Mini-Project 4: Solving 8-queens Problem

Ronngguang Ou (219817313)
John Leonardo (216649967)
Manuel Herrera (219721880)
CSC 180 Intelligent Systems
Mini-Project 4

Problem Statement

The n-queens problem is a mid-1800s puzzle in which you place n queens on an nxn chessboard such that no two queens attack each other. Our goal for this project is to solve an 8 queens puzzle on an 8x8 chessboard using the Distributed Evolutionary Algorithm (DEAP) in Python.

Methodology

The Jupyter notebook provided to us had the position-index and row-index based board representation for us. Position-index, a board where each position is represented as an integer from 0 to 63 and one integer reserved for the queen's position. Row-index, where each row is indexed from 0 to 7 and a queen is placed on different rows top-down.

Position-index

```
[61, 5, 47, 2, 43, 54, 11, 53]
-|-|X|-|-|X|-|-|
---------------
-|-|-|X|-|-|-|-|
---------------
-|-|-|-|-|-|-|-|
---------------
-|-|-|-|-|-|-|-|
---------------
-|-|-|-|-|-|-|-|
---------------
-|-|-|X|-|-|-|X|
---------------
-|-|-|-|-|X|X|-|
---------------
-|-|-|-|-|X|-|-|
---------------
```

Row-Index

```
[0, 4, 7, 7, 7, 1, 7, 5]
X|-|-|-|-|-|-|-|
---------------
-|-|-|-|X|-|-|-|
---------------
-|-|-|-|-|-|-|X|
---------------
-|-|-|-|-|-|-|X|
---------------
-|-|-|-|-|-|-|X|
---------------
-|X|-|-|-|-|-|-|
---------------
-|-|-|-|-|-|-|X|
---------------
-|-|-|-|-|X|-|-|
---------------
```

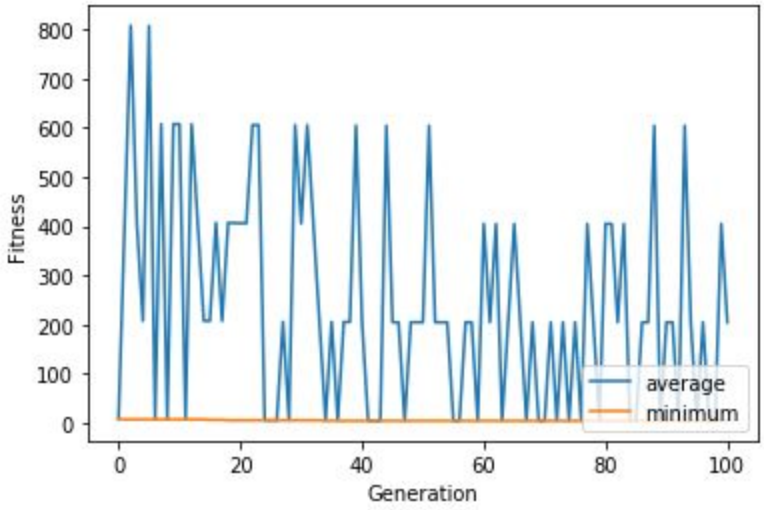
The evaFitness function for our position-index board returned the fitness of any given board. We created two arrays to store our x and y coordinates. We looped through the rows to detect conflicts (2 elements within the same row or column) and if non detected, we calculated the scope to detect any additional conflicts. The checkDuplicate function calculated the number of queen pairs in the same position for any given board. Furthermore, we created our first generation with 50 individuals and a hall of fame

variable to store the best one. We ran the eaSimple() genetic algorithm for 100 generations.

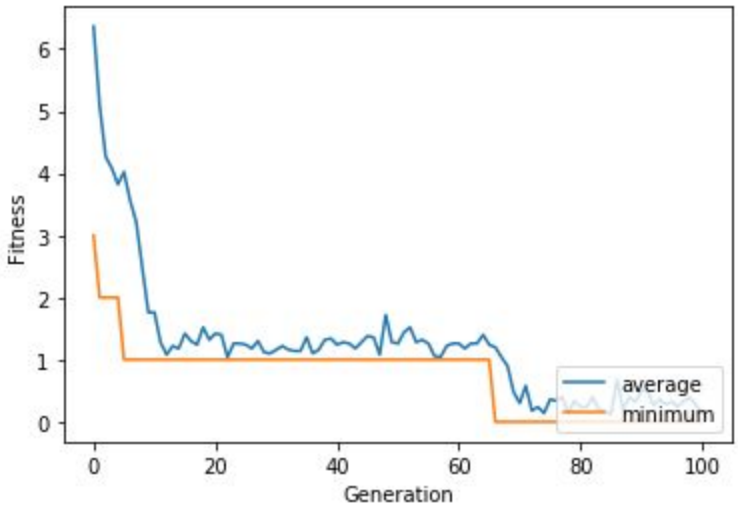
For our row-index board, evaFitness returned the fitness of any given board by calculating the total number of distinct pairs of queens that attack each other. We did the same setup as we did for our position-index board: 50 individuals for the first generation, a hall of fame for the best, and eaSimple() call for 100 generations.

Experimental Results and Analysis

After our experiment, the results in visual form can be seen below.




```
Best individual is:  [38, 43, 0, 28, 55, 17, 58, 13]  with fitness:  (4.0,)
X|-|-|-|-|-|-|-|
---------------
-|-|-|-|-|X|-|-|
---------------
-|X|-|-|-|-|-|-|
---------------
-|-|-|-|X|-|-|-|
---------------
-|-|-|-|-|-|X|-|
---------------
-|-|-|X|-|-|-|-|
---------------
-|-|-|-|-|-|-|X|
---------------
-|-|X|-|-|-|-|-|
---------------
```




```
Best individual is:  [4, 6, 0, 2, 7, 5, 3, 1]  with fitness:  (0.0,)
-|-|-|-|X|-|-|-|
---------------
-|-|-|-|-|-|X|-|
---------------
X|-|-|-|-|-|-|-|
---------------
-|-|X|-|-|-|-|-|
---------------
-|-|-|-|-|-|-|X|
---------------
-|-|-|-|-|X|-|-|
---------------
-|-|-|X|-|-|-|-|
---------------
-|X|-|-|-|-|-|-|
---------------
```

The conclusion is that after running eaSimple for 100 generations, the average values were significantly tightened. As opposed to the first generation, where the average fitness values had a really intense spread. After evaluating fitness and running genetic algorithms, we were satisfied with the best individual, which had significantly lower loss which led to better results, and move predictions.

Task Division and Project Reflection

Task division for this project were assigned as follows, Rongguang handled the fitness function of the position-index board representation and its eaSimple() call to find the

best. John and Manuel handled the row-index board fitness function and its eaSimple() call.

Challenges encountered throughout the project were only making the algorithms for finding the conflicts.