HW#1

Golnaz Mesbahi

40033628

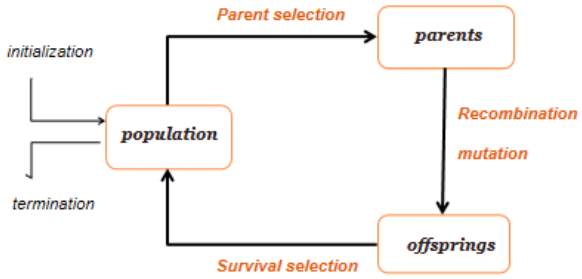
Problem description:

The goal is to place eight queens on a chessboard so that they cannot attack each other.

This implementation solves this problem using Genetic Algorithm.

This report describes the steps to this implementation and the results, and the effect of changing parameters such as population size on the algorithm:

General scheme:



Fitness evaluation:

This function assigns a value to each candidate solution indicating how close it is to the solution. **This is the basis for selection.** In the 8-queen problem, this function can be defined as follows:

the sum of penalties should be minimized to zero.

So, we define the goal to maximize 1/sum+1 (+1 is to avoid division by zero)

Algorithm description:

First, we initialize our population. (We show each candidate solution using a permutation of numbers 0 to 7, representing queens and their positions in each column.)

Then we choose five random chromosomes and choose the best two of them as parents. We apply cross-over to them and produce two children out of these two parents.

The cross-over is as follows:

Select a random position, and cut the parents into two parts in that position. Copy the first segment of parent one to child one and parent two to child two. Add values in parent two that is not in child one to child one and return to parent one for the rest—the same for child two.

After the cross-over is done, we add these two children and delete the two worst (with the least fitness score) out of the population to keep the size the same. We continue this algorithm until we find the solution (chromosome with fitness 1), or the maximum allowed generation level.

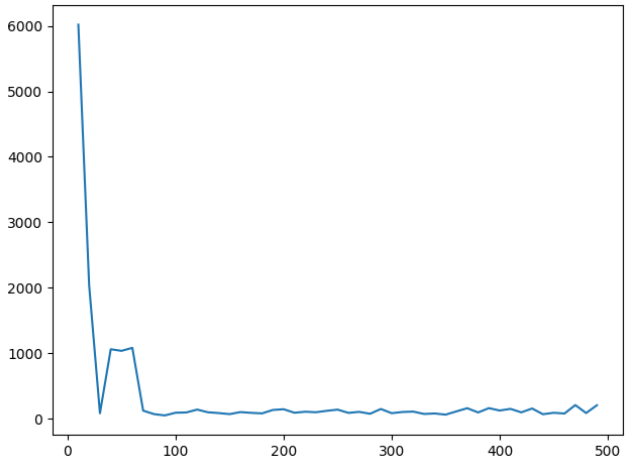
The general result for my algorithm is that it finds the solution after less than 100 generations.(for population size of 100)

Analysis of parameters:

Population size:

The plot below shows the effect of increasing the population size on the number of generations needed for the genetic algorithm to find the answer.

It shows that when the population size is small, the genetic algorithm has to go through many generations to find the solution. But as the size increases, it becomes easier for the algorithm to find the answer in fewer steps. The reason is that the larger the population size is, the easier it is to find the **optimal solution** because the initial generation is more likely to contain chromosomes closer to the optimal solution. (Or even the optimal solution itself.)



Mutation rate:

The higher the mutation rate, the more **randomness** is added to the algorithm. The job of this operator is to add **short search steps** in the search space.

Overall mean fitness score:

In order to show that the algorithm converges to the optimal solution, I have shown that the overall mean of fitness scores increases as the generations breed and go ahead.