Analysis of Genetic Algorithm for NQueens Problem

Saksham Goel
goelx029@umn.edu

October 16, 2018

1 NQueens Probblem: Size Growth Analysis

In this section we discuss the quality of results produced by Genetic Algorithm as the size of the NQueens Problem grows. The official aima-python repository on Github is used as the source code for the implementation of the Genetic Algorithm, along with their implementation of the NQueens Problem. The Genetic Algorithm is run on the NQueens Problem for different sizes and with different number of initial population sizes. The size of NQueens Problem was varied from **10** to **50**. For each of these sizes the algorithm is run with 4 different initial population sizes: **3**, **5**, **10**, **25**. The algorithm was run with the default parameter values for all except the number of generations which was changed to 5 to achieve results faster.

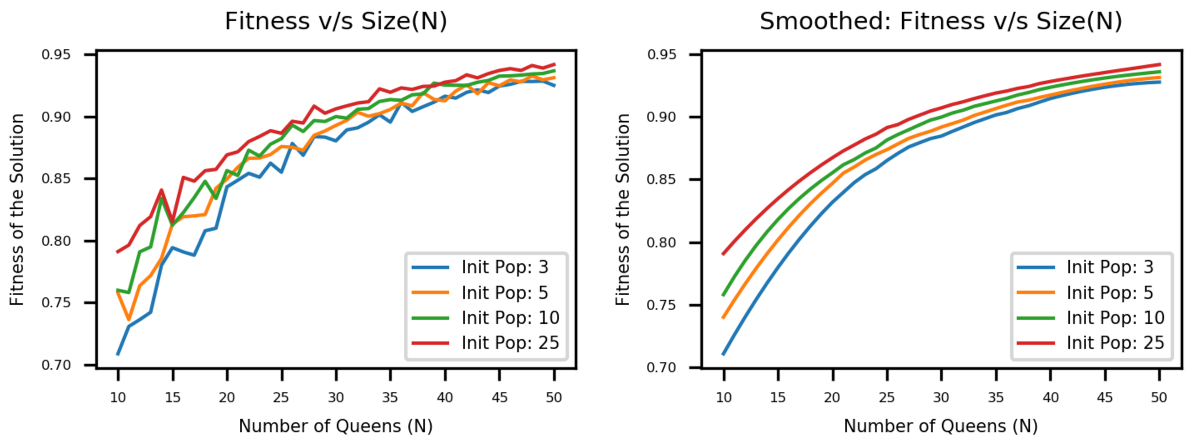

Figure 1: Plots displaying the Performance of Genetic Algorithms (Fitness Scores) with respect to the Size of the NQueens Problem.

After the experiment was run for all the values, they were collected and saved into a file and then read and plotted using Python libraries and the resulting plot can be seen in Figure 1 in Section 1. It is important to note that for determination of the quality of the results, the fitness value for the result state obtained by the genetic algorithm was used and then normalized, based on the highest fitness value corresponding to the particular size of the NQueens Problem. The figure contains two plots which display the results from the same experiment, except for the fact that the left plot uses the actual values, while the plot on the right uses a **"Savgol Filter"** to smooth the values so that a much more smoother curve can be obtained for easier analysis.

From the plots we can see a trend that the fitness values (normalized) are increasing and plateauing as the size of the NQueens Problem grows. This is not the kind of result I was expecting. According to me the curver should have gone downhill a little bit because as the size of the problem grows it is far less likely to have a more fit randomly initialized individual. Because the number of generations has stayed constant, the overall fitness after that many generations should be less for the higher size problems, however the results were completely opposite. The only reason I expect this to be true, is because when we have a large size for the problem, there is that many more number of genes, hence even when we pick random numbers the number of collissions on rows and columns decreases. Also the significance of diagonals decreases when the size of the problem increases, because there is that many more places to put the queen in such that it does not attack another queen nearby based on diagonal.

However one important property that this graph depicts is that it shows the expected results when we vary the Size of the Initial Population parameter. We can easily see the trend in the right plot that on average the fitness of the final solution of the genetic algorithm is better when the size of the initial population increases. We can see that the difference in the fitness values is very profound for the lower problem sizes while the difference decreases for the larger problem sizes. This could be argued because when the problem size is small the randomly generated population does not have that much genetic variation and the genetic variation increases as the initial population increases thus leading to better/fitter individuals, while with very big problem sizes there is already a lot of genetic variation that having larger population size doesnt affect much, but still has some effect.

2 NQueens Probblem: Runtime Analysis

For this section we are analyzing the runtime of the NQueens Problem using Genetic Algorithm with respect to two different initial parameters. For this experiment, the size of the NQueens Problem was fixed at N = 20, while two initial parameters: Initial Population Size and Number of Generations, were changed within a specific range of values and the runtime for the genetic algorithm for the NQueens Problem was collected. The initial population was ranged within the values: 3, 5, 10, 25, 50 The number of generations was ranged within the values: 5, 10, 25, 50, 100 For the other parameters, we used the default value of the function as specified in the search.py file in the aima-python github library.

For the results of the experiment: see Table 1. There also two other tables, see Table 2 and Table 3. These tables contain the aggregated values for the Time for particular values of either Initial Population or Number of Generations.

From the tables, it is easy to identify a pattern of steady increase of Time as the initial population size or the number of generations increase for the Genetic Algorithm. This kind of pattern is expected from genetic algorithm because both of these paramters are directly linked to how much processing needs to be done by Genetic Algorithm. If Number of Generations increase then Genetic Algorithm has to process that many more times, and similarly if the Initial Population Size increases then the Genetic Algorithm has to worry about that many more individuals. The relationship of both of these parameters with Genetic Algorithm is why we can see the increase trend in the time.

Table 1: Time (sec) for Genetic Algorithm to run 20Queens Problem

Pop	Gen	Time
3	5	0.008382
3	10	0.010984
3	25	0.031149
3	50	0.047293
3	100	0.086302
5	5	0.018961
5	10	0.026492
5	25	0.071403
5	50	0.125982
5	100	0.267516
10	5	0.053568
10	10	0.103829
10	25	0.269495
10	50	0.638193
10	100	1.313980
25	5	0.326115
25	10	0.666321
25	25	1.582868
25	50	3.212664
25	100	6.448169
50	5	1.273892
50	10	2.461460
50	25	6.217385
50	50	12.401954
50	100	26.398155

Table 2: Aggregated Time (sec) for Initial Population Size

Pop	Aggregated Time
3	0.184110
5	0.510355
10	2.379065
25	12.236137
50	48.752846

Table 3: Aggregated Time (sec) for Number of Generations

Gen	Aggregated Time
5	1.680919
10	3.269086
25	8.172300
50	16.426087
100	34.514122

The other important trend we can observe from the tables 2, 3 is that there is almost a linear relationship between the time for the genetic algorithm to compute results and the number of generations it is expected to run. This is true because in the implementation of the Genetic Algorithm it loops that many number of times, hence having an O(n) kind of complexity relationship. While there is polynomial (quadritic) relationship between the time for the genetic algorithm to compute results and the size of the initial population. This could be because for the Genetic Algorithm when selecting the individuals to mate the selection is done based on random selection over the probabilities distributed over the fitness of the different individuals. Because of this, the population needs to be probably sorted based on their fitness value, hence almost giving out a O(n^2) complexity.

These tables certainly provide a better and concrete way of thinking about the time complexity analysis of the Genetic Algorithm over the effect of change of the initial parameters like the Size of the Initial Population and the Number of Generations the algorithm is run.