Julio Berina

CS 4200

10/12/18

Project 2 Report

The goal of the second project is to solve the N-Queens Problem using two methods: Simulated Annealing and Genetic Algorithm. Simulated Annealing focuses on going from local extremas to find the solution with the smallest "temperature." The smallest temperature involves the solution. Genetic Algorithm has roots from biology since it models the N-Queens Problem genetically. This means that each possible configuration is considered an offspring, and it continually breeds two nodes with the lowest amount of collisions possible in a "population" of nodes.

My implementation of simulated annealing focuses on having a delta, which is the number of collisions of the current node - the number of collisions of a random child of the node. That random child is generated by making a random move based on the configuration of the current node. All this is executed until a node without collisions is obtained. I implemented Genetic Algorithm by using a priority queue in order to prioritize the nodes with the least amount of collisions. Until a solution is reached or the maximum amount of iterations (1,000,000) has been surpassed, I had the top two nodes in the population reproduce to create a child that potentially has less collisions.

Both algorithms are implemented separately as their own class. I implemented a Timer class and included a static instance of it in both SimulatedAnnealing and GeneticAlgorithm for testing purposes. Speaking of testing, I set up the code to where

simply running “java CS4200Project2” gives the final configuration for each algorithm. To actually run tests, the same command is ran with the argument “test”.

The test does 1000 test runs of NQueen instances per algorithm and solves them. My implementation of Genetic Algorithm was barely good enough since I was slightly above 94%, which is great for the purposes of this project. I had an average cost of 5790 for Simulated Annealing and 216979 for Genetic Algorithm. Overall, my algorithm implementations were pretty sufficient especially for the amount of instances I tested per algorithm. The screenshot below shows a test run that I did which represents my findings.