International Journal of Applied Science and Engineering

Special issue: The 10th International Conference on Awareness Science and Technology (iCAST 2019)

A two stage converging genetic algorithm for graph clustering

Yu-Ching Lu[1], Goutam Chakraborty[1,2*], Masafumi MatsuHara[1]

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

ABSTRACT

Graph clustering is a classical problem, and is proven to be NP-complete. It is at the core of many useful algorithms, like Network and VLSI design, computer graphics, data mining etc. In recent years, with exponential increase in the use of social network and strong incentive for creating applications exploiting the information hidden in these networks, clustering of large graphs (social networks) has become an important research topic. As the problem is NP-complete, various heuristic algorithms are proposed to find near optimal solutions efficiently. Optimization criteria are defined depending on the applications. Two important criteria for all heuristic algorithms are quality of the result and its stability over different runs on the same problem. In this work, we proposed a two stage genetic algorithm for network clustering. Modularity index for the partitioned graph is the criterion to optimize. By experimenting with several real-life networks, we have shown that our algorithm is stable and delivers a high modularity partitioning compared to other competitive heuristic algorithms. The stability of the algorithm is analyzed through simulations.

Keywords: Graph clustering, Social network analysis, Multi-objective optimization, Genetic algorithm.

OPEN ACCESS

Received: April 25, 2020
Revised: July 6, 2020
Accepted: August 27, 2020

Corresponding Author:
Goutam Chakraborty
goutam@iwate-pu.ac.jp

Publisher:
Chaoyang University of Technology
ISSN: 1727-2394 (Print)
ISSN: 1727-7841 (Online)

1. INTRODUCTION

In a random graph, the probability of any pair of nodes being connected are the same, or it follows some probability distribution. In this paper, we will use the word graph and network interchangeably, meaning the same thing. There are yet other classes of networks as follows:

1. Scale Free Networks (SFN), in which there are a few preferential nodes with higher degrees of connections. Thus, the degree distribution is exponential, with most of the nodes having very low degree. In scale-free networks (SFN), distance between two randomly selected nodes is proportional to the logarithm of the number of nodes. This happens due to preferential connection. Small World Networks form communities, where nodes within a community are strongly interlinked with short mutual distance. Two communities are often connected with a few links.

2. There are graphs consisting of dense sub-graphs, and those sub-graphs are sparsely connected. Those sub-graphs could be nested, and/or overlapping.

The above two types of graphs could be partitioned into clusters of nodes, or communities. Graph clustering is the problem to identify the dense sub-graphs, where the elements of the adjacency matrix are non-negative real numbers.

Graph partitioning problem is a classical problem (Kernighan and Lin, 1970; Andreev and Räcke, 2006) and is proved to be NP-complete (Šíma and Schaeffer, 2006). It is therefore computationally hard to find the global optimum in case the network consists of a large number of nodes. Yet, with the exponential increase in the volume of data

generation and storing, popularity of social networks, online shopping of goods and services, the need for handling networks with large number of nodes is getting more and more important.

Clustering is at the core of solving various important problems, related to analysis and exploration of data. In the area of VLSI design (Karypis et al., 1970), parallel computing, Software engineering, image segmentation (Shi and Malik, 2000), graph clustering is used. In recent years, applications have opened up of intuitively clustering of similar samples in a data set by mapping the data to a network, where elements of the adjacency matrix are the similarity between 2 data elements. Clustering of the network leads to discovering natural groups of similar elements in the data set. Thus, an important variant of data clustering is graph clustering where similarity is expressed by links on a graph. The network of data would reveal hidden structure of data after clustering. Similar data units are in the same cluster. The centrality of the cluster of nodes would reveal the prototype of the group. As the similarity between data elements could be any real number, the elements of the adjacency matrix is not just 0 or 1, but a real. The algorithm proposed in this paper could handle such links, though all the experiments are restricted to unweighted un-directed graphs.

The other important application area is the type of data that evolves. This type of data originates from social networks, or purchase records of customers from e-Commerce Portals.

Several graph clustering algorithms, based on graph cliques and cuts were proposed. Those algorithms are too complex for very large graphs. Algorithms like minimum normalized cut is proved to be NP-complete (Dahlhaus et al., 1992). In recent years, several heuristic algorithms were presented (Blondel et al., 2008; Dinh et al., 2015; Aloise et al., 2010; Newman, 2006). These algorithms optimize various fitness functions that measure the quality of a cluster which partitions the graph. Examples of such clustering measures include modularity index, conductance, local and relative densities, etc. Thus, a cluster is a connected subgraph induced by a subset of vertices and edges, where most of the links from the members of a cluster are internal edges within the group and a few edges to vertices outside the group. The clustering algorithm searches for proper partitioning of the network nodes to optimize one or more of the above mentioned fitness measures.

Several graph partitioning algorithms are proposed in recent years. All algorithms have their respective optimization criterion. Heuristic algorithms, starting with different random initialization and probabilistic search decision as it progresses, deliver different results on different runs. An example is GA based graph clustering (Chakraborty and Sato, 2017; Lu and Chakraborty, 2019; Ghosal et al., 2019), where sometimes the convergences are at different local minimums, giving different results in different runs. If the network has clear clusters, i.e., the modularity index is high, there is only a deep global minimum and the algorithm always converges there. If the modularity of the network is low, it would converge at different shallow minimums, with different clustering results.

The algorithm proposed in this work is based on GA, and works in two stages. During the first stage, the optimization criterion is Q, the modularity index (Newman, 2010). When it converges, we have an estimate of the Q value of the network, and we know the cardinalities of different clusters. We add component in the fitness function to balance the clusters (Dahlhaus et al., 1992). Based on variance of sizes, we upgrade our search criterion, to balance it. The fitness function is changed to multi-modal. In different experiments, the proposed algorithm showed better results with improved Q value and more balanced clusters. The results were compared with popular heuristics by Louvain (Blondel et al., 2008), Newman (Newman, 2006), and Aloise (Aloise et al., 2010).

The rest of the paper is as follows. Related works and the proposed algorithm in the perspective of related works, is discussed in section 2. In section 3, we explain the proposed two stage genetic algorithm. In section 4, we describe the data used and experimental results. The conclusion and future works are in section 5.

2. PROBLEM DEFINITION AND RELATED WORKS

Research in network science was initiated by Erdős and Rényi (1959), by proposing different random network models and their properties. Around the end of 20th. century, a special type of random network, called small-world network (Matsuo, 2003; Matsuo, 2005; Telesford et al., 2011; Watts and Strogatz, 1998; Barabási et al., 1999) attracted much attention, because of its ubiquity in many naturally evolving networks. These networks often form closely knitted communities. Finding those communities efficiently, for large networks, became an attractive research topic for many practical applications. A good survey of graph clustering is available at Schaeffer (2007).

The original graph partitioning algorithms were 2-way partitioning, or bi-partitioning. Graph cut (minimizing the connecting edges of two groups), or quotient cuts (conductance, where two partitions are balanced) were the optimization criterion. Spectral graph clustering, using graph Laplacian are a set of algorithms pioneered by Fieldler (Fiedler, 1975), and followed by subsequent works (Pothenf et al., 1990). In general, graphs will form multiple clusters, and multiway partitioning is more relevant for real-life networks. Depending on the application, the partition could be motivated towards finding balanced sub-sets of nodes, or communities with tight connections. Modularity index value (Q) of a graph is defined in Clauset et al. (2004), as shown in Equation (1), in section 3.4, is considered as the optimization criterion for community detection in many works following Clauset et al. (2004). Higher modularity

index means nodes within the same group have higher probability of being connected. Blondel proposed a fast heuristic algorithm (Blondel et al., 2008), popularly known as "The Louvain Method" (from the name of the author's city and university). It starts with every node assigned as its own community. For every node, evaluate modularity gain by removing it from its present community, and place it to its neighbor if there is a gain. It is repeated until there is no more gain (local maximum).

Another approach, proposed by Pascal Pons and Matthieu Latapy (Pons and Latapy, 2006), using random walks to compute similarity measure between nodes. This is also known as the "Walktrap" algorithm. Distance between nodes are calculated using probabilities for getting from node i to node j in t random walk steps. The algorithm is useful for factorization of very sparse matrix, e.g., rating matrix, for applications like recommendation system (Fouss et al., 2007). Once the weighted network is constructed, the optimization objective for clustering could be minimizing the error nodes, where an error node is one for which the number of links to nodes in the same group is less than the number of links going out to another partition.

Recently, there are several works for community detection using genetic algorithm (Hafez et al., 2012; Shang et al., 2013; Li et al., 2013; Song et al., 2012), including those proposed by us (Chakraborty, and Sato, 2017; Lu and Chakraborty, 2019; Ghosal et al., 2019). In our previous report in 2017 (Chakraborty and Sato, 2017), the main contribution was that it could always find the optimum number of clusters and node assignments after sufficient generations, but only when the modularity index of the graph is high. In Lu and Chakraborty (2019), a multi-modal GA was proposed where the diameter of the clusters was chosen as another optimization parameter. In some graph instances, it was more efficient compared to only modularity considered as the optimization criterion.

In our previous work (Lu and Chakraborty, 2019), we upgraded the fitness function. In addition to modularity index, we added another component to balance the clusters. We used a coefficient α for the node balancing term.

The community structure of network is broadly classified into 3 categories: (1) communities of nodes with more links within communities than between communities, (2) Overlapping communities, (3) Hierarchical communities. Different community detection algorithms are suitable to different graph structures. Hierarchical graph partitioning is proposed in Rossi et al. (2019).

2.1 Background of the Work

Before introducing our proposed algorithm, to emphasize the problems with clustering various types of graphs, here we introduce 3 different graph structures. A node is defined to belong to a community if the number of inward links to other members of the community is more than outward links to nodes outside the community.

Overlapping communities are as shown in Fig. 1, where the membership of node 13 and 14 are in two groups. One can coalesce overlapping clusters, which will result in poor balancing. Depending on application, it may be important to identify overlapping clusters. Gergely et al. (2005) used k-clique percolation technique to efficiently grouping the network, and separately identify the two clusters.

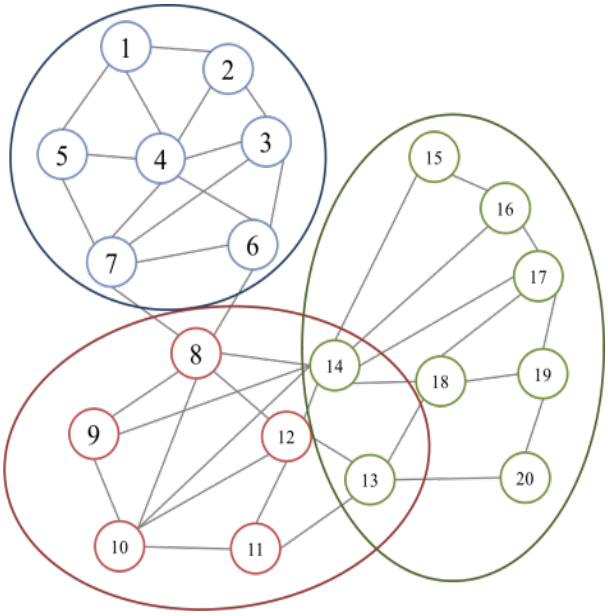

Fig. 1. Example of overlapping communities.

Hierarchical clustering of nodes is shown in Fig. 2. With higher resolution smaller clusters are revealed. This structure prevails in many real-world networks, like internet or wide area networks, power grid, etc. The Louvain Method (Blondel et al., 2008) for community detection is an efficient way to find hierarchical clustering. In phase 1, it performs local clustering. In phase 2 communities are merged recursively to "super nodes'.

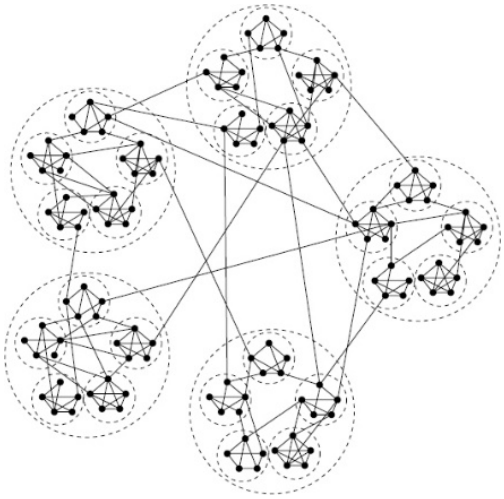

Fig. 2. Example of hierarchical communities.

Recently, many works for community detection tested their algorithm performance on real-world networks (Agarwal and Kempe, 2008; Dinh et al., 2015; Aloise et al., 2010). They try to identify community structure with high modularity index. For an unknown problem, it is the modularity index as well as the number of communities are unknown. For a real-world network there are real communities, but not known. The target is to discover similar structure as the original. How good is modularity index a measure of success is an interesting issue.

In this work, GA is used for community detection. For GA search the termination condition is an important issue. In this work, we run the algorithm for a large number of generations, and convergence occurred before the preassigned number of generations are over. It is possible to stop genetic search, when the fitness is not improving over a certain number of generations. The detail of simulations and results are explained in section 4.

3. PROPOSED MULTI-OBJECTIVE GENETIC ALGORITHM

The proposed algorithm is performed in two stages. The optimization objective changes from stage 1 and stage 2. The genetic operations, selection, crossover, mutations, are as in a conventional genetic algorithm, similar to our previous work (Chakraborty and Sato, 2017). The pseudocode is as follow:

1. Generate Initial population
2. Select subset of chromosome based on cross-over rate. 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**

3.1 An Algorithm Development Process

Step 1: Create random chromosomes as initial population. The chromosome length is the same as the number nodes of the network.

Step 2: After the crossover and mutation steps, the parents and off-spring chromosomes are evaluated by the fitness function.

Step 3: Population for the next generation is selected by tournament selection. Meanwhile, the Elite preservation would store the best chromosome till that generation.

Step 4: GA search will continue until it reaches termination criteria. After convergence, we have the best chromosomes for all population throughout the entire generations. The best chromosome is the GA search result, and will represent the clustering result.

The best chromosome is evaluated by checking the modularity index only. We do not consider to minimize the number of error nodes, i.e., nodes with more links outside its community compared to links to nodes of its own community, in the objective function. Balancing nodes in different communities is also considered in the fitness function, as explained below.

3.2 Creation of the Initial Population

A chromosome is defined as the edge list of the network. The length of a chromosome is the same as the number nodes of the network. Here, we explain how different chromosomes are created, and how they define different clustering of the network. Let us assume that there are n nodes in the network, and we labelled them with the ordinal numbers, from 1 to n. For explanation, we use a 10 nodes network example, which is shown in Fig. 3. We purposely used two colors for two groups of nodes, to explicitly show two communities, a possible partition.

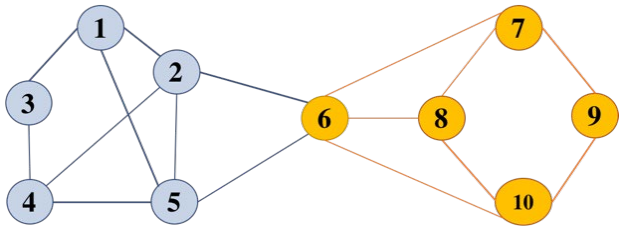

Fig. 3. Example of 10-nodes network.

While the chromosome is created by a random permutation of n, it will represent different partitions of the network.

Table 1 and Table 2 show two random permutations (the second rows of the tables) and corresponding partitions of the network. By chance, both chromosomes created 2 partitions. The first row represents the serial number of nodes, from 1 to 10. The second row, we randomly put one of the nodes based on the original network links. In original structure, node 1 has connection with node 2, 3, and 5. Here, second row entry is filled with any of the node 2, 3, or 5. In Table 1, node 2 is selected. In the same way, all entries from 1 to 10 are filled. The edges from row 1 entries is connected to the corresponding row 2 entries, and the generated network is illustrated. Similarly, from chromosome of Table 2, we form the network of Fig. 5. How the two partitions are completed, is explained in section 3.3.

Table 1. Connection list of the network with unbalanced clusters.

Nodes	1	2	3	4	5	6	7	8	9	10
Chromosome	2	5	1	3	6	2	8	10	7	9

Table 2. Connection list of the network with balanced clusters.

Nodes	1	2	3	4	5	6	7	8	9	10
Chromosome	2	5	1	3	4	7	8	10	7	6

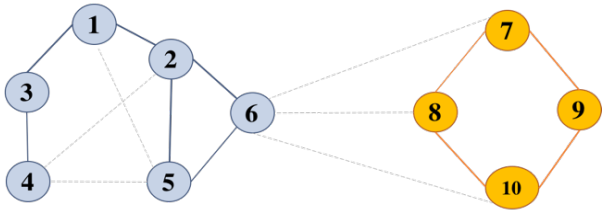

Fig. 4. An unbalanced clustering of a 10-nodes network, clustered in two communities.

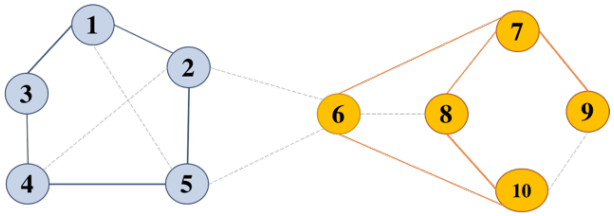

Fig. 5. A balanced clustering of a 10-nodes network, clustered in two communities.

3.3 From Chromosome to Cluster

In this section, we explain how chromosomes are translated to communities within the network, from what we had in Fig. 4 and Fig. 5. As in Fig. 4, after all the connections defined in the chromosome, e.g., node #1 is connected to node #2, node #2 is connected to node #5, node #3 is connected to node #1, etc., we end up with two disjoints groups. Within a connected group of nodes, we further add the links which were in the original network but not connected based on the chromosome. We then get 2 subgraphs of the original graph. Both chromosomes create 2 groups of nodes, Fig. 5 is more balanced compared to Fig. 4, and in Fig. 4 node #6 is an error node. For these reasons, we can claim that Fig. 5 is a better clustering, and chromosome 2 will have higher fitness value.

3.4 Fitness Evaluation

Most of the recent graph clustering algorithms (Newman, 2010) use Newman's modularity index (Q) as the metric to maximize for detecting communities. In our work, two optimization criteria are used. One optimization criterion is (Q) of the resulting communities calculated by evaluating Equation (1).

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

Where, Q is the modularity index, E is the number of edges of the network. A is the adjacency matrix, where A_{ij} is the i^{th} row and j^{th} column element. If node i and node j are connected, A_{ij} returns 1 and otherwise 0. k_{ij} is the degree of node i, and C_i is the community to which node i belongs, as a result of the partitioning. If node i, and node j belongs to the same community $\delta(C_i, C_j)$ returns 1, and otherwise 0.

A graph has an optimum value of Q^*, when the clusters are optimally partitioned. Depending on the nature of connection between nodes, Q^* could be high or low. When Q^* is high, it is easy to get the best community partition, even with heuristic or random search algorithms like GA. Most of the time, the global optimal could be found. If Q^* is low, there is higher probability for the algorithm to converge to local minimum, failing to achieve the best clustering results.

As the Q value is higher for a chromosome, during genetic search (in the search space of $\prod_1^n k_i$), it means the chromosome represents a better community structure. Therefore, one optimization criterion is to maximize Q_i, or minimize $(1-Q_i)$, for the i^{th} chromosome. $(1-Q_i)$ ranges between 0 to 1. The second criteria for optimization is balancing the number of nodes in different communities. Though with a less priority compared to modularity index, we direct the search to balance the nodes in each community. We use a term representing variance of cardinalities of different subsets of network partitions. Fitness function will minimize that.

The aim of genetic search is to minimize the fitness function (f_i) of the i^{th} chromosome. The search is done in two stages. In the first stage, the $f_i = (1-Q_i)$. Once the genetic search converges, we get a community structure based on every chromosome. Suppose, we have K communities, and the number of nodes in k^{th} community is n_k. We calculate the term σ_i (representing variance of node counts in different clusters) for the i^{th} chromosome as:

$$\sigma_i = \frac{\sum_{k=1}^{K}|n_k - \bar{n}|}{n} \quad (2)$$

Where, n is the total number of nodes in the graph, the average number of nodes per cluster is $\bar{n} = \frac{n}{K}$, and $0 \leq \sigma_i \leq 1$.

Once stage 1 genetic search converges, the generations continue with a different fitness function as defined in Equation (3).

$$f_i = (1 - Q_i) + \alpha \times \sigma_i \quad (3)$$

The coefficient α tunes the priority between the modularity Q_i of the communities and the variance of the number of nodes in different communities. We did various experiments to find proper σ value.

3.5 Crossover and Mutation

A subset of chromosomes, called cross-over pool, are selected. The cardinality of cross-over pool depends on the

Crossover probability. If it is set to 90%, 90% of the chromosomes from the population is randomly selected for the crossover pool. Two chromosomes from the pool are then randomly selected for crossover. We used two-point crossover. The middle part of the two points, from two chromosomes, are swapped. As, every gene is a valid entry, we do not need any validity test after crossover. In addition, for better exploration of the search space and to avoid local minimum, we use mutation. Because, mutation disrupts a chromosome, we kept mutation probability to be low. We randomly replace some of the genes on the chromosome, with another valid node number, using network adjacency matrix, so that mutation does not generate an invalid chromosome.

3.6 Tournament Selection

Tournament selection, with tournament size 2, is used for selecting an individual from the population. 2 chromosomes, from the pool of parents and off-springs (after crossover and mutation) are selected randomly. The one with lower f_i (better chromosome) goes to the population of the next generation. The "Selective pressure" is low with tournament size 2. In general, during the beginning of genetic search, tournament size of 2 is good for better exploration, but while nearing convergence, the tournament size needs to be increased. In this work, we kept it same during the whole generations.

3.7 Convergence Rule

Defining an appropriate termination condition is difficult. If we terminate early, the resulting solution could be a shallow local minimum. If we run for many generations, it will be inefficient. For an unknown network, the optimum clustering and the corresponding Q value is unknown. The optimum solution will require computation of the order of d^n, where d is the average degree and n is the number of nodes.

The common convention is to run genetic search 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 pre-assigned few generations in a row. If it does not, the algorithm ends. In our simulation, we run the algorithm for a fixed number of generations. We monitor the fitness value for convergence and stepwise update the value of α to attain better modularity. Further explanation is available in section 4.3.

4. EXPERIMENTAL RESULTS AND COMPARISON

We did simulation experiments on different networks, to evaluate the performance of the proposed algorithm, and ascertain the appropriate values of parameters so that the algorithm converges to a good optimal solution efficiently. We also investigate how appropriate values of the parameters depend on network size, its optimum modularity index, etc. We start with a preliminary experiment of small network with 10 nodes created manually, as shown in Fig. 6. Simulations were performed on a computer with Intel i7-3930 K CPU @3.20 GHZ and 16 GB RAM. Simulation programs were written in MATLAB, and codes not optimized.

4.1 Preliminary Experiment

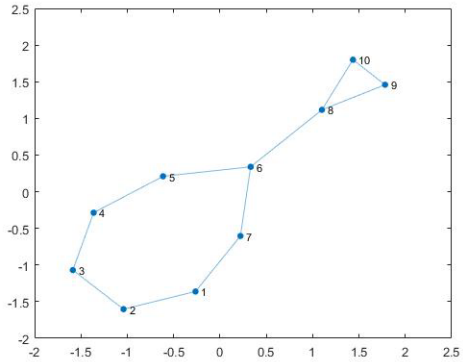

Fig. 6. Example of 10-nodes network. There is no optimal clustering defined here.

The nodes in Fig. 6 would be partitioned in different groups, and the number of clusters the network is partitioned would be different, depending on the fitness function. The obtained modularity index value will be different too. For example, when the network is divided over the link between the nodes 6 and 8, we have minimum cut, but the partitions are not balanced, with one group much larger. If minimum cut is the optimization criterion, this division will be acceptable. Else, if inter-cluster conductance or balanced partition are the fitness goals, this result is not satisfactory.

Table 3. Parameters values for Stage 1 and Stage 2 of genetic search.

Stage →	Stage (1)	Stage (2)
Alpha	0	0.01
# of generations	10	10
Population size	10	10
Crossover rate	0.2	0.2
Mutation rate	0.01	0.01

We can apprehend how the community structure changes when our proposed algorithm runs over generations with changing optimization objectives. Two simulations were done: (1) We set the value of $\alpha = 0$, so that the fitness only depends on the modularity index Q. Genetic search was run till convergence of this first stage. (2) We change the value of α from 0 to 0.01, without resetting the chromosomes we had after convergence of stage (1). Introduction of a non-zero α changes the fitness criterion and GA search would converge at a different state where the number of nodes in

different partitions are forced to balance. Detail parameters settings, for this experiment, are shown in Table 3.

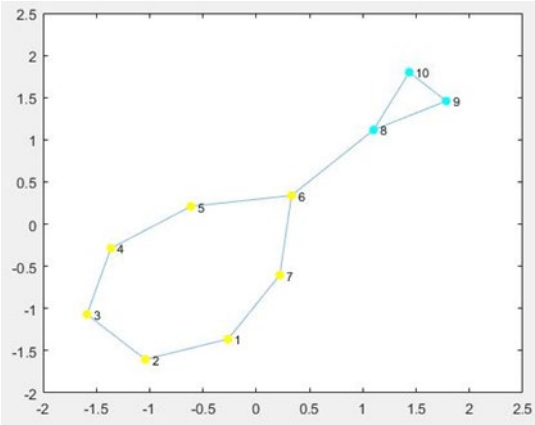

Fig. 7. Stage (1) convergence result ($\alpha = 0$), two unbalanced clusters, minimum cut, $Q = 0.34$.

The experiment is performed 30 times. For this small network, the result of several runs are the same, except the convergence takes a little longer or shorter generations. For stage (1), the convergence, almost always, is reached in just 5 generations. Fig. 7 shows the result when GA reaches stage (1) convergence. The network is divided into two communities: one community consisting of nodes {#1, #2, #3, #4, #5, #6, #7} colored yellow, and the other community consists of nodes {#8, #9, #10}. There is no error node, it is a lowest cut partitioning, and the modularity index was $Q = 0.34$. But, the nodes in two groups are not balanced. The other important question is, whether this partition maximizes the modularity index or not, in spite of the fact that the only fitness criterion was the modularity index in stage (1) of genetic search. For stage (2) simulation, the fitness function is modified, changing the value of α from 0 to 0.01. This term will drive the genetic search to balance the number of nodes in the two clusters. Stage (2) search converges in 3 more generations. Fig. 8 illustrates the resulting partition, after stage (2) convergence. We have three groups of nodes, cluster whose members are {#1, #2, #3, #4} colored yellow, members of the pink community are {#5, #6, #7}, and the blue community nodes are {#8, #9, #10}. With $\alpha = 0.01$, we could achieve communities with nodes balanced, the number of error node is still zero, and modularity index Q is improved to 0.39. Another possible partition would be two groups, one consisting of nodes {#1, #2, #3, #4, #5}, and the other {#6, #7, #8, #9, #10}. We did not have that result, with genetic search.

From the results of Fig. 7 and Fig. 8, we can conclude that by adding the new factor of variance of nodes in the fitness function, objective of genetic search, we could achieve balanced partition, which is important in many graph partitioning applications. In addition, we could improve the modularity index. How to set the proper value of α, and how

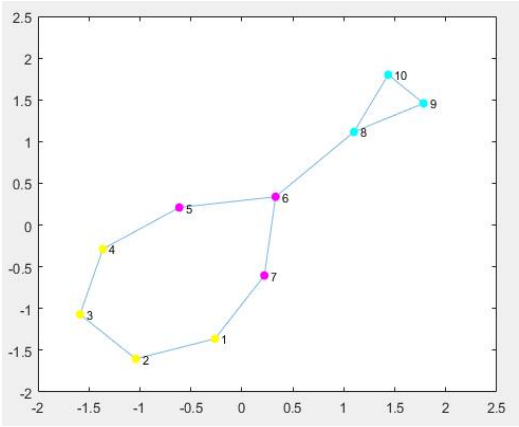

Fig. 8. Communities formed at the end of Stage 2 convergence, 3 clusters are formed, nodes are balanced, $Q = 0.39$.

it depends on the size of the problem, are discussed later in this section.

4.2 GA Parameter Setting

For efficient genetic search, optimal parameter setting is important. It would result in faster convergence, and to a better solution. Depending on the problem complexity, appropriate population size, crossover and mutation rates are different. For example, larger problems will need large population size, longer generations, high crossover and mutation rates, to ensure better exploration of the search space to avoid converging to local minimum.

During the initial generations, it is important to ensure sufficient exploration so that the potential region of global optimum is not missed. Large values of crossover and mutation probabilities is required to facilitate sufficient exploration of the search space. On the other hand, large crossover and mutation probabilities will disrupt chromosomes which already attained a high fitness value, and approaching optimum. When chromosomes are near optimum values, crossover and mutation could destroy the results achieved during previous generations of search. The best approach is to start with high crossover and mutation rates, and slowly reduce them as convergence is approached. In all our simulations, we take that strategy. The detail about GA parameter settings in different experiments is explained in section 4.3. To reduce probability of disrupting good chromosomes, crossover and mutation rate are reduced in steps.

In section 4.1, though we have shown that introducing α improves balance and modularity index of clustering, it is important to estimate the appropriate value of the coefficient. An equal priority to the two terms of the fitness function means $(1-Q) \approx \alpha \times \sigma$. At the end of stage 1, we can take the best chromosome and estimate the value for α. Based on the value of Q obtained for the network in Fig. 8 obtained after stage 1, the estimated value of α would be ~0.2, if we assign equal priority to both fitness terms. But, in actual

experiment we have seen that a much lower value of α = 0.01 is a better choice. The other approach could be to start with a low value of α and gradually increase it, and monitoring the change of modularity index over generations. In the next section, we discuss simulation results with larger real world networks. In all simulations, we start with high crossover and mutation rates, and then gradually decrease. The starting value of α is zero and slowly increased in steps.

4.3 Experiment with a Real-world Networks

In this section, we explain the simulation and results of clustering using the proposed genetic algorithm on several real-world networks. We compared results with three other popular heuristic algorithms, as proposed by Newman (2006), Aloise et al. (2010) EIG (Eigenvector-based algorithm), and Blondel (Blondel et al., 2008) (Louvain algorithm). The different data used in our simulation experiments, together with their names and structure details, are listed in Table 4. The same set of data was used in other works (Dinh et al., 2015; Aloise et al., 2010). The datasets are available on the internet (Koblenz, 2013). Short descriptions of different networks are given here. Zachary's karate club is a social network of a university karate club, where 34 nodes are members of the karate club. A link between a pair of nodes exists when the two members have interaction outside the club. Dolphin Network contains an undirected social network of frequent associations between a pair of dolphins in a community of 62, living off a creek called Doubtful Sound in New Zealand. “Books about US Politics” is the network of books published around the year 2004 US presidential election and sold by Amazon.com. Books are the nodes.

Edges between books represent frequent co-purchasing of a pair of books by the same buyer. The link weight is 1/0 by a threshold. “Football” network is the American College Football network, where nodes are teams, and links are football games between Division IA colleges during the fall of the year 2000. “Jazz Musicians” is the collaboration network between a pair of musicians. Each node is a musician and edge is the relation between two musicians having played together in a band. The data is collected in the year 2003. The number of nodes and edges of all networks is listed in Table 4.

Table 4. Structure of networks for experiments.

ID	Name	#Nodes	#Edges
1	Zachary’s karate club	34	78
2	dolphin’s social Network	62	159
3	Books about US politics	105	441
4	American college football	115	613
5	Jazz musicians	198	2,742

The detail of the GA parameters, as set for our simulations, is shown in Table 5. As the network size increases, we increased the initial values of crossover and mutation rates, for better exploration of the search space. Population size is fixed throughout generations. Initial value of α is zero. For network #1 and #2, crossover rate and mutation rate were decreased in steps of 0.1 and 0.01 respectively. For large networks, e.g., Books, American College Football, Jazz Musicians, the initial crossover and mutation rate were higher, and were decreased by 0.1 and 0.01 at each step.

Once there is a convergence, with α = 0, the value of α is increased, in steps when the GA search converges for a few generations. Each experiment, with a particular dataset, is repeated for 30 times. The average of the modularity index value attained after 200 generations is reported in Table 6.

Table 4. Initial parameters of genetic algorithm.

Network ID	#1, #2	#3, #4	#5
Generation	200	200	200
Population size	20	20	20
Alpha	0	0	0
Crossover rate	0.3	0.5	0.9
Mutation rate	0.03	0.05	0.1

For every data, the algorithm converged before running 200 generations. Typical plots of the change of modularity index with search generations are shown in Fig. 9.

As is evident from Fig. 9, the genetic search converges in 60 to 80 generations. It takes a little more search generations when the network size increases. As already mentioned, we started with α = 0, and gradually increased it, after the modularity index remain same over a predefined number of generations, which is set at 10 in all simulations. Immediately after α is increased, the modularity index improves, as is evident from the step jumps in the plots of Fig. 9. This is not always clear from Fig. 9, as there are occasions when the fitness is not changed for a few generations (< 10), and then increases by itself without increasing the value of α. The algorithm converges in less than 100 generations.

The actual membership of nodes to different clusters are known for two networks: “Zachary’s Karate Club” and “American College Football”. But, that real-world grouping does not always make the highest value of modularity index. In Aloise et al. (2010), the modularity values were claimed to be optimum, as shown in the rightmost column of Table 6. But, as shown in column second, results from the proposed algorithm could achieve better or equally good modularity index compared to any previous works, especially for “Dolphins social networks” and “Books about US politics” data.

Modularity index, as defined in Equation (1), is the measure for ensuring good community structure in a network. High modularity index value means network have dense connections between the nodes within community but sparse connections between nodes in different communities,

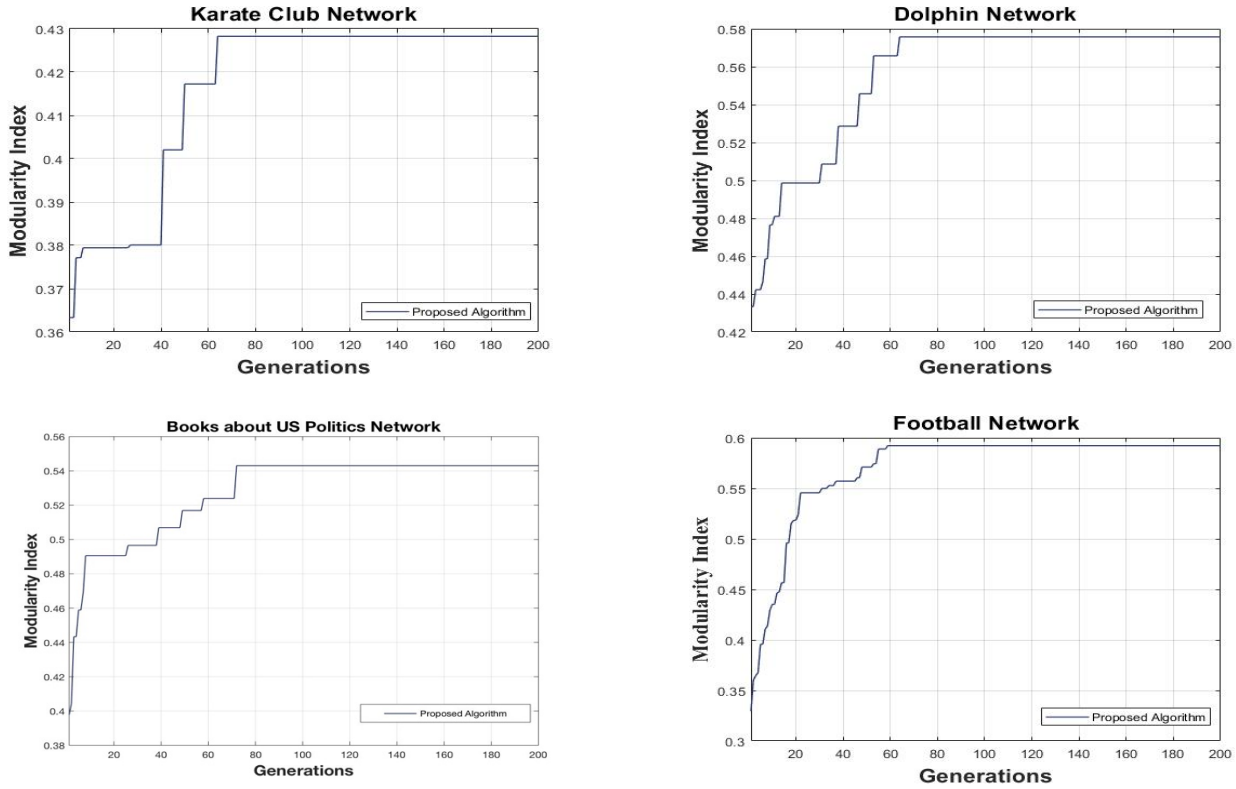

Fig. 9. Improvement of the modularity index with generations, for different networks.

Table 5. Comparison of modularity indices obtained by different algorithms: Equation (1), Louvain algorithm (Blondel et al., 2008), EIG (Aloise et al., 2010), and Newman (Newman, 2006).

	Proposed algorithm	Blondel et al. (2008)	Aloise et al. (2010)	Newman (2006)	Eq.(1)
Karate	0.42	0.42	0.42	0.42	0.40
Dolphin (Lusseau, 2003)	0.58	0.42	0.53	0.49	0.38
Books	0.55	0.52	0.53	0.53	–
Football	0.60	0.61	0.60	0.49	0.55
Jazz (Gleiser and Danon, 2003)	0.44	0.44	–	0.42	–

Table 6. Comparison of erroneous nodes of the proposed algorithm and Louvain algorithm.

	Proposed algorithm	Louvain algorithm	Actual data
Karate	4	11	3
Dolphin	3	3	2
Books	10	4	–
Football	12	8	10
Jazz	7	10	–

if the nodes are properly assigned to their respective communities.

The actual community information of "Karate", "Dolphin" and "Football" data is available. We calculated the corresponding modularity index, using Equation (1) and included in the last column of Table 6. Our algorithm could achieve higher Modularity Index compared to the actual communities. This is possible, because in real life the community structure does not always form to attain highest modularity.

Louvain algorithm (Blondel et al., 2008) is an efficient heuristic bottom up clustering algorithm which converges fast. Our algorithm could achieve higher or equally good fitness values compared to Louvain algorithm (Blondel et al., 2008). We also compared the result with the Eigenvector-based algorithm (EIG) (Aloise et al., 2010). Our algorithm could achieve higher fitness value when the number of nodes is small, e.g. Karate, Dolphin, and Jazz. For the large networks, e.g., Books, and Football, our algorithm and EIG algorithm achieve similar results.

In addition to modularity index, and node balancing, the other evaluation criterion for good clustering is the number of error nodes, as explained in section 2.1. In our genetic search, in the fitness function, it is not included as an optimization parameter. We are not minimizing the number of error nodes. In Table 7 we showed the number of error nodes after convergence. We can see that for the "Dolphin social networks" data, and "Books on US politics" data, GA

search did not produce the minimum error node clustering, though in both cases we could achieve the best modularity indices.

The number of error nodes are shown in Table 7. Except for "Books on US Politics", we could achieve better or equally good number of error nodes, compared to Louvain Algorithm. Error nodes from actual data are listed in the last column. The number of error nodes using the proposed algorithm is slightly higher than the actual data.

4.4 Stability of the algorithm

As the genetic search starts with random chromosomes, and because the search space is multi-modal, the solution often converges to a local minimum. In fact, most of the heuristic algorithms (like *k*-means) initialize randomly, and depending on the initial state, converges to different solutions. Thus, the stability of result is an important concern with such algorithms. In this section, we evaluated the stability of our proposed algorithm, and compared it with Louvain algorithm (Blondel et al., 2008), which is a bottom up greedy heuristic algorithm starting with a randomly selected node.

For evaluation of stability, we ran both the proposed algorithm and Louvain algorithm for 30 times, for all the 5 networks listed in Table 4. The modularity index values for 4 networks, on 30 different runs, for both algorithms, are plotted and shown in Fig. 10. "Zakary's karate club" network is small with only 34 nodes. Almost all runs converged to the similar nearly optimum results, except Louvain converged twice to a bad solution. For "Dolphin social networks" and "Books on US politics", the proposed algorithm did better than Louvain, in all runs, and the variance of the results are similar.

For the rest two networks, "American College football" and "Jazz", the results are similar too. The average of the modularity indices for all the five networks are given in Table 6, for the proposed algorithm and Louvain algorithm. The variance of results were calculated, and presented in Table 8. The low variance confirms stability of the proposed algorithm, in comparison to Louvain.

Table 7. Standard deviation of 30 runs.

Networks	Louvain algorithm	Proposed algorithm
Karate	0.007	0.004
Dolphin	0.004	0.007
Books	0.006	0.002
Football	0	0.004
Jazz	0.003	0.002

5. CONCLUSIONS

The result of community detection in a network depends on what is the optimization criterion. Depending on the application, the optimization metric could be different. There could be cases, where the actual community may not be the one with highest modularity index. Depending on the application, the objective function could be Inter-Cluster

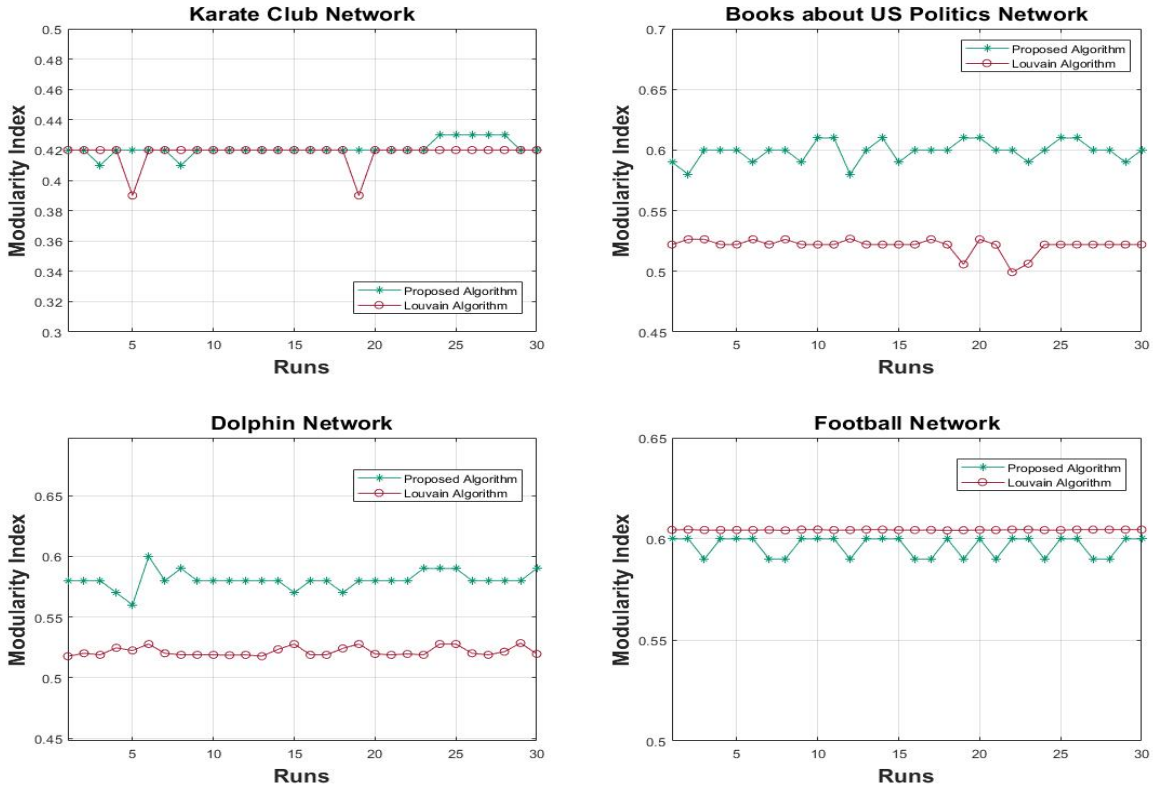

Fig. 10. Simulation results from 30 runs.

Conductance, node balancing, modularity index, or minimizing error nodes.

In this paper, we proposed a two stage genetic algorithm for community detection in a network. In stage 1, the modularity index is maximized, and in stage 2 the fitness function adds another optimization parameter to balance the number of nodes in different communities. If the network consists of highly modular, nearly balanced communities, the first stage of the algorithm could achieve the optimum result. The probability of GA to converge in local minimum is low. When the communities are not well defined or overlapping, GA and other heuristic algorithms often converge to local optimum. We have shown, by simulation experiments, that our two stage algorithm could improve not only the balance of different clusters but also the modularity index.

Setting proper balance in the fitness function, between two optimizing parameters is difficult. In our previous works (Lu and Chakraborty, 2020), we used fixed α, but could not achieve optimal solutions. In this works, monitoring the convergence, we slowly increase alpha value to avoid local minimum and attain better results.

We need to continue simulation to other networks, with overlapping and hierarchical clusters, and how to program the genetic search to explore them in stages. We also need to run our algorithm on bigger real life networks, and evaluate the efficiency and efficacy of the algorithm. We need to explore with fitness function that minimizes the number of error nodes.

The algorithm is suitable to solve networks of a few thousand nodes in reasonable time, but does not scale for networks with the number of nodes counting a million to a billion. As Ant Colony Optimization (ACO) is faster compared to GA, we are working to extend our idea to solve the problems using ACO, and do comparison of computational efficiency.

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

Agarwal, G., Kempe, D. 2008. Modularity-maximizing graph communities via mathematical programming, Eur. Phys. J. B, 66.

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., Liberti, L. 2010. Column generation algorithms for exact modularity maximization in networks, Phys. Rev. E, 82.

Andreev K., Räcke, H. 2006. Balanced graph partitioning, Theory of Computing Systems, 39, 929–939.

Barabási, Albert-László, Albert, Réka, 1999. Emergence of scaling in random networks, Science, 286, 509–512.

Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E. 2008. Fast unfolding of communities in large networks, Statistical Mechanics: Theory and Experiment.

Chakraborty, G. Sato, N. 2017. A reliable graph clustering method using genetic algorithm, International Symposium on Nonlinear Theory and Applications, Cancun, Mexico, December.

Clauset, A., Newman, M.E.J., Moore, C. 2004. Finding community structure in very large networks, Physical Review, 066111.

Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakal, M. 1992. The complexity of multiway cuts (Extended Abstract), 24th. Annual ACM STOC, 241–251.

Dinh, T.N., Li X., Thai, M.T. 2015. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits, 2015 IEEE International Conference on Data Mining, Atlantic City, NJ, 101–110.

Erdős, P., Rényi, A., 1959. On random graphs, Publicationes Mathematicae, 6, 290–297.

Fiedler, M. 1975. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, Czechoslovak Mathematical Journal, 25, 619–633.

Fouss, F., Pirotte, A., Renders, J.M., Saerens, M. 2007. Random-Walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE trans. On Knowledge and Data Engineering, 19, 355–369.

Gergely, P., Imre, D., Illes, F., Tamas, V. 2005. Uncovering the overlapping community structure of complex networks in nature and society, Nature, 435, 814–818.

Ghosal, A.K., Das, N., Bhattacharjee, S., Chakraborty, G. 2019. A fast parallel genetic algorithm based approach for community detection in large Networks, 11th International Conference on Communication Systems and Networks (COMSNETS2019), Bengaluru, India.

Gleiser, P., Danon, L. 2003. Community structure in jazz, Advances in Complex Systems, 6, 565–573.

Hafez, A.I. Ghali, N.I. Hassanien, A.E., Fahmy, A.A. 2012. Genetic algorithms for community detection in social networks, in 12th International Conference on Intelligent Systems Design and Applications (ISDA), 460–465.

Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S. 1970. Multilevel hypergraph partitioning: Applications in VLSI domain, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 7, 69–79.

Kernighan, B.W., Lin, S. 1970. An efficient heuristic procedure for partitioning graphs, Bell System Technical Journal, 49, 291–307.

Koblenz Network, 2013. Available: http://konect.uni-koblenz.de/.

Li, Y., Liu, G., Lao, S.Y. 2013. A genetic algorithm for community detection in complex networks, Central South University, 20, 1269–1276.

Lu, Y.C., Chakraborty, G. 2019. Definition and goal of graph clustering motivation to explore a new Algorithm," ICAST conference, Japan.

Lu, Y.C., Chakraborty, G. 2020. Improving efficiency of graph clustering by genetic algorithm using multi-objective optimization, International Journal of Applied Science and Engineering.

Lu, Y.C., Chakraborty, G., March, 2019. An efficient graph clustering algorithm using multiobjective pareto optimization, Advance Information Technology, 29–30 Taichung, Taiwan.

Lusseau, D. 2003. The emergent properties of a dolphin social network, Proc. of the Royal Society of London B, 270, 186–188.

Matsuo, Y. 2003. Prediction and discovering small world network, Society of Artificial Intelligence (in Japanese), 18.

Matsuo, Y. 2005. Network structure and its emergence, Proceedings of AAAI, 11–14.

Newman, M.E.J. 2006. Modularity and community structure in networks, Proceedings of the National Academy of Sciences, 103.

Newman, M.E.J. 2010. Networks: An introduction, Oxford.

Pons, P., Latapy, M. 2006. Computing communities in large networks using random walks, Graph Algorithms and Applications, 10, 191–218.

Pothenf, A., Simon, H.D., Liou, K.P. 1990. Partitioning sparse matrices with eigenvectors of graphs, Industrial and Applied Mathematics, 11, 430–452.

Rossi, R., Ahmed, N.K., Koh, E., Kim, S. 2019. Linear-time hierarchical community detection, https://arxiv.org/pdf/1906.06432.pdf.

Schaeffer, S.E. 2007. Graph dlustering, Computer Science Review, 1, 27–64.

Shang, R., Bai, J., Jiao, L., Jin, C. 2013. Community detection based on modularity and an improved genetic algorithm, Physica A: Statistical Mechanics and its Applications, 392, 1215–1231.

Shi, J., Malik, J. 2000. Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence", IEEE Transactions on, 22, 888–905.

Šíma, J., Schaeffer, S.E. 2006. On the np-completeness of some graph cluster measures. In SOFSEM 2006: Theory and Practice of Computer Science, 530–537.

Song, Y., Li, J., Zhang, X., Liu, C. 2012. Community detection using parallel genetic algorithms, Fifth International Conference on Advanced Computational Intelligence (ICACI), 374–378.

Telesford, Q.K., Joyce, K.E., Hayasaka, S., Burdette, J.H., Laurienti, P.J. 2011. The ubiquity of small-world networks, Brain Connect, 1, 67–375.

Watts, D.J., Strogatz, S.H., 1998. Collective dynamics of 'Small-World' networks, Nature, 393.