EC Assignment 1 Part B

Sandhya Gopchandani

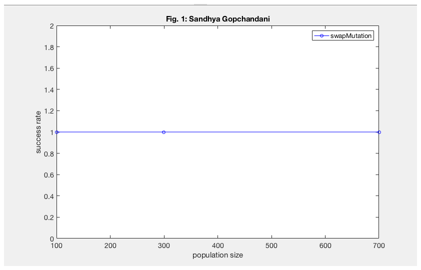
September 20, 2017

1 N-Queens Problem

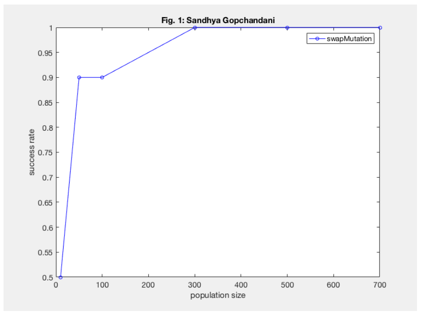
N-Queens problem is a problem of placing N queens such that no two queens attack each. The solution requires that no two queens share a row, a column or a diagonal. We are treating this problem as an optimization problem and trying to solve it using Genetic Algorithm using GA toolbox in MATLAB. We define Fitness as: minimum or zero number of clashes. This report comprises of all the experiments required to solve 16- Queens problem and later extending it to N-Queens.

2 Setting the Population Size

By setting up basic values in GA toolbox, our goal is to test different population sizes, find out the success rate of each population per x number of repetitions and see which population size has the maximum success rate. I have tried to see the impact of various population sizes over the increasing number of reps. The experiment shows that changing the number of reps results in different success rate for the same population. Following are the results of some of my trials.

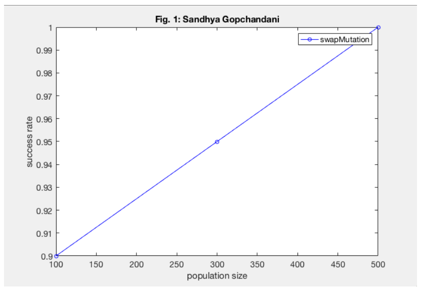


(a) fig:N=16,Generation=1000,reps=5

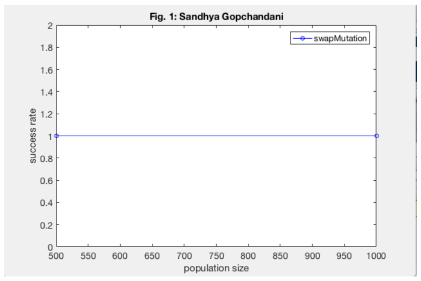


(b) fig:N=16,Generation=1000,reps=10

Figure 1: success rate at various population size over various reps



(a) fig:N=16,Generation=1000,reps=20



(b) fig:N=16,Generation=1000,reps=20

Figure 2: success rate at various population size over various reps

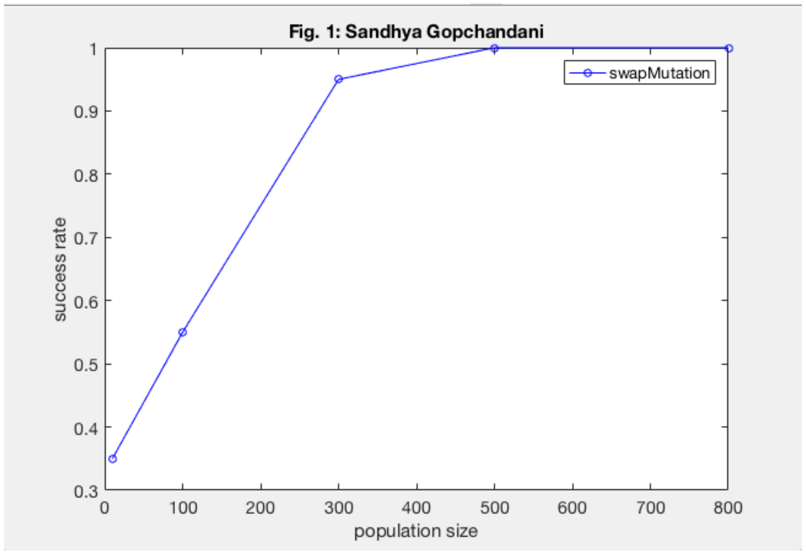


Figure 3: fig:N=16,Generation=100,reps=20

Looking at the figure 1a, it seems 100 is a good population size because we see 100 percent success rate but if we increase the number of reps as illustrated in the figure 1b, we see a dip in the ratio. Same case can be seen with population size of 300 in Figure 1b that shows that 300 is a good population size but as we increase the number of reps (fig 2a), we will see the decrease in success rate.Finally, As we increase the reps to 20, we see that 500 population size will give an optimal results for 16 queen problem(fig2 a and b). It is also shown that 500 can be a reliable population size even after significant low number of generations. Figure 3 shows that with generation 100 and reps 20, population size of 500 will result in 100 percent success rate.

3 Comparison between two mutation functions

I am defining the effectiveness of mutation operators based on these two conditions.

1. Number of the generations taken by two mutation operators to reach the best solution over some number of reps. The lower the number of generations is, the more effective mutation operator is.

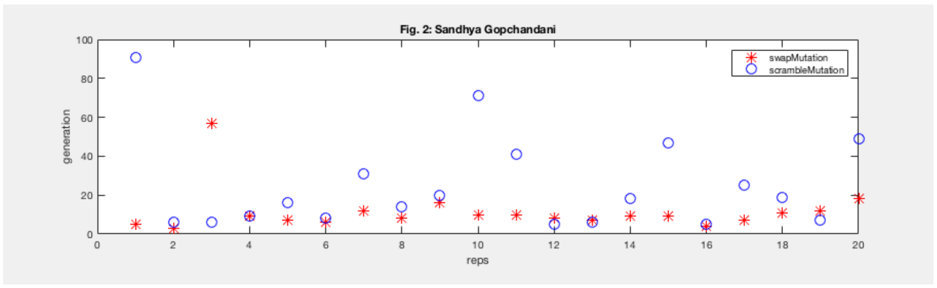
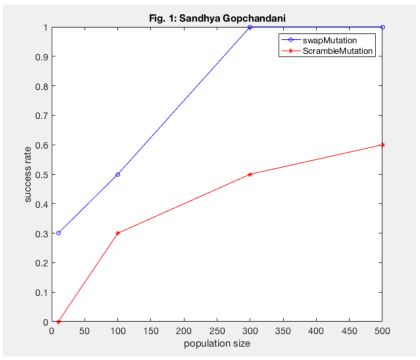
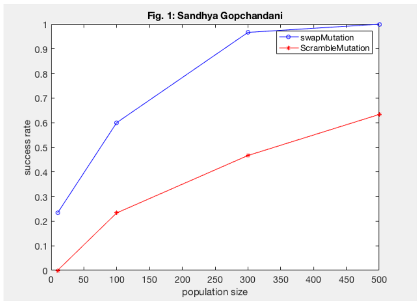


Figure 4: generation of the best solution for various reps at 500 popsize

2. Success rate of the two mutation operators at various population sizes. The greater the success rate is, the more effective mutation operator is. we can determine if one functions needs more population size to have a greater success rate.



(a) fig:N=16,Generation=100,reps=10



(b) fig:N=16,Generation=100,reps=30

Figure 5: success rate at various population size over reps

Analyzing the results from both criteria, it is apparent that Swap Mutation operator works better for 16-queen problem because it reaches the best solution in less number of generations and have greater success rate over the scramble mutation operator.

As answered in the Part A of the assignment, It was my prediction that Swap Mutation Operator should work better on Queen Problem because it introduces small change in the genome by swapping only two alleles. As discussed in the class, Swap operator disrupts the genome by breaking more adjacency and scramble mutation breaks the order. I believe, we need to maintain the order in Queens Problem hence, swap mutation operator seems to work better here.

4 Simple scaling study

As per my results, my algorithm setup to solve 16-queen problem is such that it has 100 generations, population size of 500 and a swap mutation operator. In order to see, how this algorithm is going to perform with N queens problems, I am starting off with finding Average Fitness Evaluation for 16-queen problem over 20 reps.

$$FitnessEvaluation = Populationsize * LastImprovementGeneration$$

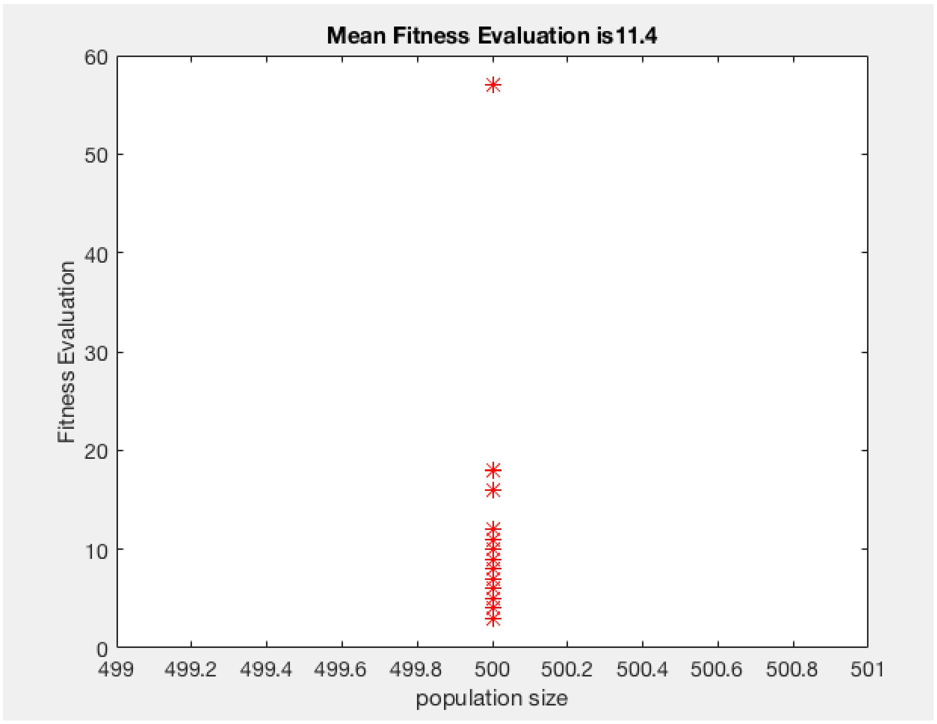


Figure 6: Avg Number of Fitness Evaluation for 16-queen problem over 20 reps

As per my understanding, Average number of Fitness Evaluation of 11.4 means that the current GA algorithm will find its best solution at about 11 generation if run for 20 reps. We want to extend this experiment to N-Queen problem. This means that we want to find the population size and max generations with the increase in problem size.

Keeping the same settings of the algorithm that is generations=100, popSize=500 and reps=20, I ran it for the various problem size to find out the number of fitness evaluation of each problem size.

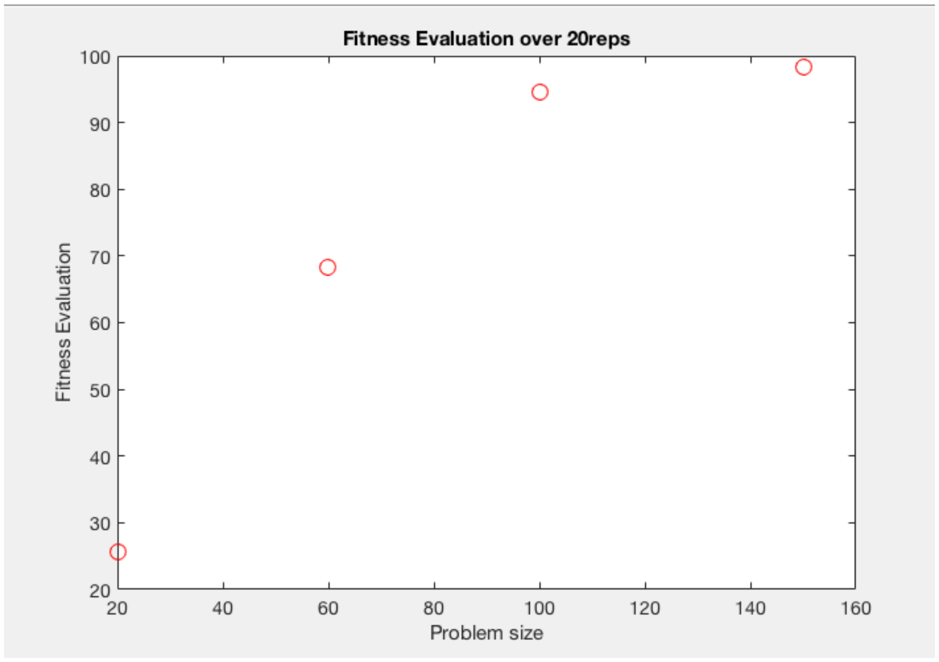


Figure 7: Avg Number of Fitness Evaluation for N-queen problem over 20 reps

It is apparent that as the N increases, the more generations will be required to solve the problem hence the number of evaluation fitness will increase with the increasing number of N.

The next step would be to actually run the GA algorithm for various problem sizes with different population sizes while keeping max generation at 100.

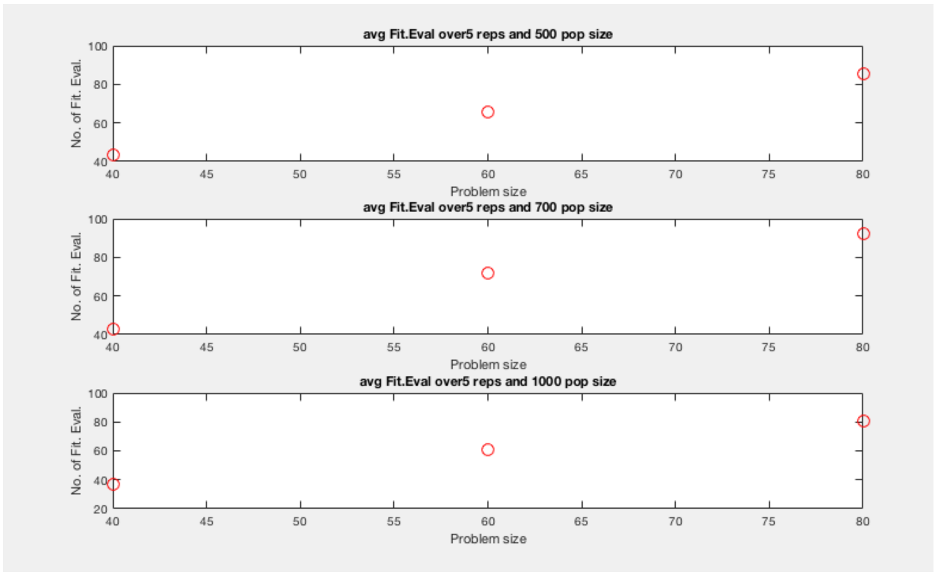


Figure 8: Avg Number of Fitness Evaluation for N-queen problem over 20 reps

The figure shows the number of average number of evaluation fitness for various problem size at various population size. It is clear from this picture that as the higher problem size will require more population size to get less number of fitness evaluation.

As we try out more experiments with by changing the values in GA algorithm i-e Generations, population size and number of reps, we will be able to find out the ideal population size for given N problem.