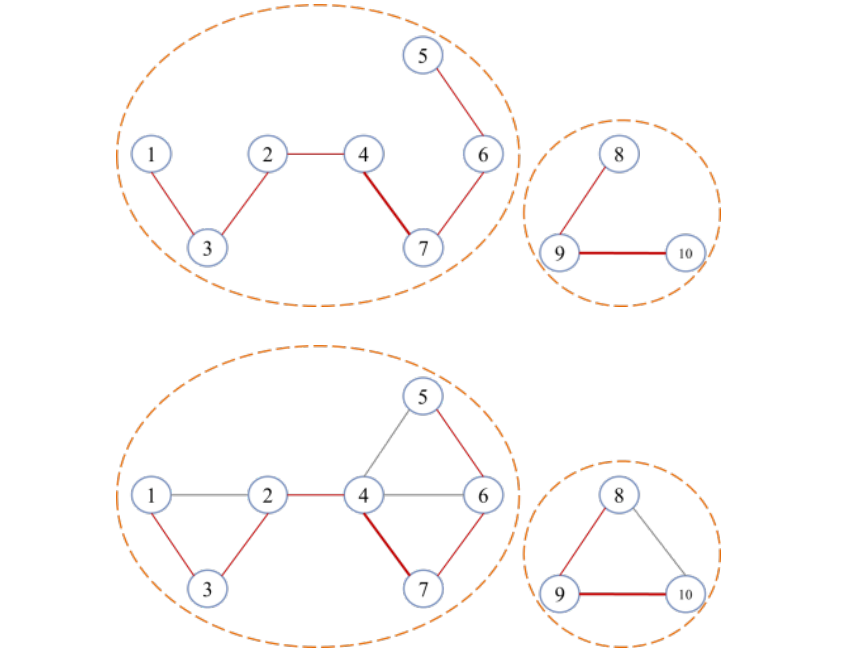

Figure 4. A sample chromosome leading to 2 clusters.

Node	1	2	3	4	5	6	7	8	9	10
Chromosome i	2	1	1	6	4	5	6	10	8	8

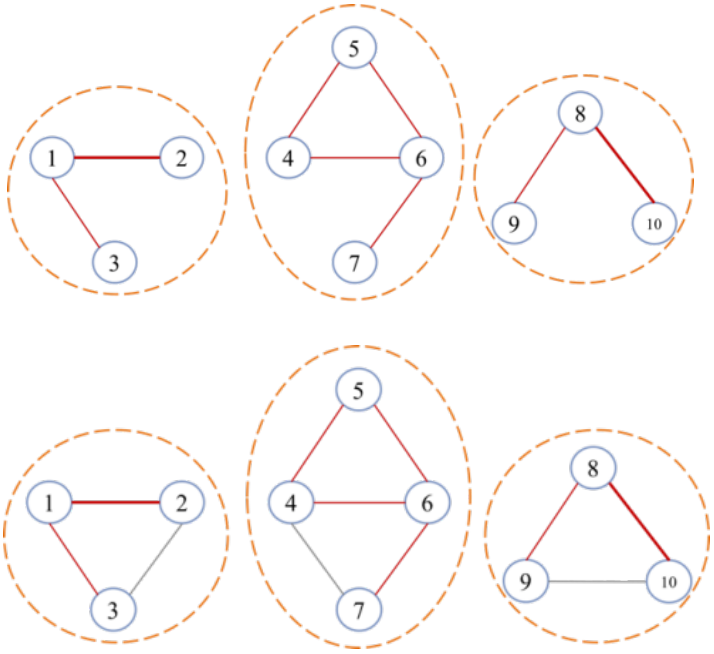

Figure 5. A sample chromosome leading to 3 clusters.

The length of a chromosome is the same as the number of nodes in the network. Here, we explain how different random chromosomes are created and how they are translated to divide the network into different community groups. We assume that nodes are labelled with ordinal numbers, from 1 to n, where n is in the number of nodes. In Figure 4, at the top the chromosome is shown. The first row is the serial number of nodes, from 1 to 10. For the second row, we randomly put one of the nodes that are connected to the node for which the label number is in the upper row. Thus, column 1 second row is filled with any of the node connected to node 1, i.e., node 2 or node 3. Here, node 3 is being randomly selected for entry in column 2. Similarly, for column 2 the second row entry could be any of the nodes 1, 3, or 4 which are all connected to node 2.

In this example, node 4 is selected. In the same manner, all entries from 1 to 10 are filled. Figure 5 shows another chromosome for the same network. From a chromosome how to define communities is explained in section 3.3.

3.3 Chromosome to Community

In this section, we explained how a chromosome is translated to communities within the network. First, we connect nodes in the upper row to the corresponding entries in the lower row. Thus, as in Figure 4, 1 is connected to 3, 2 to 4, 3 to 2, etc., whereby we get 2 disjoint graphs as shown below the chromosome. Within a connected group of nodes, we further add the links which were in the original network but no connected. Here, nodes 1 and 2 are connected, 4 is connected to node 5 and 6. In the other group, node 8 is connected to 9 and 10. Finally, we get 2 distinct groups of nodes as shown at the bottom in Figure 4. For the chromosome in Figure 5, we get three clusters of nodes as shown. Obviously, the numbers of nodes in different communities are balanced for the chromosome in Figure 5, but not balanced for the chromosome in Figure 4.

3.4 Fitness Function

We used multi-objective optimization, with two optimization criteria. One optimization criterion is the modularity of the resulting communities by evaluating Equation. (1) [18].

$$Q = \frac{1}{2E}\sum_{i,j}\left\{\left(A_{i,j} - \frac{k_i k_j}{2E}\right) \times \delta\left(C_i, C_j\right)\right\} \quad (1)$$

Q is the modularity calculated on the basis of communities defined by the chromosome. E is the number of links of the network. A is the adjacency matrix of the network, where $A_{i,j}$ is 1 when i^{th} node is connected to j^{th} node. Obviously, $A_{i,j}$ is symmetric. k_i represents the degree of node i, and C_i represents the class to which node i belongs to $\delta(C_i, C_j)$ returns 1 if node i and j belong to the same class, and 0 if not. According to [18], one can conclude that the network consists of clear communities. If Q is larger than 0.4. Less than 0.4, the connection between nodes are more random.

For a well clustered network, if the clustering assignments of nodes are correct, the value of Q goes high. One optimization criterion is to maximize the value of Q, so that nodes which are mutually connected are assigned to the same community. For the l^{th} chromosome, if the value of Q is Q_l, then one part of the fitness function is $(1 - Q_l)$. We want to maximize modularity, i.e., minimize $(1 - Q_l)$.

The other optimization criterion is the diameter of clusters. It was observed that if the diameter of clusters are restricted to some value, i.e., penalizing clusters with too large diameter, then the genetic algorithm search converges quickly. Of course, this threshold value of diameter is unknown, and depends on the configuration of the network. The maximum diameter of all communities, when the network nodes are correctly clustered, will be different for different networks. We do not know what will be the optimum highest diameter in advance. Our second optimization criterion is to keep the diameter of the clusters low. If for the l^{th} chromosome, d_l is the highest diameter among all its clusters, then the lower the value of d_l better is the fitness of the chromosome. Of course, a chromosome with very high Q_l will not be penalized for a little higher value of d_l. The overall fitness function of the l^{th} chromosome is defined as:

$$f_l = \left(1 - Q_l\right) + \alpha d_l \quad (2)$$

where, $\alpha = 1/(\tau D)$, where D is the largest diameter of all communities from all chromosomes. The coefficient τ tunes the priority between the modularity index Q_l and community diameter. We did various experiments to find proper value of τ. Overall, lower the value of f_l, higher is the probability of the l^{th}chromosome being selected to survive to the next generation.

3.5 Crossover and Mutation

Two chromosomes are first selected for crossover. A mask of length same as the chromosome is used. The mask is a contiguous array of 1's of arbitrary length and starting location, surrounded by 0's on both sides covering the remaining of the mask. At locations, in the sequence of node numbers, where the mask entry is 1, the entries in two chromosomes are swapped. Where the mask entry is 0, corresponding gene of the two chromosomes are kept unchanged. The crossover probability is set to different values, as explained in section 4.2.3. The population size was different for different networks, larger the network larger population size is necessary for early convergence. For mutation, an entry on the chromosome is randomly changed with probability 0.1%. During mutation, network connectivity information is used, so that mutations do not generate an invalid chromosome.

3.6 Ranking and Selection

Tournament selection method is used for selection. Throughout all generations, a fixed tournament size, of say γ, is used. From the pool of chromosomes, after crossover and mutation, γ number of chromosomes are randomly picked up, and their finesses are calculated. The chromosome with the lowest fitness is selected to survive to the next generation. The common strategy is to start with a low value of γ (say 2) for better exploration in the early generations, and slowly increase it, to say 4, when convergence approaches. In this experiment, we used γ=2 throughout all generations.

3.7 Convergence Rule

It is necessary to decide an appropriate termination condition. For an unknown network, the optimum clustering and the corresponding Q value is not known. The common convention for GA is to run the algorithm for a large number of generations and expect that the algorithm will converge to global optimum. The other possibility is to check whether the fitness is improving or not, for a few generations in a row. If it does not, the algorithm ends. We performed a set of preliminary experiments, with networks of known clusters, to see how it is converging. This is to find a cue to set the stopping criterion. For a real life network, the best option is to monitor the improvement of result with progressing generations, and check whether there is any improvement or not.

4. Preliminary Experiments

4.1 Data Creation

Before simulation, a set of benchmark networks are created for evaluation. The number of clusters and their member nodes are known and the Q values for the optimum clustering could be calculated. We created an example of 100-nodes network as shown in Figure 6. In this network, there are five clusters, and we know the member nodes belonging to each cluster. Searching for the best chromosome using genetic algorithm is performed, and compared with the known optimum result.

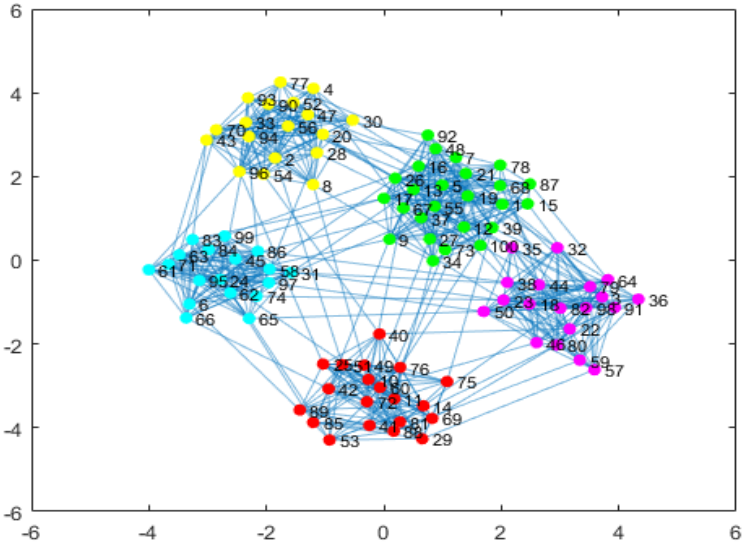

Figure 6. Example of a network with 100 nodes and 5 clusters. Different clusters are shown with different colors.

During simulation, we created many 100 nodes networks and run the proposed algorithm for clustering. The genetic search was terminated when the fitness reaches the known optimum value. This convergence criterion cannot be used for a network with unknown communities.

For GA searching, optimal parameter setting to get faster convergence is important. Depending on the problem complexity, suitable value for population size, cross-over and mutation probabilities will be different. Even, for a problem with the same number of nodes, depending on whether different clusters are clearly separated or partially overlapped, the complexity changes. A proper setting of the following parameters could lead to faster convergence:

(1) Number of generations - Larger the search space longer generations are needed.
(2) Population size - Larger the search space larger population size is needed.
(3) Crossover and mutation probability – higher value ensures better exploration but poor convergence.

Though, mutation is important for exploring the search space, as large mutation will disrupt high fitness chromosomes, the mutation rate must be kept low, usually lower than 0.1%. High mutation helps exploration of the search space during early generation.

Similarly, high rate of crossover, when convergence is approaching, will delay the search for best solution. We performed several runs of experiments, with changing value of crossover probabilities, which is reported in 4.2.3.

In addition to finding optimum parameter values, in this work we investigated whether using multi-objective search would lead to faster convergence or not. We run simulations with and without using multi-objective fitness to see the difference in performance, whether and how much the multi-objective GA improve the efficiency of search and quality of the result.

In addition, as explained in section 3.4, in selection part, the f_l function is used to evaluate the fitness of a chromosome. For an unknown problem, we do not know the best τ value. Thus, we experimented with τ value from 1 to 4, increasing in steps of 0.5. When τ value is set to 3.5, the optimum clustering is obtained in lowest number of generations, on an average. Therefore, $\tau = 3.5$ is used for all multi-objective GA simulations. When a single objective of modularity is used, α is set to 0.

4.2 Experiments with Varying Parameters

4.2.1 Experiment with and without Multi-Objective Optimization

Our first experiment was to verify whether multi-objective optimization could speed up the convergence or not. In the fitness function, we set the value of α=0, when we do not use the cluster diameter as part of the fitness function. When multi-objective GA is used, α is set to 3.5. The cross over rate was 90% and mutation rate 0.1%.

Table 1 and 2 show the results of the experiment, when GA is run for 100 generations. Conv. Gen means from which generation the fitness value of the chromosome does not improve any more. Table 2 shows the results of experiment executed 10 times. The optimum modularity index of the network is 0.6956. The network converged to global minimum, on an average in 45 generations. Table 1 shows the results when α= 0. The maximum value of fitness is 0.6956. On an average, the network converged in 62 generations. Out of 10 runs, in 3 runs it did not converge to global optimum. One or two error nodes remained. The average execution time for multi-objective GA was 20% less.

Table 1. Convergence with 100-nodes network when $\alpha = 0$. GA executed for 100 generations.

Expt.	Conv. Gen /100	Fitness	Error Node
1	78	0.6956	0
2	63	0.6956	0
3	38	0.6823	1
4	77	0.6956	0
5	46	0.6956	0
6	54	0.6956	0
7	45	0.6956	0
8	92	0.6678	2
9	80	0.6798	1
10	44	0.6956	0
Average	62	0.6899	0.4

Table 2. Convergence with 100-nodes network when α = 3.5. GA executed for 100 generations.

Expt.	Conv. Gen /100	Fitness	Error Node
1	47	0.6956	0
2	47	0.6956	0
3	62	0.6956	0
4	60	0.6956	0
5	51	0.6956	0
6	9	0.6956	0
7	54	0.6956	0
8	40	0.6956	0
9	40	0.6956	0
10	35	0.6956	0
Average	45	0.6956	0

The erroneous node is counted from the definition of clustering, mentioned in section 2. The number of clusters in all runs is the same. That is, we can always converge with the correct number of clusters. However, a few nodes were assigned to wrong clusters, when multi-objective GA is not used. Form results in Table 1 and Table 2, we can conclude that running the network with two objectives could get faster convergence with better quality of node clustering.

4.2.2 Experiments with Different Population Sizes

Our next experiment was to estimate the effect of population size on convergence. For a complex problem, a large population size is necessary to have convergence in reasonable number of generations. In this experiment, we run GA with different population sizes to find how the computation cost depends on this parameter, how the population size is related to the required number of generations for convergence. The small population size was set at 20, and the large at 100. Crossover and mutation were set at 90% and 0.1%.

Table 3. Convergence with 100-nodes network with two type of parameter settings. GA executed for 200 generation for Population Size = 20, and 100 generation for Population Size = 100.

Small Population Size = 20				**Large Population Size =100**			
Expt.	Conv. Gen /200	Fitness	Error Node	Expt.	Conv. Gen /100	Fitness	Error Node
1	80	0.6956	0	1	37	0.6956	0
2	157	0.6956	0	2	55	0.6956	0
3	124	0.6751	1	3	58	0.6956	0
4	100	0.6956	0	4	51	0.6956	0
5	89	0.6956	0	5	34	0.6956	0
6	89	0.6956	0	6	33	0.6956	0
7	176	0.6956	0	7	38	0.6956	0
8	156	0.5945	2	8	25	0.6956	0
9	173	0.6827	1	9	17	0.6956	0
10	90	0.6956	0	10	50	0.6956	0
Average	123	0.6822	0.4	Average	40	0.6956	0

Table 3 shows the result when the population size is set to 20 (small population) and 100 (large population). With population size of 20, the network converged, on an average, in 123 generations. But, it could not converge to the optimum solution 3 times, out of 10 trails. The algorithm converged with error nodes. With population size of 100, the network converged on an average in 40 generations. The average run time was about 30% longer. Though it took a little longer time, the algorithm always converged in optimum solution.

4.2.3 Experiment on Crossover Rate

In our previous experiments the crossover rate was fixed at 0.9. High crossover is good for exploration. But, at later generations, the good chromosomes will be disrupted due to high probability of crossover. In this experiment, we reduce the crossover rate from 0.9 to 0.1 in steps, as the generation progresses. Table 4 shows the result of convergence with decreasing crossover rate. In this simulation, the population size is 100 and mutation rate is fixed at 0.1%. Though the final results are similar to Table 3 (fixed crossover rate), the convergence is 35% faster.

Table 4. Convergence with gradually decreasing crossover rate.

Expt.	Conv. Gen /100	Fitness	Error Node
1	20	0.6956	0
2	16	0.6956	0
3	34	0.6956	0
4	35	0.6956	0
5	35	0.6956	0
6	37	0.6956	0
7	46	0.6956	0
8	23	0.6956	0
9	20	0.6956	0
10	47	0.6956	0
Average	31	0.6956	0

4.2.4 Experiment on Real-World Data

In this experiment, a real-world data is used. The data is available on the internet website, known as Football network: http://networkdata.ics.uci.edu/data/football/. The network contains 12 communities. The adjacency matrix and the community information is available, for which the actual modularity index can be calculated using Equation 1. The Q value turns out to be 0.54. Our algorithm searches to find the maximum fitness value. Figure 7 shows the actual structure of football network with 12 communities. Different communities are shown in different colors.

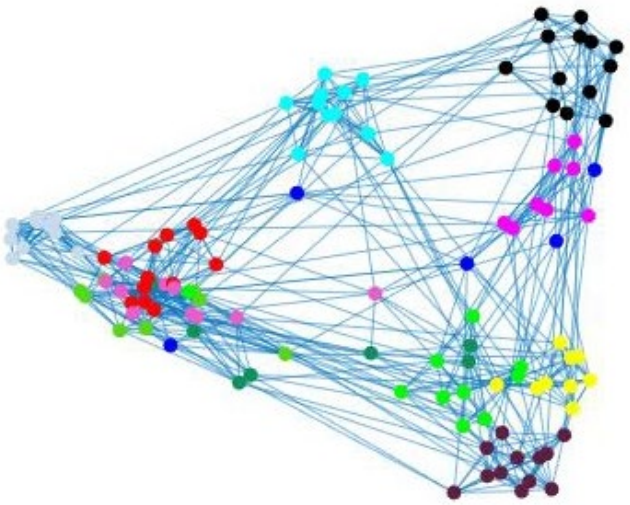

Figure 7. Structure of football network with 12 communities, as constructed from actual data.

As we can see in Figure 7, there are overlapping communities. Merging closely connected communities of the actual data would lead to better modularity. The detail of GA parameters, used in this experiment are shown in Table 5. From Table 4, we know that starting with a high probability of crossover and mutation is good for exploration and could reach convergence faster. But, at later generations, slowly reducing the crossover and mutation rate achieves efficient convergence. In this simulation, the GA is run for 1000 generations. The population size is set to 100. The probability of crossover and mutation are reduced every 100 generations. We reduce the crossover rate from 0.9 to 0.1, and the mutation rate is reduced from 0.4 to 0.

Table 5. The initial parameter of efficiency Genetic Algorithm.

Number of Generations	1,000
Population Size	100
Crossover Rate	Initial value 0.9, reduced to 0.1.
Mutation Rate	Initial value 0.4, reduced to 0.
τ	2

Figure 8 to Figure 10 show the results of this experiment. Proposed algorithm is executed 10 times, and each time for 1,000 generations.

Figure 8 shows the number of clusters at the end of every 100 generations. The network always converged in around 800 generations or less.

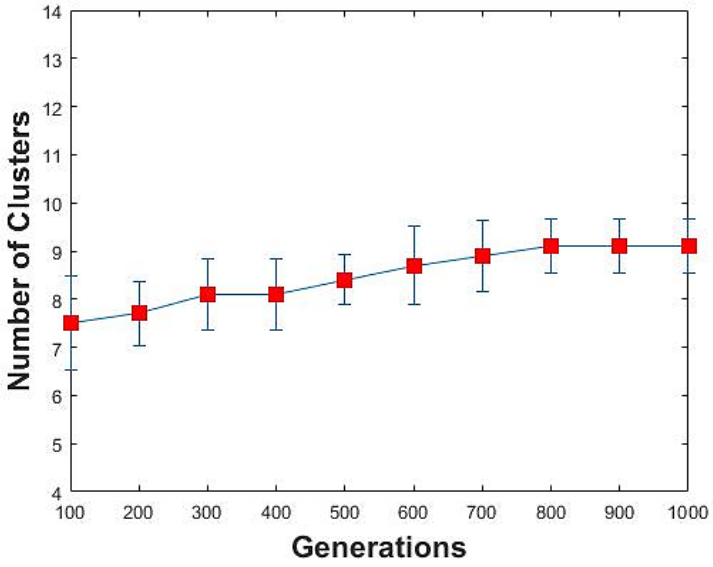

Figure 8. Number clusters at the end of every 100 generations.

There are two aspects to evaluate the results, as defined in section 2: (1) The number of error nodes, (2) Comparison with the original network structure. In this experiment, we used these two criteria to evaluate our proposed algorithm. In Figure 9, the result shows that some error nodes remain. This is because the original network is not having clear communities, and a few nodes are having more links with nodes outside their own community. In fact, the proposed algorithm could minimize error node as defined in section 2. In addition, the result is stable.

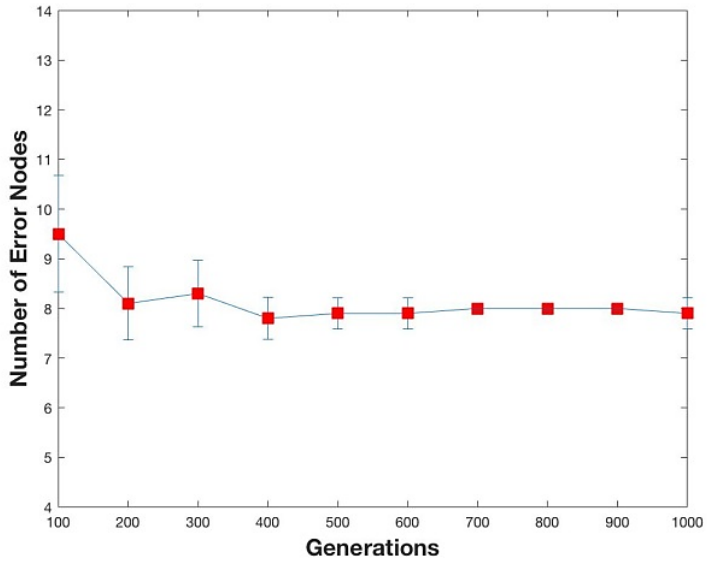

Figure 9. Number error nodes at the end of every 100 generations.

Figure 10 shows the result of modularity value. The modularity tends to stabilize after 600 generations, and the fitness value is higher than what is calculated from the original data. The membership relation of the original data does not form the best possible communities as defined by Equation 1.

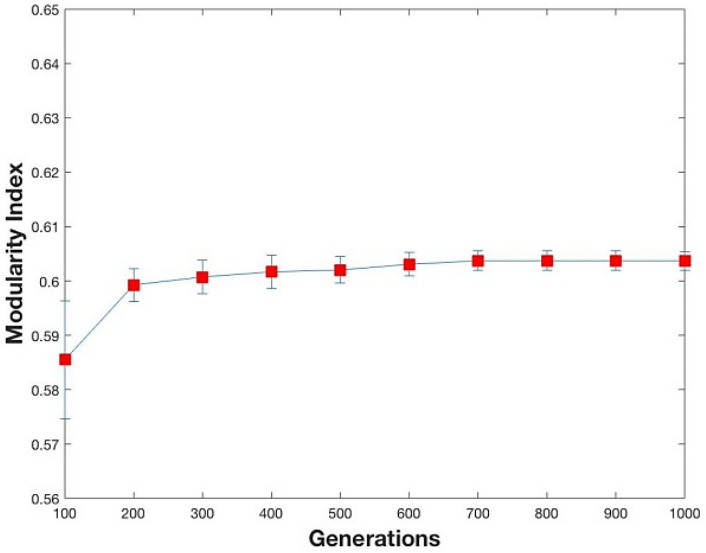

Figure 10. Modularity index at the end of every 100 generations.

4.2.5 Comparison with Louvain Algorithm

Louvain algorithm [9] is an efficient heuristic hierarchical clustering algorithm. It is bottom up algorithm, and converges fast. We want to compare performances of the proposed algorithm with Louvain algorithm. In this simulation, we execute proposed algorithm and Louvain algorithm 10 times. Comparison of their performances for different runs are shown in Figure 11 and Figure 12.

The fitness values for different runs are shown in Figure 11. We can observe that the proposed algorithm could achieve higher modularity index values than Louvain algorithm. In addition, results for the proposed algorithm is stable for different runs, whereas Louvain algorithm gives fluctuating results, sometimes converge to quite low modularity index.

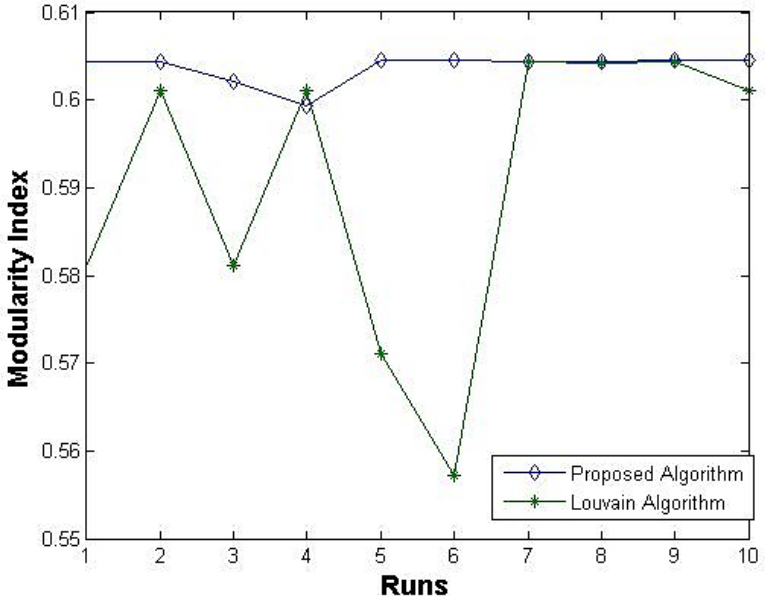

Figure 11. Comparison of modularity performances of the proposed algorithm and Louvain algorithm.

In Figure 12 we show the number of error nodes. The error nodes are calculated with respect to the original data. In Louvain algorithm, the number of error nodes is quite high around 20. In our algorithm, the number of error nodes is always 8. In addition, performance of Louvain largely fluctuates over different runs.

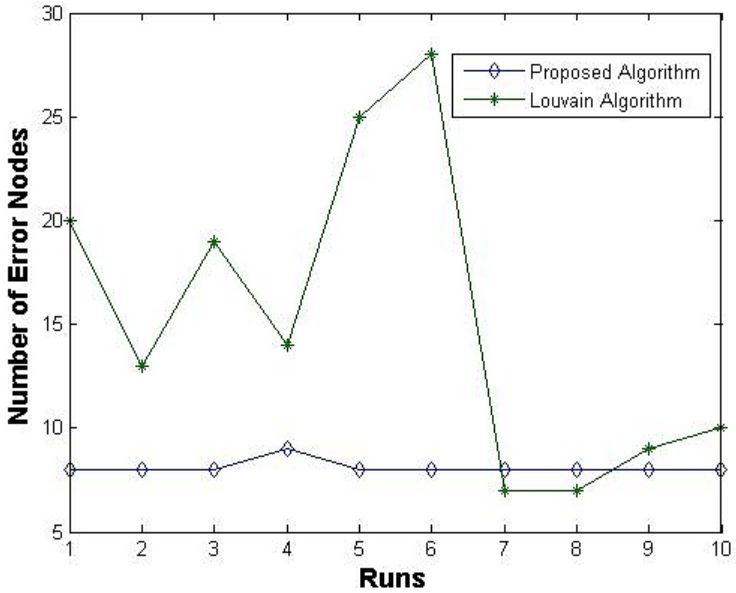

Figure 12. Comparison of the proposed algorithm and Louvain algorithm.

5. Conclusions

In this paper, we proposed a multi-objective genetic algorithm for community detection in a network. Two optimization criteria were used, the modularity index of the community assignment of the nodes, and the highest diameter of the communities. We proposed a fitness function, which is a combination of both criteria. As the optimum diameter of the communities are not known, and is different for different networks, it is difficult to balance the influence of the two optimization criteria. We used a term τ, which controls the influence of diameter on the overall chromosome fitness value. In this work, we fixed the value of τ after a few experimentations. Larger τ is suitable, when the largest diameter of the optimum cluster (D) is large. In SWN, as D is not correlated with the number of nodes of the network (n), we cannot estimate proper τ from n. Adjusting τ dynamically would be a better approach, on which we are working.

Even though the number of generations needed in this proposed algorithm is less compared to our previous work [20], in this work we need an added computation of diameter of clusters defined by different chromosomes. Computation complexity for finding diameter of a graph G(V, E) is O(V×E), where V is the number of vertices and E the number of Edges. Even for a sparse graph, when V is large, the computation cost is also high. We are working on modification of the algorithm to compute diameter only at intervals of generation to reduce computation cost.

We are doing a parallel implementation using GPU. This will enable us to run the algorithm for large networks in reasonable time. Value of the constant τ is also an important factor to influence the network convergence. Dynamically adjusting τ would improve performance.

Ant Colony Optimization (ACO) based algorithm is another popular optimization algorithm. Compared to GA, ACO can discover good solutions rapidly and it is suitable for discrete problems like graph clustering. But, ACO has a tendency to converge to local minimum. We will experiment with ACO for graph community search algorithm, including ways to avoid local minimum.

Acknowledgment

Part of this research is funded by iMOS research fund, research promotion wing of Iwate Prefectural University, Japan and Sendai Foundation of Applied Information Sciences, Japan.

References

[1] Brin, S. and Pag, L. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30: 107-117.
[2] Yutaka, M. 2003. Prediction and Discovering Small World Network. *Society of Artificial Intelligence*, 18, 3.
[3] Yutaka, M. 2005. Network Structure and its Emergence. *Proceedings of AAAI*, 11-14.
[4] Newman, M. E. J. 2010. Networks: An Introduction, Oxford.
[5] Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H. and Laurienti, P. J. 2011. The Ubiquity of Small-World Networks. *Brain Connect*, 1, 5: 67-375.
[6] Vincent, L., Clrot, F. and Creff, N. 2015. K-means Clustering on a Classifier induced Representation Space: Application to Customer Contact Personalization. *Real World Data Mining Applications Springer*, 139-153.
[7] Yuasa, M. and Hayama, S. 2012. Node Classification Method for Complex Network and its Application. *Society of Artificial Intelligence*, 111-120.
[8] Leicht, E. A., Holme, P. and Newman, M. E. J. 2006. Vertex similarity in networks. *Physical Review*, 73, 026120.
[9] Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. 2008. Fast Unfolding of Communities in Large Networks. *Statistical Mechanics: Theory and Experiment*.
[10] Chakraborty, G., Sato, N. 2017. A Reliable Graph Clustering Method Using Genetic Algorithm. *International Symposium on Nonlinear Theory and Applications*, Cancun, Mexico.
[11] Erd″os, P. and R´enyi, A. 1959. On Random Graphs. *Publicationes Mathematicae*, 6: 290-297.
[12] Watts, D. J. and Strogatz, S. H. 1998. Collective Dynamics of 'Small-World' Networks. *Nature*, 393.
[13] Barab´, A., Albert, L. A. and Albert, R. 1999. Emergence of Scaling in Random Networks. *Science*, 286, 5439: 509-512.
[14] Schaeffer. S. E. 2007. Graph Clustering. *Computer Science Review*, 1, 1: 27-64.
[15] Mitchell, M. 1998. *An Introduction to Genetic Algorithms*, MIT press, ISBN: 0262631857.
[16] Ghosal, A. K., Das, N., Bhattacharjee, S. and Chakraborty, Goutam. 2019. A Fast Parallel Genetic Algorithm Based Approach for Community Detection in Large Networks. *Conference on Communication Systems and Networks*, Bengaluru, India.
[17] Palla, G., Derenyi, I., Farkas, I. and Vicsek, T. 2005. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. *Nature*, 435: 814-818.
[18] Clauset, A., Newman, M. E. J. and Moore, C. 2004. Finding Community Structure in Very Large Networks. *Physical Review*, 066111.
[19] Hafez, A. I., Ghali, N. I., Hassanien, A. E. and Fahmy, A. A. 2012. Genetic Algorithms for Community Detection in Social Networks. *Conference on Intelligent Systems Design and Applications*, 460-465.
[20] Shang, R., Bai, J., Jiao, L. and Jin, C. 2013. Community Detection based on Modularity and An Improved Genetic Algorithm. *Physica A: Statistical Mechanics and its Applications*, 392, 5: 1215-1231.
[21] Lancichinetti, A., Fortunato, S. and Radicchi, F. 2008. Benchmark Graphs for Testing Community Detection Algorithms. *Physical Review*, 78, 046110.
[22] Mitchell, M. 1998. An Introduction to Genetic Algorithms. *MIT press*, ISBN: 0262631857.