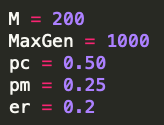
Exam 2

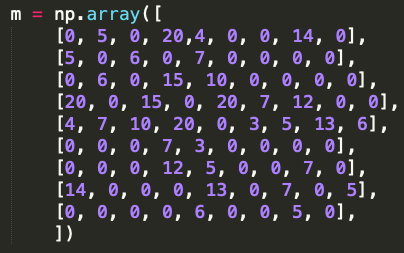
AEEM 7010

XIN LI

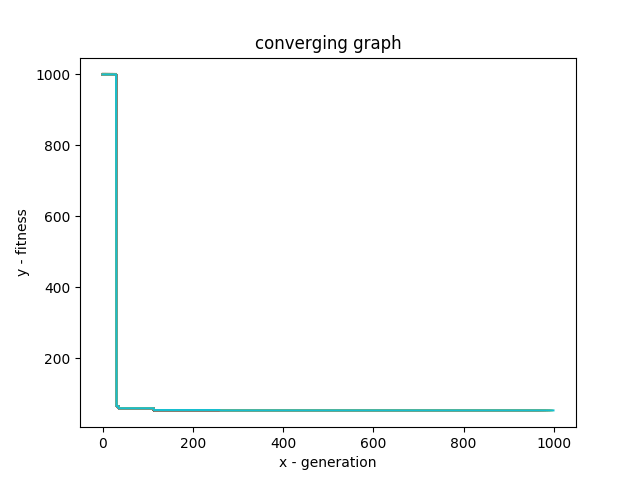
**Problem 1**

a)

As the computation complexity is much lower without using FRBS (Fuzzy Rule Based System), I set 200 for my population size, as known as number of chromosomes per generation, so that my GA can have larger searching space for each generation, and it will help converge faster. I set 1000 for number of generations, just to make sure it is enough for convergence and as a result, my GA converge at about 200 generations. As I am not sure how my crossover affect the fitness of parent and corresponding child, I set 0.5 for the PC in order to make the GA has a 50-50 even chance of picking parent or child. 0.25 is picked for PM in order to have a ¼ chance of getting mutated for each gene. This value is picked for not being too high, so that the population can be stable enough to be benefited from self-crossover. It is also not too low, so that certain degree of randomness ensure the GA can explore the unknown searching space. I set 0.2 for ER since 0.2 is a common ratio for elites if no other relevant information is given.

My pipeline is designed to be generated randomly, however with some constrains, such as, all wells must be connected in a chain form, as well as each well is only called once in each chain. Then I provide a distance map, which is generated from the given figure (the connection & distance among wells), and use this distance map to calculate the fitness, as known as the total distance for each pipeline. During the calculation of distance, if a “0” is detect, the calculation is force to be stop and the fitness will be assigned to “999” to indicate this pipeline is a bad choice. This is because “0” in the distance map indicates that there is not route connected between these 2 wells. With help of crossover, mutation, and elitism, bad choices of pipelines will eventually be eliminated, and the optimal pipeline choice will be found as the fitness curve is converged. If terminal well head has to be well head 1 (as mentioned by HINT), then the best fitness that I found is distance of 53, with the pipeline setup as “[9, 8, 7, 5, 6, 4, 3, 2, 1]”. However, if the terminal well head is not the concern, a better sequence as “[6, 4, 7, 8, 9, 5, 1, 2, 3]” can produce a shorter distance in sum as 52.

As the convergence plot shown below, my GA reach the convergence pretty early at 250 generation.



b)

A very important lesson that I learned from this project is that “ALWAYS analyze the natures of the problem before developing the algorithm”. For example, ask questions such as “Does this problem has some special constraints?”, “Are features of the dataset independent?”, and so on.

For this pipeline problem, constraint plays a significant role so that this problem is different from the normal problems we solved before. At the beginning of the development, I was using 2-parents-crossover method. However, soon I realized that using crossover on 2 different parent pipelines have very high chance of producing wrong child pipeline, which mean some wells can be called more than once, and some wells can be eliminated. Such pipeline choices against the constraint of this problem. After reviewing Kelly’s paper provided on Blackboard, I found that “self-crossover” is a very good solution for solving such issue in traditional crossover method. By using this method, constraint, such as “all wells are connected and are connected only once” can be preserved. Some other constraints, such as “Not all wells have connection to each other” is not preserved by “self-crossover”, but we can use GA to solve this constraint, as we have “penalty” to eliminate the “non-possible” pipeline choices.

Another lesson that I learned from this problem is that the choice of datatype is very import to software development. Sometimes, simple datatype can do better than fancy datatype. I am using 9X9 “ndarray” 2-D binary matrix to represent the pipeline connection, so that a 0 for the cell means no connection, and a 1 for the cell means a connection. It is a fancy datatype and it has very useful library for manipulation. However, I realize it cannot preserve the constraint of “all wells are connected and are connected only once”. After self-crossover, some wells are connected with infinite loop, such as “4 to 3 to 4 to 3 to …”. To solve this issue, I use a simple 1-D array, so that no matter how I self-crossover the pipeline, infinite loop issue will never happen.

c)

First of all, the most obvious challenge will be the larger searching space, so that convergence is harder to reach. If the connections from each well to others are limited, for example, each well is only connected with several other wells instead of all other wells, having large number of well heads may not be too bad. However, if all connection are considered, for example, before we drill the wells, the government want to design the most efficient well network, then it mean we will have searching space, where n is the number of well heads. It will not only slow down the convergence, but also increase the computation time for each generation. Thus, better algorithm and better data structure need to be used to solve such problem.

Another challenge will be the choice of stochastic parameters, such as PC, PM, and ER. It is easy to choose these parameters for small searching space, because it is easy for such searching space to converge as long as the parameters are not chosen carelessly. However, for larger searching space, a good combination of stochastic parameters can significantly increase the converging speed, while a bad combination of such parameters may lead to non-convergence. The bad news is, there is no efficient way to figure out how to determine what is a good or bad combination of such stochastic parameters. It is depending on your instinct and experience of particular problem.

d)

**Problem 2**

a)

b)

c)

**Problem 3**

a)

b)