**Assignment 2**

**Tarek Fouda**

*Nqueens problem using Genetic Algorithm:*

Simply, I implemented the code that perform the genetic algorithm for the Nqueens problem. I initialized 80 as the population size (Because the initial population size should be number of variables multiplied by 10). Performed Genetic algorithm which I basically chose out of this 80 chromosome population, the chromosomes with the best fitness function ( least number of pair attacks) to be the parents of the next generation children.

The crossover technique used in my algorithm was PartiallyMappedcrossover.

**N=8:**

I noticed, by running the code 100 times I was able to get 150 solutions. Also some other times I got 120 solutions. It all depends on how the initializeChromosomes() function initialize the initial population.

**N=16:**

By running the code 100 times, I was able to get 8 solutions, and some other times I got 14 solutions.

Note: I increment the solution as soon as I find no conflicts in a chromosome (Meaning, no queen is attacking another). The computeConflict is the method responsible for this task.

*Nqueens problem using hill climbing algorithm:*

Here the code is a little bit different than the genetic algorithm idea, because I start with a random board that has the queens randomly distributed, then I calculate the conflicts for such a state.

Then list all the possible neighbors for such a state and choose the one with the least conflicts ( better fitness function). If I reach a dead end, then I randomly shuffle all the queens and repeat the steps.

**N=8:**

After 100 iterations, I got 30 solutions and that is the output of my code:

The plot of the initial state

\* \* Q \* \* \* \* Q

\* \* \* \* Q \* \* \*

Q Q \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* Q \* Q \* \*

\* \* \* \* \* \* Q \*

The plot of the 10th state

\* \* \* \* \* Q \* \*

Q \* \* \* \* \* \* \*

\* \* \* \* Q \* \* \*

\* Q \* \* \* \* \* \*

\* \* \* \* \* \* \* Q

\* \* Q \* \* \* \* \*

\* \* \* \* \* \* Q \*

\* \* \* Q \* \* \* \*

Hill climb solutions: 24

**N=16:**

After 100 iterations I got 5 solutions. The output of the code is as shown:

The plot of the initial state

\* \* \* \* \* \* \* \* Q \* \* \* \* \* \* \*

\* \* \* \* Q \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* Q \* \* \* \*

\* \* \* \* \* \* \* \* \* \* Q \* \* \* \* \*

Q \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* Q \* \* \* \* \* \* \* Q

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* Q \* \* \* \* \* \* \* \* \* \*

\* \* Q \* \* \* \* \* \* Q \* \* \* Q \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* Q \*

\* Q \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* Q \* \* \*

\* \* \* Q \* \* Q \* \* \* \* \* \* \* \* \*

The plot of the 10th state

Q \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* Q \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* Q \* Q \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* Q \* \* \*

\* \* Q \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* Q \* \*

\* \* \* \* \* \* \* \* Q \* \* \* \* \* \* \*

\* \* \* Q \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* Q \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* Q \*

\* Q \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* Q \* \* \* \*

\* \* \* \* \* \* Q \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* Q

\* \* \* \* \* \* \* \* \* \* Q \* \* \* \* \*

Hill climb solutions: 5

Note: Solutions for 16 Queens are always less than 8 Queens in both algorithms.

*Optimizing Ackley function using Hill climbing algorithm:*

We have the Ackley function and we want to get the lowest point we can get out of an initial value for X and Y.

I start my code by randomly getting two values; X and Y. I calculate the Ackley function using these two values. Then I randomly get another two values X’ and Y’ with maximum step size 0.1. And I keep repeating this loop for 100 times until I reach the lowest local minima I could get. In a sample run for my code, I got:

F(x,y) = 1.0419068714912543 Such that X = 0.019341931902274977, and Y = 0.1623988672508263

--------------------------

F(x,y) = 0.7700828864801039 Such that X = 0.031037943416233414, and Y = 0.1271098292467889

--------------------------

F(x,y) = 0.6537016093819901 Such that X = 0.045875615653721426, and Y = 0.1065980693331035

--------------------------

F(x,y) = 0.5695328363830976 Such that X = 0.04744853950701066, and Y = 0.09380333914993488

--------------------------

F(x,y) = 0.23914987802878462 Such that X = 0.02153463838117499, and Y = 0.051534366509899475

--------------------------

F(x,y) = 0.06925395633583609 Such that X = 0.01848200482576531, and Y = 0.008936846158111605

--------------------------

F(x,y) = 0.03549964027552255 Such that X = 0.0022820950909634235,

Y = -0.011109339358273875

--------------------------

These were the minimas I got after running

the code for 100 times.

*Differential Evolution for Ackley*

F(x,y)= 1.04233523311

F(x,y)= 0.125506800724

F(x,y)= 0.118147806308

F(x,y)= 0.0435156543425

F(x,y)= 0.0382363979718

F(x,y)= 0.00617917091892

It is noticeable that the solution is more efficient and close to the global minimum point (0.0000 ).

* From the results we can see that differential evolution algorithm gives more precise results than the hill climbing algorithm regarding getting the global minimum for Ackley function. Actually lots of times I got the global minimum using DE which I have never got using the hill climbing algorithm.
* The number of initial generation I made was to be 10 in the Differential evolution part.