CS 541: Artificial Intelligence, Winter 2023

Programming Assignment 2

Submitted by: Chitradevi Maruthavanan

PSU ID: 950828319

Goal: To implement a Genetic Algorithm to solve the 8 queens problem.

8 queens problem is where you place 8 queens on the chess board in such a way that each of the queen's moves don't conflict with one another.

Genetic algorithms use the concepts of biology where off springs are formed from two parents where the chromosome from each parent is crossed over and then sometimes the resulting offspring chromosome is mutated. There is a fitness component involved in the selection of the parent's chromosome and likewise we compute the fitness for the input sample and choose the sample with the best fitness before we do the cross over and mutation to get to the output state.

Python program for the 8 Queens GA:

The class is created to define each of the input sample (board configuration) for the problem. The objects of this class are different inputs for the GA and this has the fitness function computed for each input board configuration. An example input sample on the chess board with a specific configuration of the eight queens is [3,5,7,1,8,2,6,4]. Goal is to reach a conflict free arrangement like [8,2,5,3,1,7,4,6]. **Fitness** function for the 8 queens problem has a maximum value of 28. The number of pairs of non-attacking queens in an input sample determines its fitness. For a successful configuration like the [8,2,5,3,1,7,4,6] the fitness value is 28.

Fitness Formula = (no. of queens * (no. of queens - 1))/2 = ((8 * 7)/2)] = 28

Selection probability is calculated as individual fitness for the input configuration divided by the sum of fitness of all individuals in the input population.

The initial population is generated by creating new objects with a random number ordering between 1 and 8. Each newly produced object is added to the array. The selection probability value of each person in the population is then calculated and recorded.

Selection of Parents: The rotate() function is used to select the two parents in each iteration.

Crossover: For each pair of parents, the crossover point is produced at random. The two children are formed at the crossover point by joining the parent arrays.

Mutation: In a call to the mutation() method, each child is passed as an argument. This function generates a random number, and the mutation occurs if the number is less than the globally set parameter MutationPct. The integer is replaced with a randomly generated integer between 1 and 8 at a random index in the child array.

Inference:

1. Increased population size helps to the resolve to a solution faster. To compensate for the reduced population size increasing the number of iterations helped the program to resolve to solution.
2. The program behavior with random cross over points vs fixed cross over points showed varying results in the lower population size case. At higher population cases the results were not very different.
3. Increasing the mutation parameter Pct delayed/prevented the chance of resolving for all population sizes tried 1000, 500 and 100.

Steps to run the program:

Execute the attached python program and change below parameters as needed.

```
#MutationPct any value between 0.0 and 1.0
MutationPct = 0.9
#PopulationSize
PopulationSize = 1000
#Number of Iterations
NumIterations = 200
```

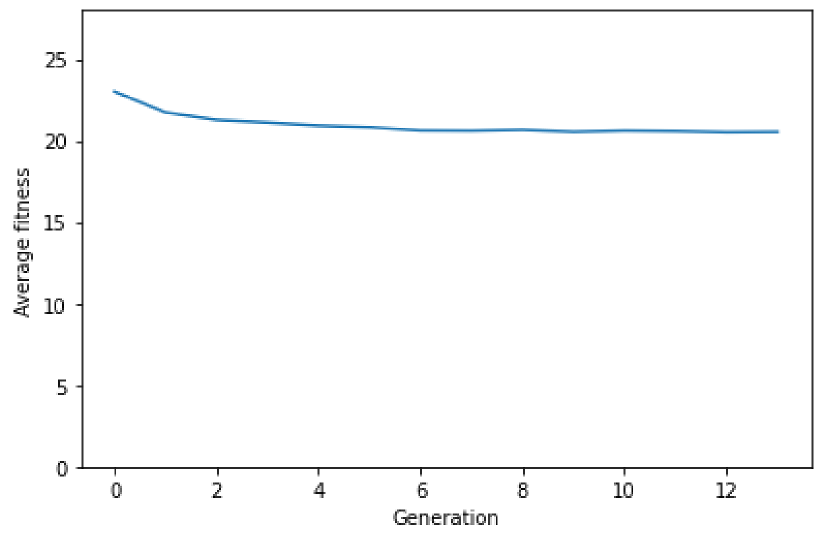
Results:

Example 1:

PopulationSize = 1000

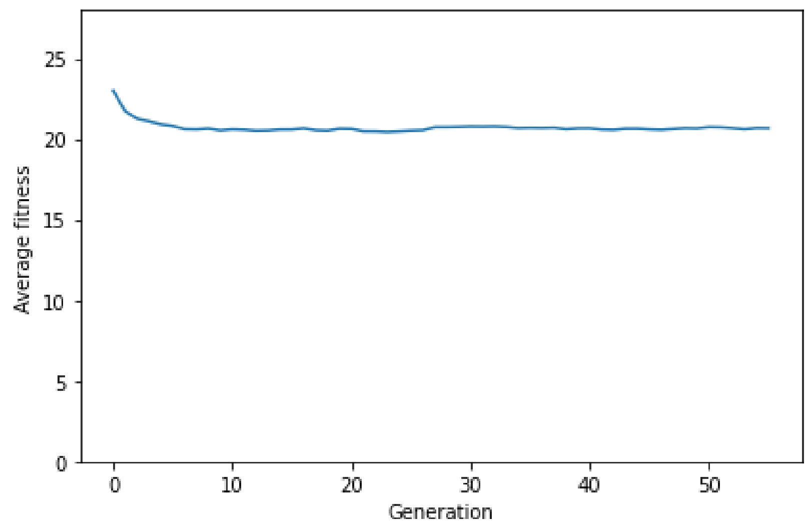
NumIterations = 200

```
Resolved: [5, 7, 2, 4, 8, 1, 3, 6]
Iteration resolved = 13
```

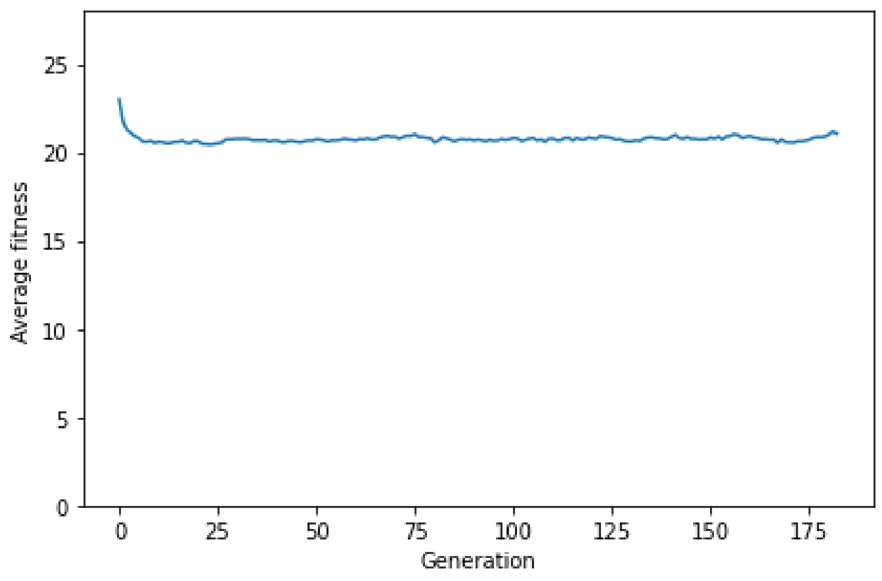


Resolved: [6, 2, 7, 1, 4, 8, 5, 3]

Iteration resolved = 55



Resolved: 4, 6, 8, 3, 1, 7, 5, 2]
Iteration resolved = 182



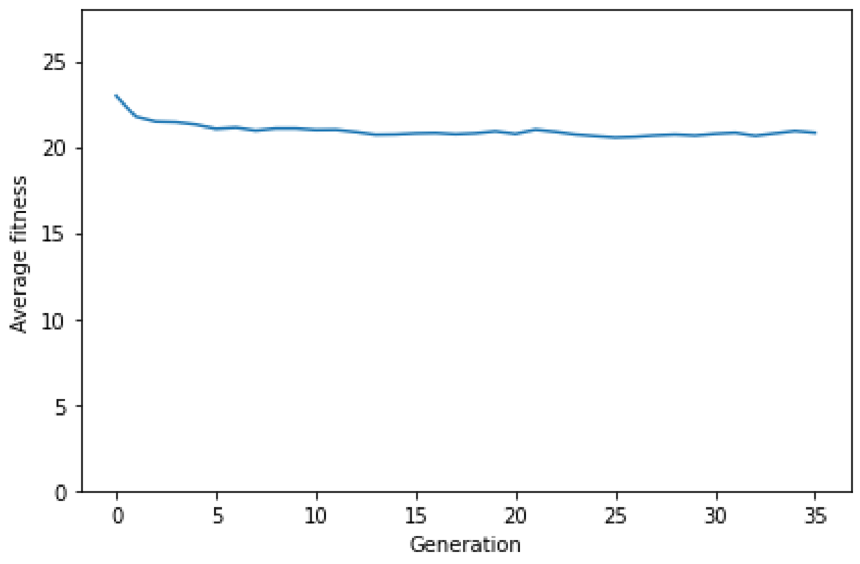
<u>Example 2:</u>

PopulationSize = 500

NumIterations = 200

Iteration resolved = 35

Resolved: [5, 8, 4, 1, 7, 2, 6, 3]



<u>Example 3:</u>

PopulationSize = 100

NumIterations = 200

Not resolved within the 200 iterations.

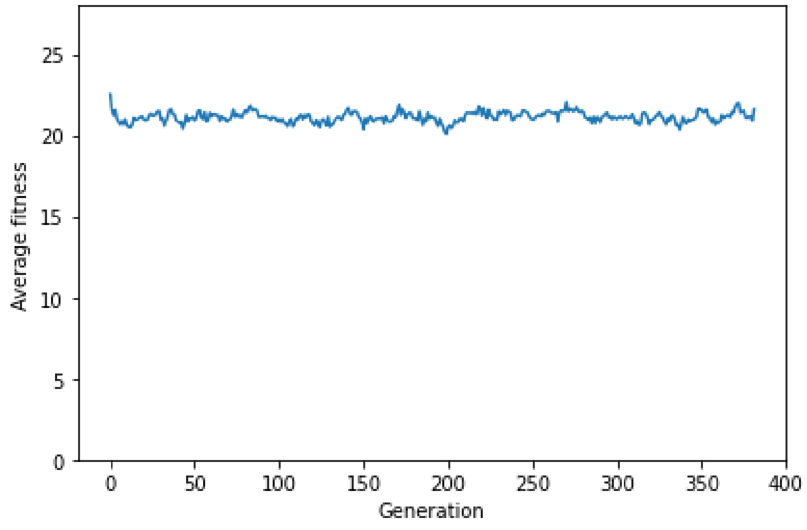
<u>Example 4:</u>

PopulationSize = 100

NumIterations = 600

Iteration resolved = 383

Resolved: [6, 1, 5, 2, 8, 3, 7, 4]



<u>Example 5:</u>

PopulationSize = 100

NumIterations = 600

Cross over = 4

Not resolved

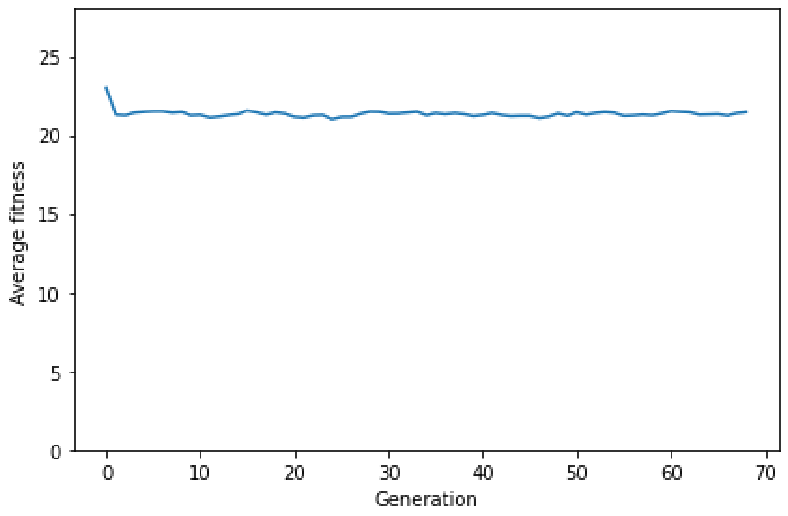
<u>Example 6:</u>

PopulationSize = 500

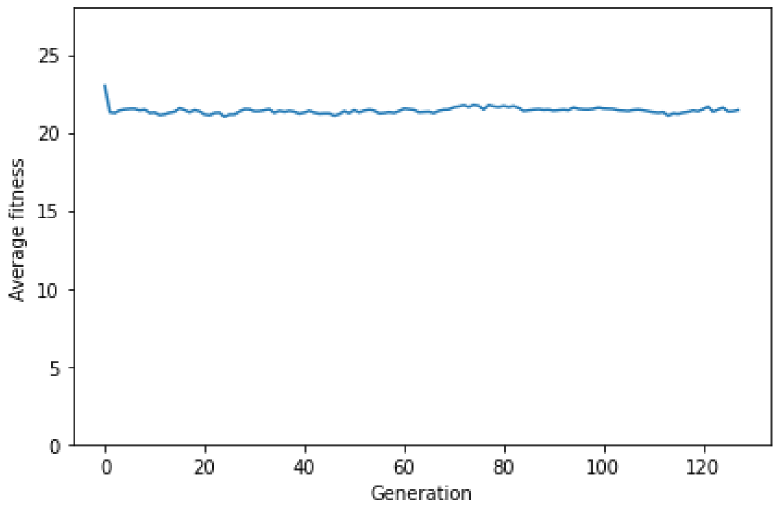
NumIterations = 200

Cross over = 4

Resolved: [4, 1, 5, 8, 2, 7, 3, 6]



Resolved: [3, 6, 2, 7, 1, 4, 8, 5]


Average fitness
Generation