Salvatore Grillo

N-Queen Project Report

My approach to this project was to initially use a stock model as provided by the pseudocode in our textbook in order to solve the problems. After reading through the simulated annealing version and implementing it, it worked great aside from trying to find a way to get temperature to zero and staying low enough to only accept good moves. However, most of the difficulty from this project, in my implementation, was genetic.

The way the book explained, and the easiest way forward was to calculate fitness for an initial population, and then calculate weighted probabilities assigned to each member of the population, with a high probability of getting selected based off of a higher fitness score. In this case, a higher fitness score was given to board states which had a higher number of non-attacking queens. From here, a weighted random selection was done and after two parents were selected, a child was created with some mixture of the two parents and a random mutation chance. However, this method did not work for me. It worked well at a low number, around n = 5 queens, however it capped out at around n = 10 or 11. From there I looked around online and in journals, eventually stumbling across what my final implementation would be, tournament style generic.

Tournament style genetic works as such: from initial population, randomly select k number of boards, pick the best from this and set as parent one. Do this again to get a second parent and create a child with a random chance of mutation. Repeat this process until a population of the same size as the initial is created and continue for a set number of generations. While a simple enough concept, it proved to be remarkable in results, and while my

implementation of such takes longer than simulated annealing, it proves to be more accurate, as shown from the data below.

N=25 Queens, 510 cases

	Simulated Annealing	Genetic
Success Rate	96.2	98.4
Average Runtime	0.069 Seconds	8.31 Seconds
Average Search Cost	20000	433790

From the above chart, we see that simulated Annealing is a fast algorithm that achieves a very respectable success rate in a fast amount of time. Compared to genetic, while it may be a bit less successful, it blows it out of the water in terms of time. As such, it is hard to recommend it in almost any case unless you need absolute precision. In that case, you can turn up the amount of generations of genetic to be higher and almost guarantee a success. The reason that genetic takes so long is that if you compare the search costs, genetic is higher by almost 21 times that of simulated annealing. This is due to the fact that the boards are randomly created and breed in a direction which can end up going the wrong way and doubling back several times. However, simulated annealing can mostly avoid this as it takes some bad moves but slowly begins to only take good ones which result in it reaching its target rather quickly.

Overall, from the results it is clear to see that simulated annealing is a powerful algorithm that can quickly accomplish its task quickly. While genetic may have an advantage in terms of success rate, that extra success rate comes at a cost of taking a lot more time. Furthermore, roulette style genetic, while a good idea, doesn’t panning out and shouldn’t for N-Queen.