Prctica 3:Optimization with Genetic Algorithms

Pulak Roy

Universitat Rovira i Virgili,
Av. Països Catalans 26,
E-43007 Tarragona, Catalonia, Spain
e-mail {pulakroy.roy}@estudiants.urv.cat

1 Introduction

A genetic algorithm (GA) is great for finding solutions to complex search problems. They're often used in fields such as engineering to create incredibly high quality products thanks to their ability to search a through a huge combination of parameters to find the best match. For example, they can search through different combinations of materials and designs to find the perfect combination of both which could result in a stronger, lighter and overall, better final product. They can also be used to design computer algorithms, to schedule tasks, and to solve other optimization problems. Genetic algorithms are based on the process of evolution by natural selection which has been observed in nature. They essentially replicate the way in which life uses evolution to find solutions to real world problems. Surprisingly although genetic algorithms can be used to find solutions to incredibly complicated problems, they are themselves pretty simple to use and understand.

2 Objective

Implementation of a genetic algorithm (GA) for the clustering of the nodes of a graph, by means of the optimization of modularity.

Modularity:

Let G be an undirected network (graph) with N vertices, L edges, no self-loops and adjacency matrix a_{ij}. If we have a partition C of the nodes of the graph in disjoint clusters, the *modularity* Q is defined as

$$Q(C) = \frac{1}{2L}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(a_{ij} - \frac{k_i k_j}{2L}\right)\delta(C_i, C_j)$$

where k_i is the degree (number of edges) of node i, and $\delta(C_i,C_j)$ is 1 if nodes i and j belong to the same cluster, 0 otherwise. More precisely,

$$k_i = \sum_{j=1}^{N} a_{ij}$$

$$2L = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} = \sum_{i=1}^{N} k_i$$

Modularity is zero when all nodes belong to the same cluster, and in general it may take values in the range -1 $\leq$ Q(C) $\leq$ 1. We are only interested in partitions of the nodes in two clusters, which we will call the left and right clusters respectively. Therefore, if we define Si as 0 if node i belongs to the left cluster, and 1 if it belongs to the right cluster, then the partition in two clusters is completely determined by the vector S, and $\delta_{(C_{\mathrm{i}},C_{\mathrm{j}})} = SiSj + (1 - S_{\mathrm{i}})(1 - S_{\mathrm{j}})$

Genetic Algorithm:
Given the network G, any vector S may be seen as the chromosome corresponding to a partition in two clusters, and the fitness is related to its modularity Q(S). The objective is the implementation of a genetic algorithm to obtain the partition which maximizes modularity.The basic process for a genetic algorithm is:

1. Initialization - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.

2. Evaluation - Each member of the population is then evaluated and we calculate a 'fitness' for that individual. The fitness value is calculated by how well it fits with our desired requirements.

3. Selection - We want to be constantly improving our populations overall fitness. Selection helps us to do this by discarding the bad designs and only keeping the best individuals in the population.

4. Crossover - During crossover we create new individuals by combining aspects of our selected individuals.

5. Mutation - We need to add a little bit randomness into our populations' genetics otherwise every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individuals genome.

6.And repeat - Now we have our next generation we can start again from step two until we reach a termination condition.

Fitness:
In genetic algorithm, it is important to have a good fitness function.For example, if we consider $F = (1+Q)^n$ for a Certain exponent n (eg n = 2) ,then we are giving more importance to higher values of Q. Overall, the objective is to maximize modularity Q, and the fitness function is just a means to accomplish it.

Data:
Input: a network in Pajek format (*.net)
Output: the highest modularity partition found, in Pajek format (*.clu), and its corresponding value of modularity.

Implementation approach:

Language: Java

Tool: Eclipse JavaEE mars

Implementation decisions:

Selection:
There are several processes of selection like: Roulette wheel, Rank selection and Tournament selection.As Tournament selection is efficient to code. It also works on parallel architectures and allows the selection pressure to be easily adjusted (changing the number of individuals in a tournament). If the tournament size is larger, weak individuals have a smaller chance to be selected. Thats why I have chosen Tournament selection

methodology.This methodology includes choosing few individuals at random from the population (a tournament).Then individual with the best fitness (the winner) is selected for crossover.

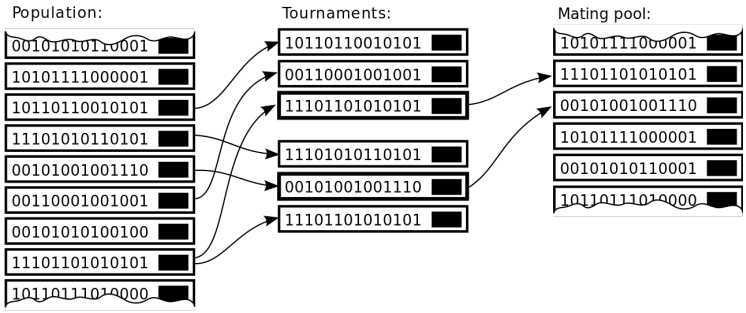

Figure1: Tournament Selection

Crossover:
There are several types of cross over such as One-point ,Two-point ,Uniform Crossover.Among them uniform Crossover uses a fixed mixing ratio between two parents. Unlike one and two-point crossover.Uniform Crossover enables the parent chromosomes to contribute the gene level rather than the segment level.The Uniform Crossover evaluates each bit in the parent strings for exchange with a probability of 0.5. Empirical evidence suggest that it is a more exploratory approach to crossover than the traditional exploitative approach that maintains longer schemata. This results in a more complete search of the design space with maintaining the exchange of good information. For this reason I have chosen uniform crossover approach to create new individuals.

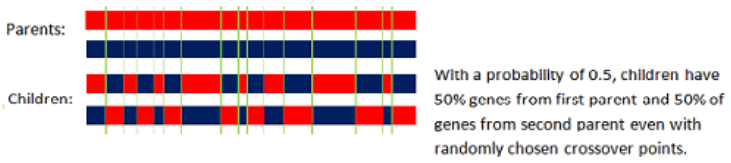

Figure2: Uniform Crossover

Mutation:
Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next.Different types of mutation exists such as bit string,flip Bit,boundary, Non-Uniform,Uniform,Gaussian etc.Among them Non-Uniform mutation consists probability (or ratio) is basically a measure of the likeness that random elements of your chromosome will be flipped into something else. For example if your chromosome is encoded as a binary string of length 100 if you have 1% mutation probability it means that 1 out of your 100 bits (on average) picked at random will be flipped.In this case I have chosen this non-uniform mutation process.

Results (partitions)
There are lots of input files, among them 20x2+5x2.net file we get the maximum fitness value of 5.007432765936259.Then there is another network which depicts partitioned network well is zachary_unwh.net Following figures shows some of the network partitions and their corresponding fitness value.

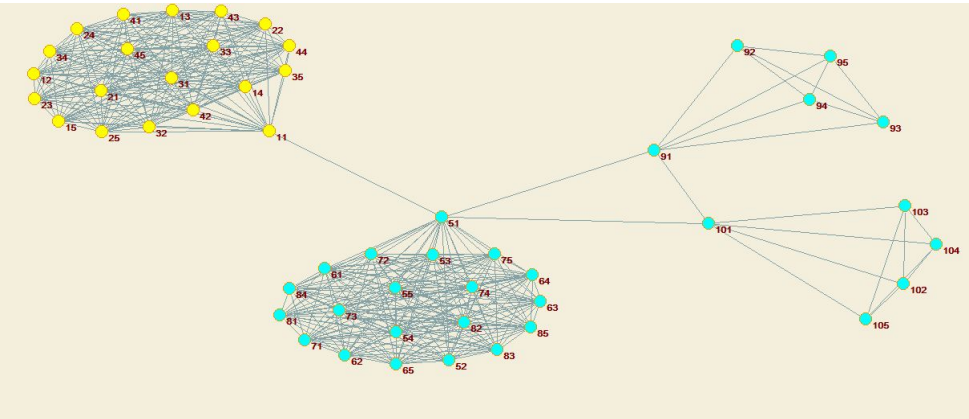

Figure3: Network partition:20x2+5x2 with fitness : 5.007

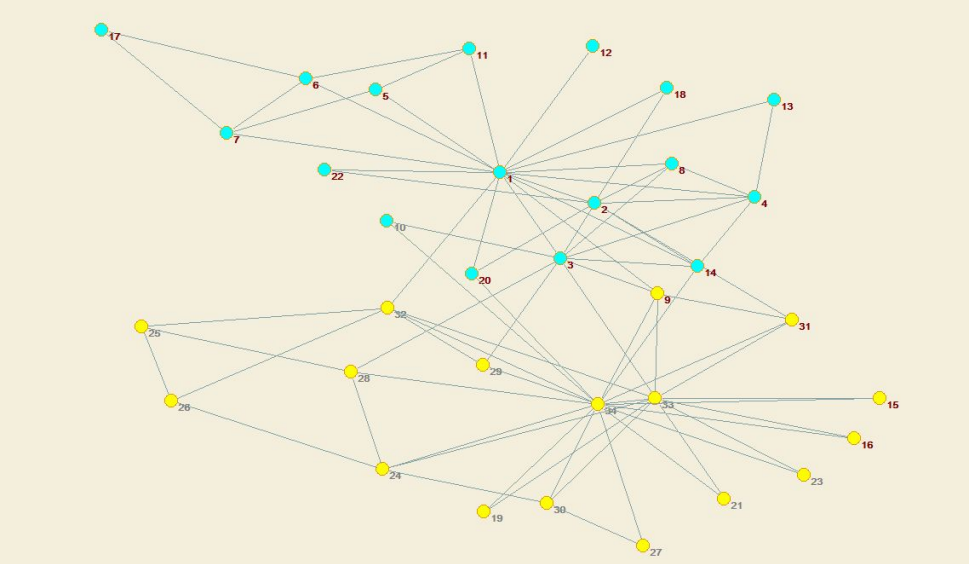

Figure4: Network partition:zachary_unwh with fitness : 3.541

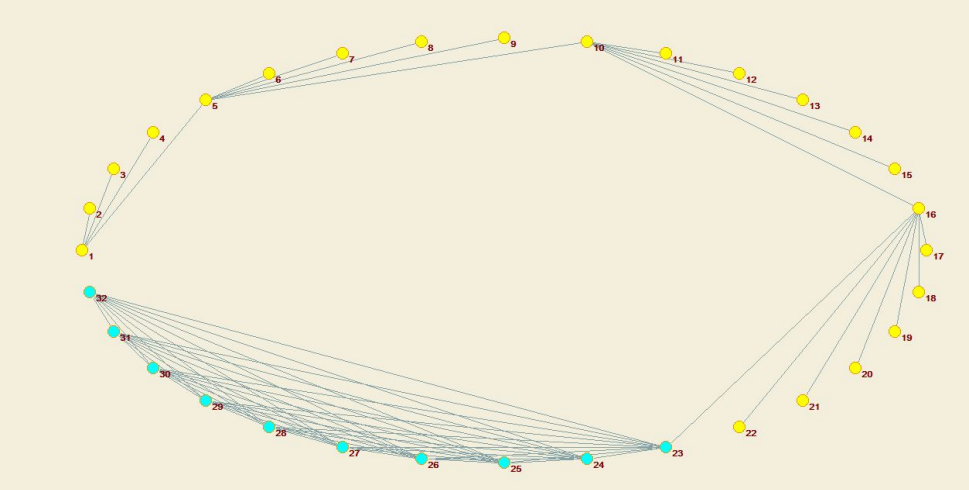

Figure5: Network partition:clique_stars with fitness : 4.076

3 Conclusion

Genetic algorithms are good at taking huge search spaces on navigating them for optimal combination of things. Solutions you might not otherwise find in a life time.Rather than looking on the best solution it searches for good and robust solution written against fitness criteria, so it avoid local optimal and searches for global fitness.

References

1. "Creating a genetic algorithm for beginners", [Online]. Available: , http://www.theprojectspot.com/tutorial-post/creating-a-genetic-algorithm-for-beginners/3 [Accesed: Jun. 5, 2016]

2. "Selection (genetic algorithm)", [Online]. Available: , https://en.wikipedia.org/wiki/Selection_(genetic_algorithm) [Accesed: Jun. 5, 2016]

3. "Crossover (genetic algorithm) ", [Online]. Available: , https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm) [Accesed: Jun. 5, 2016]

4. "Mutation (genetic algorithm)", [Online]. Available: , https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm) [Accesed: Jun. 5, 2016]