Kai Kit Lok (Brian) - 301108938
Kevin Estrada - 216179224
(Team Lead)Cody Wuco -301090621

CSC 180.02 - Intelligent System

Project 4 : Solving N-Queens problem using Genetic algorithm

Due Date: November 18

Problem Statement

In this project, we aim to practice genetic algorithm by using Distributed Evolutionary Algorithm in python or DEAP. In chess, a queen is the only piece that can attack in any direction on the chess board. The puzzle is to place a number of queens on a board in such a way that no queen is attacking any other.

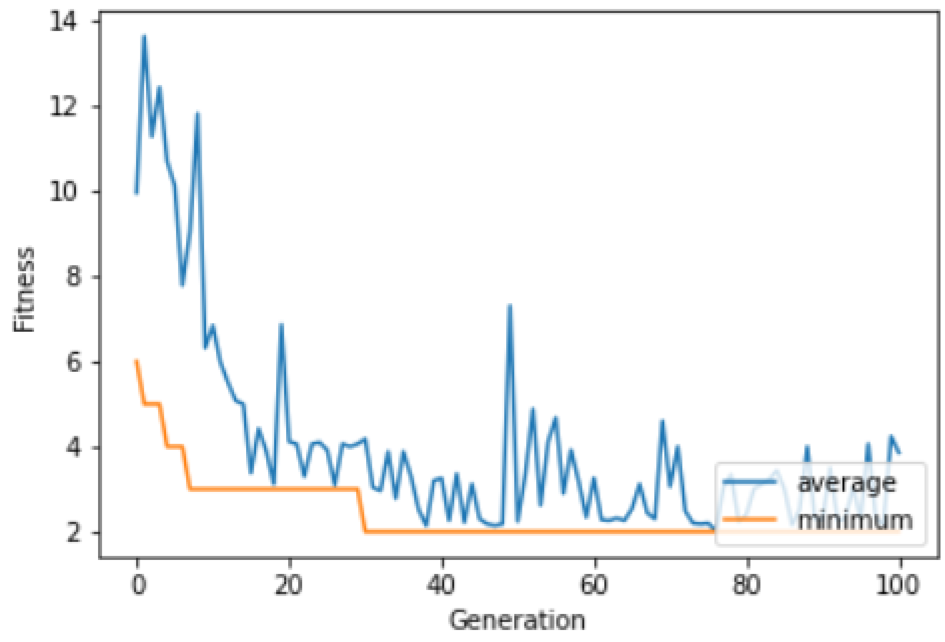
Methodology

In this project, we use DEAP to solve 8 queens puzzle. We would be placing eight queens on a 8x8 chess board so that no two queens attack each other. We would represent the board using a position-indexed-based on a 8x8 board, the position will be represented as an integer from 0 to 63. We will also use row-index-based which will represent the board indexed from 0 to 7. Queens are placed in different rows from top to bottom. The sequence [a b c d...] meaning that in the 0th row, a-th column, the queen is present and so on.

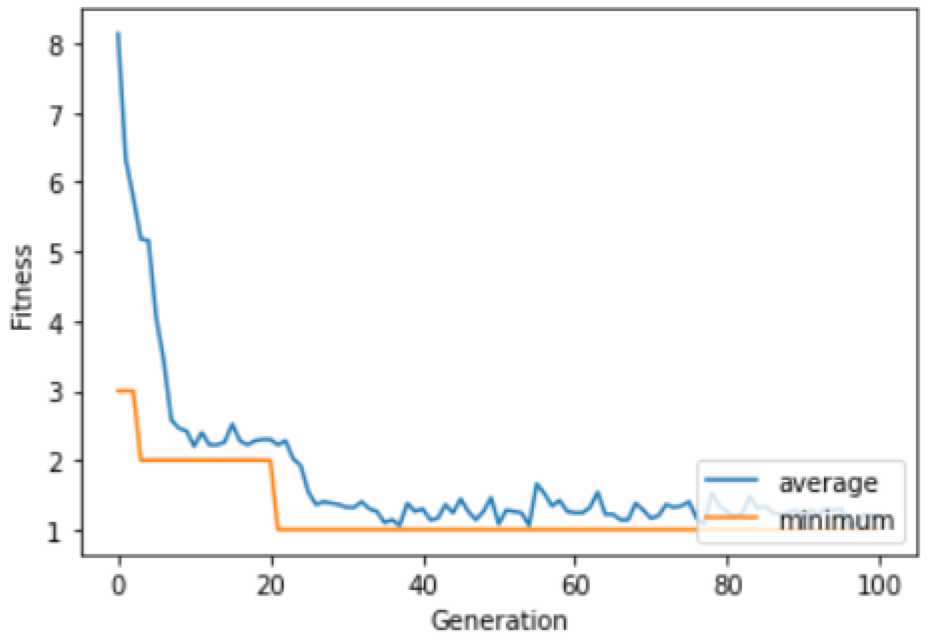
Tools: deap, numpy, matplotlib, tensorflow, Jupyter notebook, google colab

Experimental Results and Analysis

Position-index-based



Row-index-based



Both models were able to achieve the same minimum results, but the position-index-based model usually took longer, more runs, or just varied more wildly than the row-index-based model, and that was likely due to the fact that it had more freedom in where it could place the queens and a more complex evaluation function.

We also found that increasing the mutation rates in the case of the position-index-based model caused it to progress much faster, but vary more wildly in the average, which is great for producing a minimal result quickly, but the overall population seems to suffer.

Task Division and Project Reflection

(Brian) Was in charge of the initial pass on the code design.
(Kevin) Was in charge of code review and writing the report.
(Cody) Was in charge of evaFitness function and code evaluation form.

The project was quite small, so we handled it similar to last time. (Brian) was the first one to get started on the project. He filled in each section with the proper code and got the project working. (Brian) got everything working at a basic level, so he could make sure that all the code he wrote was correct.Then (Cody) went through the project afterwards to separate the code into the proper sections, finish the check and finish the evaFitness functions, and made sure that all of the code complied with the evaluation form. Then (Kevin) went through the code to make sure that there were no mistakes in the design or implementation. Then he wrote up the report, and everyone gave their input and looked it over.

This project was quite easy. The way Brian handled his initial pass was great. It allowed us to divide the work pretty efficiently. Cody really likes to work on details, so having mostly working code let him play to his strengths, and Kevin was able to keep us working at a pretty fast pace by giving us the room to make mistakes, because we knew he would comb through the code and fix any that we made.

Working on this project was really fun. We were surprised at how easy it was to work with GA's. Learning how different constraints, varying the rates of changes, and the weights can affect the progress of each generation was quite interesting, and if we were to revisit this in the future, learning to use this in a real time simulation could be an amazing learning experience.