Marlon Aquino

Professor Dominick Atanasio

CS 4200.01

15 March 2020

Overview

For this project, I have generated 401 cases for the N-Queens constraint satisfaction problem. This particular project was required to have the programmer to create a 25 by 25 board that has 25 queens on each column, and the objective is to keep each of the queens from attacking each other. I was required to implement the minimum conflicts algorithm. This algorithm chooses a random column of the board and finds the fitness score heuristic of how many queens are attacking in each of the coordinates of that random column. After you calculate the fitness score, the algorithm chooses the lowest score and moves the queen there and repeats the process of randomly choosing the column and calculating the fitness score until you find the solution of zero, which means that there are no more queens attacking each other. This algorithm goes through each column multiple times until a solution is found.

Approach

My approach to the project included implementing a method that places the queens in a random position and making sure that each queen is in a different column. I used an ArrayList that had an initial capacity of 25 to hold the positions of each queen. Calculating the fitness function was the same from the last project which returned the number of queens attacking from a single queen. The more queens that were attacking, the higher the score and the less queens that were attacking, the lower the score. The minConflicts method selects a column at random and moves the queen in the column that minimizes the number of queens attacking it. This repeats until the number of attacking queens is 0, or the method has taken the max number of steps which in this case is 5000.

Results

For three outputs, the average run times and percentage solved were as follows:

1. Output 1: 12s, 99.5%
2. Output 2: 11s, 100.0%
3. Output 3: 11s, 100.0%

Comparing this algorithm to the algorithms in Project 2, since I was not able to get the Genetic algorithm to work, I will be comparing the Min-Conflicts algorithm to the Simulated Annealing algorithm. The Simulated Annealing algorithm had the following runtimes and percentage solved:

1. Output 1: 16s, 87.5%
2. Output 2: 15s, 90.2%
3. Output 3: 31s, 85.78%

The Min-Conflicts algorithm was better in finding a solution and faster as well. I think the reason why the Min-Conflicts algorithm is more optimal is because it goes through each column multiple times until the solution is found.