Project 2

CS 4200 - Artificial Intelligence

Instructor: Dominick A. Atanasio

Computer Science Department

California State Polytechnic University, Pomona

Report Submitted On: 08-MAR-2020

Report Submitted By:

Damian Ugalde

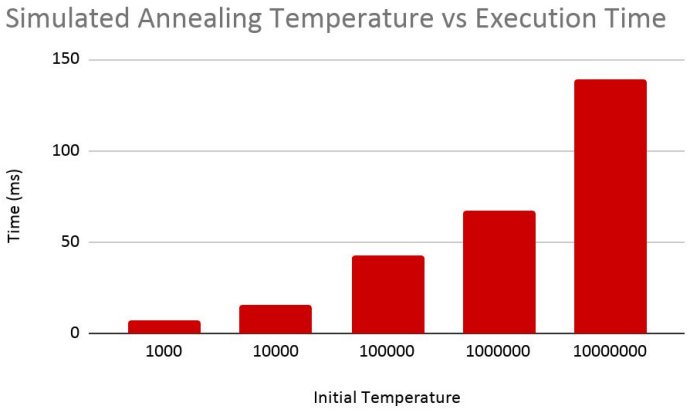
Approach:

To create a faster algorithm for the simulated annealing technique, this project constrained how the board was created and how the successors were calculated. Since two queens may not have the same row or column, the problem was constricted by only allowing one queen per row and one queen per column. The data is stored in an integer array, where the index of the array represents the column, and the value in the array represents the row. When the array is first created, each value is assigned a unique number from 0 to N. This guarantees that each queen has a different row and a different column. For calculating successors, two random indexes are found and their value swapped. This way, you preserve the different rows and columns while still calculating a successor that is one step away from the parent.

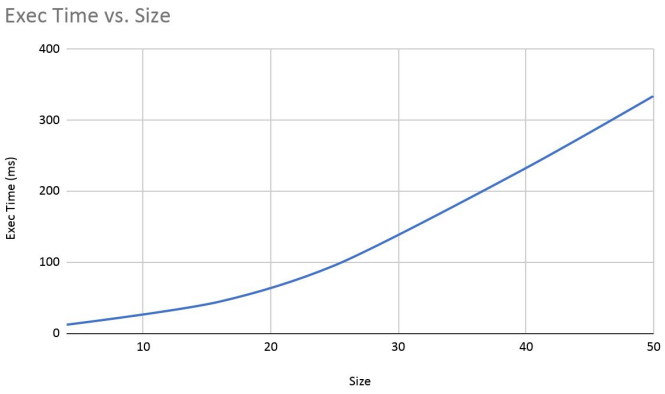
This approach however doesn't work for the genetic algorithm. It was hard to create a child board from two parents following a row-col constraint. How I solved it for this implementation, was that the initial value of the boards in the population was row-col constrained. This way, the population is closer to a solution from the beginning. For reproduction, the regular method of choosing a pivot and choosing all elements less than the pivot from parent A and all elements greater than the pivot from parent B. This didn't preserve row-col constraint, but it still allowed for the algorithm to continue without this added benefit.

Analysis:

Simulated Annealing:



Size	%success
4	100
8	100
16	100
25	100
40	99.25
50	98



Rate	% Success
0.01	26.25
0.001	69.5
0.0001	100
0.00001	100

- Genetic Algorithm

Top Population	% Success
0.05	35
0.1	44
0.2	54
0.3	60
0.4	51

Mutation Probability	% Success
0.05	21
0.1	48
0.2	78
0.25	76

Iterations	% Success
1000	17
2500	23
5000	50
7500	64
10000	67

Population Size	% Success
100	33.5
200	40.8
300	42
400	58
500	55
600	57

Findings:

For simulated annealing, the initial temperature didn't make a difference. After running the program 400 for each initial temperature of 1000, 10000, 100000, 1000000, and 10000000, all 2000 solutions found had 0 queens attacking for a 100% success rate. As the initial temperature increased, the amount of time needed to run each solution increased from 7 ms average at 1000, to 139 ms average at 10000000. As the size of the board increased, the time for execution increased exponentially. Also, running each puzzle size 400 times found a success rate of 100% for puzzles sizes less than or equal to 25. When the puzzle size is 50, the success rate drops to 98%. The rate of change for the cooling is the greatest variable for the performance of the algorithm. Testing each rate 400 times for values 0.01, 0.001, 0.0001, and 0.00001, it was found that 0.01 has a success rate of 26.25%. Lower than acceptable. The rate reaches 100% success rate at 0.0001.

For the genetic algorithm, the mutation probability that seemed to work the best was at 0.25, with a success rate of 76%. As the iterations increase, the probability of success also increased. As the initial population increased, the probability of success also increased. For the top population chosen, the success probability seemed to increase when 40% of the population was considered for the next iteration, compared to 5%.