Nasri Chiheb

UNIVERSITAT ROVIRA I VIRGILI

Neural and Evolutionary Computation

A4: Optimization with Genetic Algorithms

Introduction:

The goal of this assignment is to implement a genetic algorithm for graph clustering problem by optimizing modularity. This solution will be applied with python and tested on 20 different graphs. The partition will be maid in two clusters using modularity as fitness function. In this report we will discuss about the implementation and the results.

Description of the implementation:

The implementation is made with python as it the language that I know the most even if it is not the most optimized for this task. The networkx library will be used to read the Pajek format graph and to compute modularity of the partitions. In the code you will find a modularity function that I tried to make but the results are not the same. The random library is used for the random initialization of the chromosomes and the different other probabilities. Finally, I used matplotlib for different plots. The code is made to be as generic as possible. It needs as input the .net file path, size of the population, number of generations, number of parents to choose for tournament method selection and the mutation and crossover probabilities.

The genetic algorithm was made following the pseudo code in the course slides. The initialization is made randomly and stored in a dictionary. The fitness function is the modularity function implemented by networkx (a version of my modularity function is available, but it is not working). The selection is made following tournament method to maintain diversity. The crossover is made by Uniform method with a probability of 50% of swap. For the mutation each gene has a probability of mutation. Elitism is not implemented yet but will come in V2 of the algorithm.

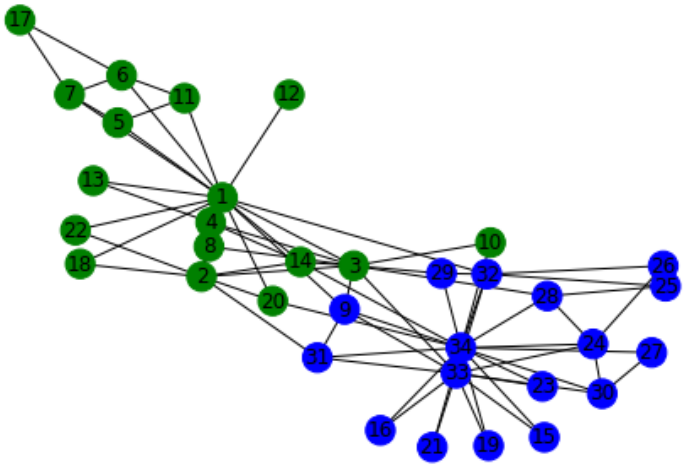
At the end a representation of the nodes with different colors is shown. A graph is plotted showing the evolution of the population best modularity. The best partition is saved in ".clu" file.

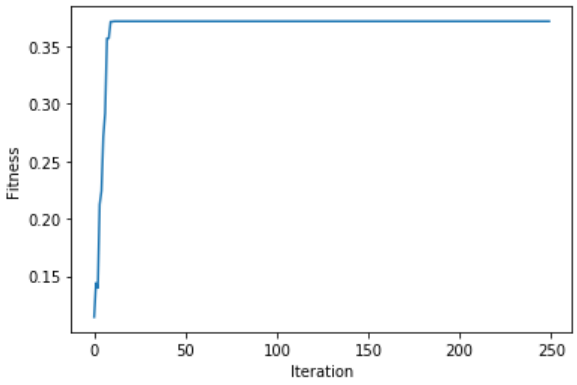
I have tried multiple combinations for the genetic algorithm by choosing different selection schemes, crossover, and mutation methods. Due to computation time I couldn't optimized at its maximum

Results:

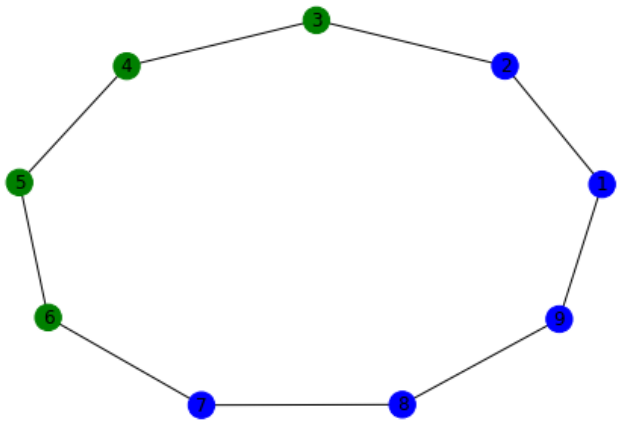
All the results here are made with a population size of 100 and 250 generations. Algorithm doesn't work with graph that have different labels than numbers (problem with modularity function). To apply the algorithm on them please modify the ".net" file. Here are the graphs that work with the algorithm.

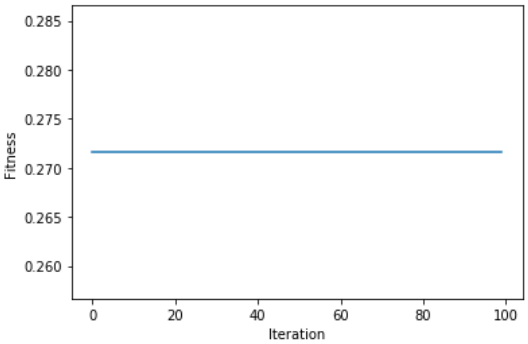
zachary_unwh.net : Modularity = 0.3717948



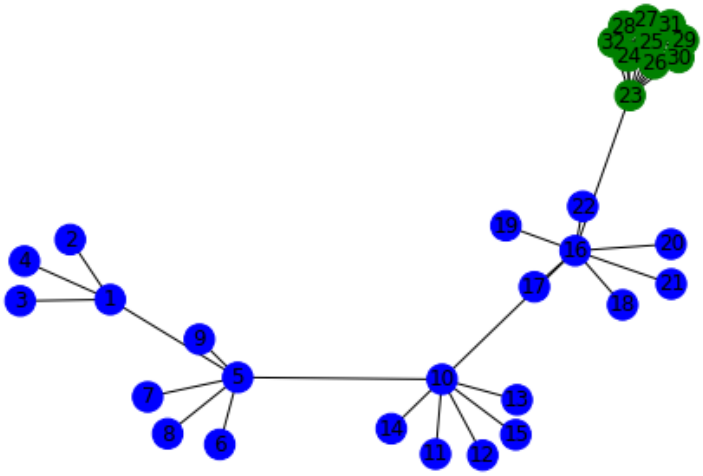


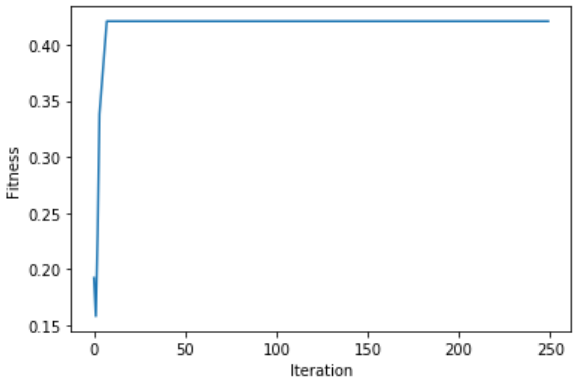
Circle9.net : modularity = 0.271604



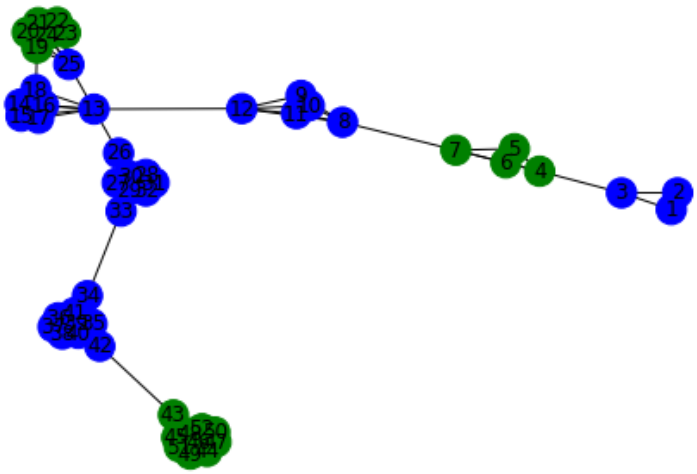


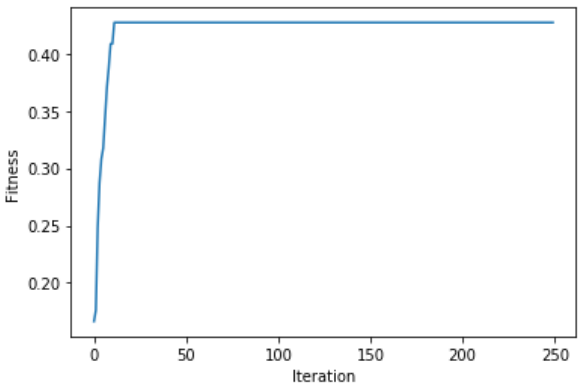
clique_stars.net : modularity = 0.420917



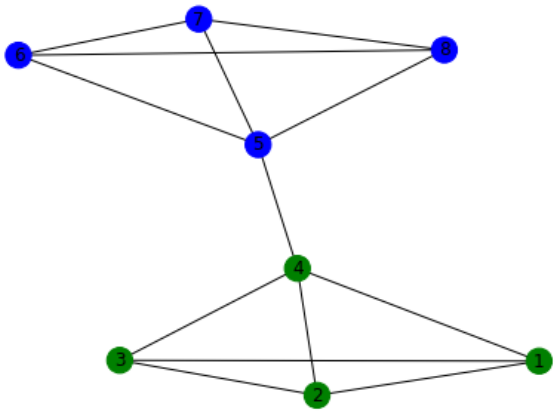


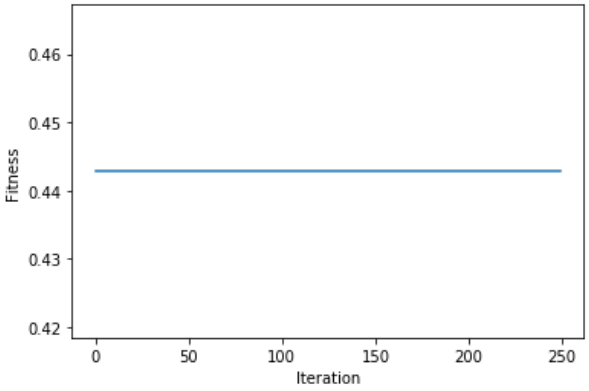
cliques_line.net: modularity = 0.427139974



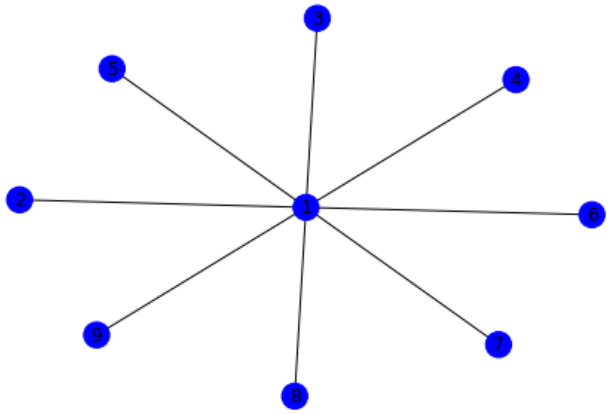


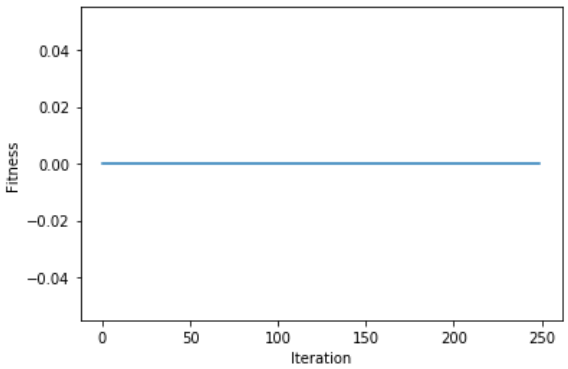
graph3+2+3.net: modularity = 0.44285



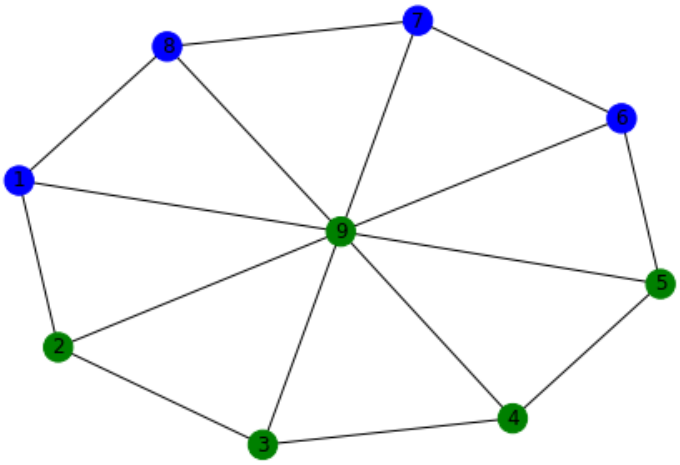


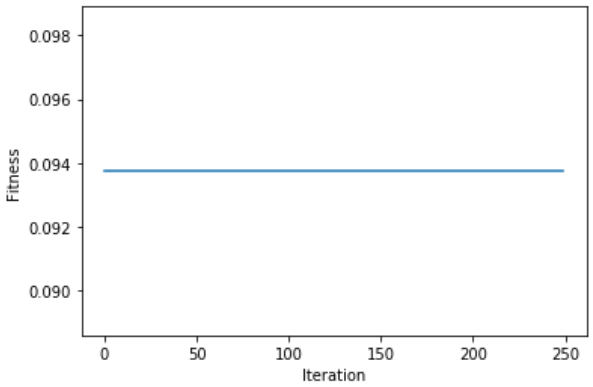
star.net: modularity = 0





wheel.net: 0.09375





How to run:

Prerequisite :

- Python 3.7 or greater
- Networkx, random, matplotlib libraries

Run :

- Open main.py
- Main.py , utils.py needs to be in the same directory same for graph folder
- Select the configuration values
- Run