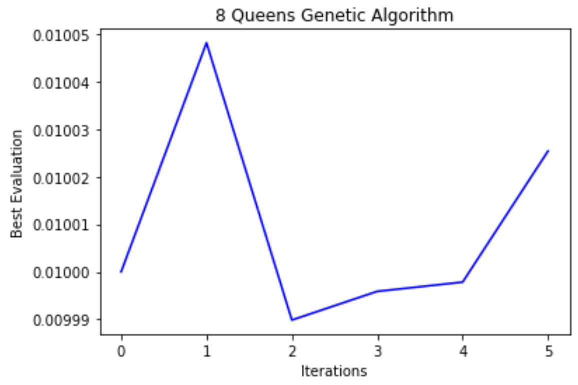
Keaton Spiller
CS 441
Spring 2022
Programming Assignment #2

8-Queens Problem

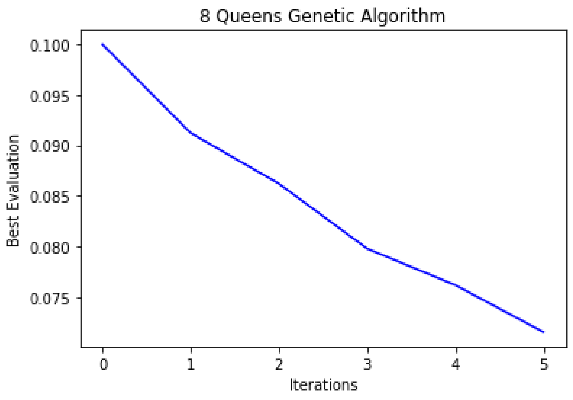
Initially I started off creating “PopulationSize” number of random states of queens, each one in a column indicated by the index and the value holding the row. Incorrect queens are diagonal to each other or horizontally intersecting. Since we have every queen in their own column we don’t have vertical intersections. The fitness function works by scoring which pairs of queens make a wrong move for each state. The states with higher scores have larger intersecting pairs of queens in incorrect positions and the states with lower scores have a better chance of being picked for creating offspring.

Each score is normalized with respect to the entire population to ensure each score is represented by the entire population. Two parents from each generation are chosen with the 2 best scores and create offspring split from a random index. One value is chosen from 1-7, this value indicates the range to slice the parents. For instance, if the value 1 was randomly selected the slice would happen between 0 and 1, and if 7 was selected between 6 and 7. The higher the population chosen the better the evaluation, showing the incorrect avg states of the population significantly decreasing over iterations.

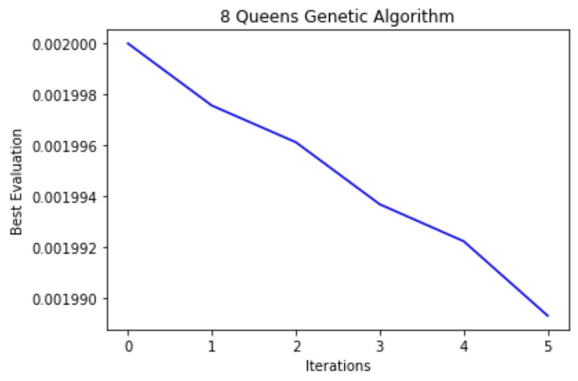
For mutation I decided to stick to 5% mutate rate for the entirety of the project.



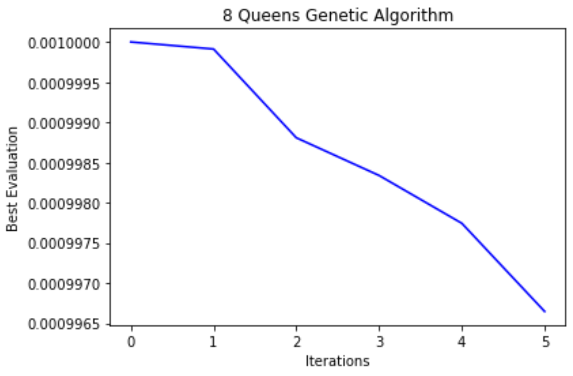
Population = 10, Iterations = 5, Mutation = 0.05



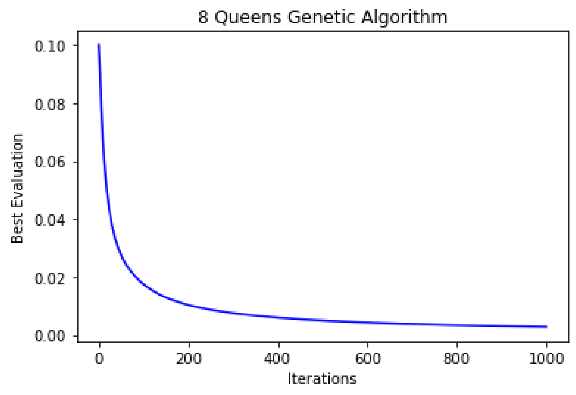
Population = 100, Iterations = 5, Mutation = 0.05



Population = 500, Iterations = 5, Mutation = 0.05

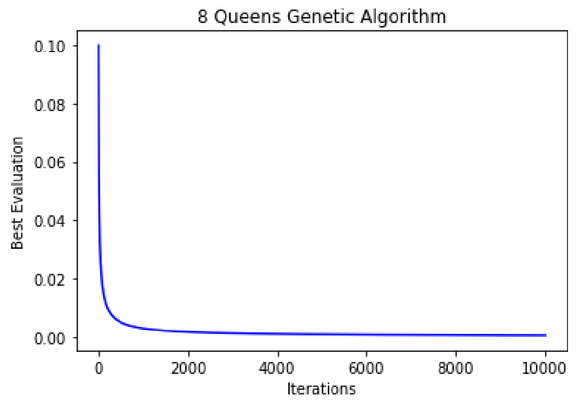


Population = 1000, Iterations = 5, Mutation = 0.05



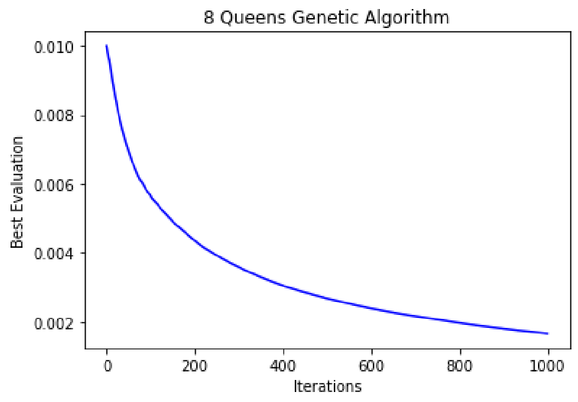
Population = 10, Iterations = 1000, Mutation = 0.05

```
Initial parents: [1 6 1 5 4 1 4 2] [4 4 2 4 4 6 5 0]
random samples from inital population:
[4 6 3 4 2 2 1 2] [0 0 0 7 4 2 2 0] [2 3 6 5 6 4 6 6]

Final parents: [2 0 7 0 4 6 1 3] [6 2 0 5 1 3 5 2]
random samples from final population:
[6 6 0 5 1 1 4 2] [6 6 0 5 1 6 1 3] [2 0 7 0 4 6 1 3]
```



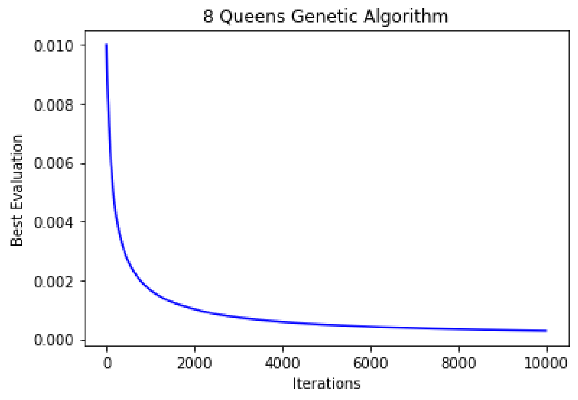
Population = 10, Iterations = 10000, Mutation = 0.05

```
initial parents: [7 7 4 5 1 4 6 1] [3 3 5 7 5 6 7 4]
random samples from initial population:
[7 1 4 6 2 0 0 2] [6 5 1 6 7 1 0 0] [5 3 2 1 6 1 3 7]

Final parents: [3 5 7 1 6 0 2 4] [6 4 0 7 5 2 5 1]
random samples from final population:
[5 1 4 7 6 5 0 7] [7 5 1 6 6 0 2 1] [6 4 0 7 6 0 2 4]
```



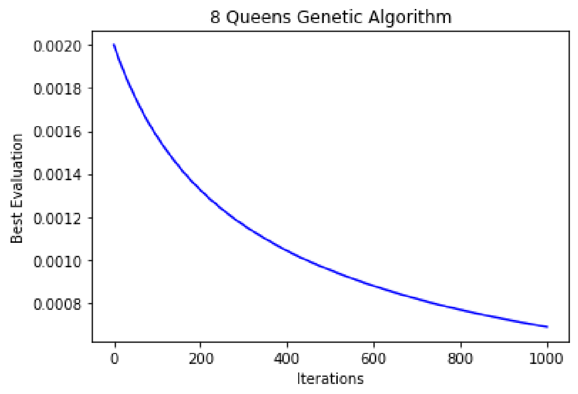
Population = 100, Iterations = 1000, Mutation = 0.05

```
Initial parents: [6 1 7 5 3 6 2 7] [2 5 1 0 0 3 4 1]
random samples from initial population:
[2 3 3 2 7 4 3 3] [5 0 0 1 1 6 4 5] [2 3 3 7 2 7 0 7]

Final parents: [2 5 1 4 7 3 6 3] [2 5 1 0 0 3 4 7]
random sample from final population:
[2 5 1 0 0 7 6 3] [2 5 1 0 0 3 4 7] [7 4 0 2 0 3 4 7]
```



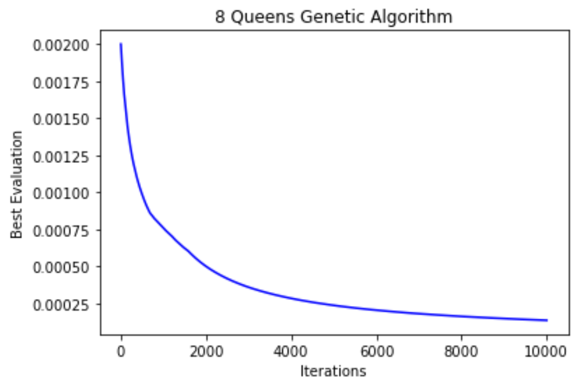
Population = 100, Iterations = 10000, Mutation = 0.05

```
initial parents: [7 3 1 7 0 6 6 4] [7 4 1 6 5 5 0 7]
random samples from initial population:
[0 1 7 1 3 4 6 7] [2 3 6 7 4 6 0 3] [3 2 2 7 1 2 4 4]

Final parents: [3 1 7 5 0 6 4 2] [5 0 1 6 2 5 7 4]
random samples from final population:
[7 0 1 6 2 5 2 0] [7 0 7 5 1 6 4 0] [3 1 7 6 2 5 2 4]
```



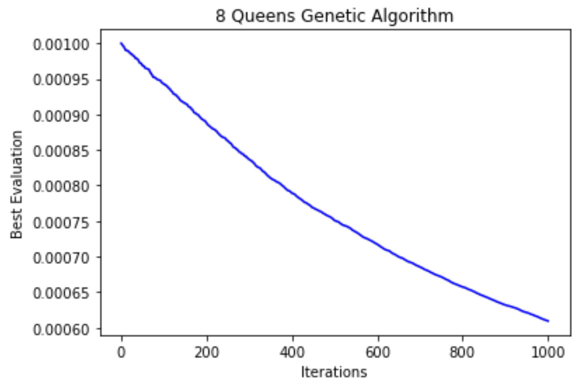
Population = 500, Iterations = 1000, Mutation = 0.05

```
initial parents: [5 3 0 4 1 7 2 6] [2 7 2 7 5 1 6 4]
random samples from inital population:
[2 7 6 7 4 7 6 1] [7 5 1 4 4 2 7 0] [2 2 1 4 7 2 2 6]

Final parents: [5 3 0 7 5 1 6 4] [2 7 2 7 5 1 6 4]
random samples from final population:
[5 3 7 1 6 1 0 6] [2 7 2 7 5 1 6 4] [5 3 0 7 5 1 6 4]
```



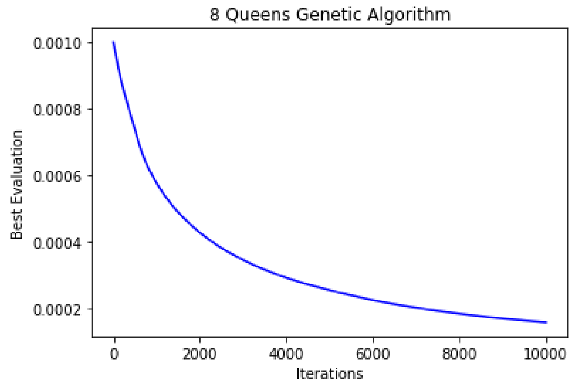
Population = 500, Iterations = 10000, Mutation = 0.05

```
initial parents: [0 3 6 0 7 1 4 5] [4 5 1 6 2 7 1 3]
random samples from inital population:
[7 7 1 0 3 3 6 0] [3 6 6 1 4 3 4 5] [6 2 3 5 2 1 7 3]

Final parents: [5 3 6 0 7 1 4 2] [0 3 6 0 7 1 4 2]
random samples from final population:
[5 3 6 0 7 1 4 2] [5 3 6 0 7 1 4 2] [5 3 6 0 7 1 4 2]
```



Population = 1000, Iterations = 1000, Mutation = 0.05

```
initial parents: [3 5 3 1 7 4 6 0] [2 5 1 4 1 0 4 7]
random samples from inital population:
[2 3 7 1 0 2 3 7] [3 3 6 4 4 6 7 6] [7 3 6 1 5 2 6 7]

Final parents: [3 5 3 1 7 4 6 0] [0 4 6 6 2 7 5 3]
random samples from final population:
[3 6 5 7 4 4 0 0] [7 3 6 2 0 4 2 1] [0 4 6 6 2 7 5 0]
```



Population = 1000, Iterations = 10000, Mutation = 0.05

```
initial parents: [4 1 3 0 2 3 5 6] [7 4 1 5 7 0 3 6]
random samples from inital population:
[7 0 1 0 1 7 5 1] [6 2 2 5 6 7 7 7] [4 0 4 0 6 7 6 2]

Final parents: [4 1 3 5 7 2 0 6] [7 3 5 0 4 6 4 2]
random samples from final population:
[6 3 5 0 4 2 0 6] [4 1 3 5 7 6 4 2] [1 3 5 5 7 2 0 6]
```

Looking at parent1 [4 1 3 5 7 2 0 6] we have a correct solution, yet looking at one of the random chosen [6 3 5 0 4 2 0 6] there are two incorrect rows.
Out of the population of correct states, this is very unlucky.