REPORT ON AI ASSIGNMENT- 1

Q1. *Use genetic algorithm as given in Figure 4.8 (Page 129) of the textbook to solve the 8-queens problem.*
Let, Fitness function = 1 + number of pairs of queens not in attacking position.
First, implement the version of the GA algorithm given in the textbook. Assume, that initially all the states in the population are the same and correspond to the eight queens being on the same row. (That is, initially every state in the population has a fitness value of 1.) For the first algorithm, let the population size be 20.
Next, come up with a variant of the GA algorithm so that the best fitness value in a population improves in a faster manner over successive generations. In the second algorithm, use the same initial population, the same state representation and the same fitness value function as used in the first algorithm. Make changes only to the other details of the GA algorithm (i.e. size of the population, Reproduce() function, M utate() function etc.).
Plot a graph that compares the first algorithm with the second algorithm. The y-axis of the graph must represent the best fitness value in a generation and the x-axis must represent the number of generations.
-
The population size is assumed to be 20.
The small random probability for mutation is taken as **0.03**.
The fitness is calculated by number of horizontal clashes and main and off diagonal clashes.
The probability for an entity is calculated by fitness(entity)/max_fitness(which is 29).

The basic version of Genetic Algorithm obtains maximum fitness after on average 5000-10000 generations.
The improved version of Genetic Algorithm does the same in 100-1000 generations and sometimes even less.
The improvement is based on improving reproduce function and doing double mutation based on small probability.
In the improvement, Crossover for reproduce function is by selecting a section of allele from first parent and other allele from second parent, which are not in already selected selection from first parent, and are in same order as in second parent.

Parent1: 5 2 3 1 6 4 8 7
Parent2: 1 8 6 4 7 5 3 2

Child: 8 7 3 1 6 4 5 2

We also do the following two modifications:

1. In the reproduce function, while selecting a random section out of the first parent, check if reversing the section can give better fitness value.
2. Modify the reproduce function to get the best child out of a number of children(here 10) produced from two chosen parents. This will get the best child out of the 10 children, which may or may not be better than the parents.

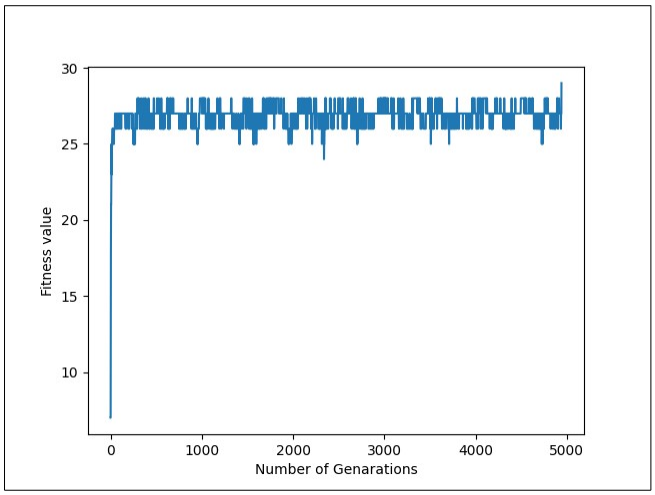
Double Mutation is randomly selecting 2 alleles and swapping them.
The probability for double mutation is kept at 0.02.

Both these combined makes genetic algorithm better.

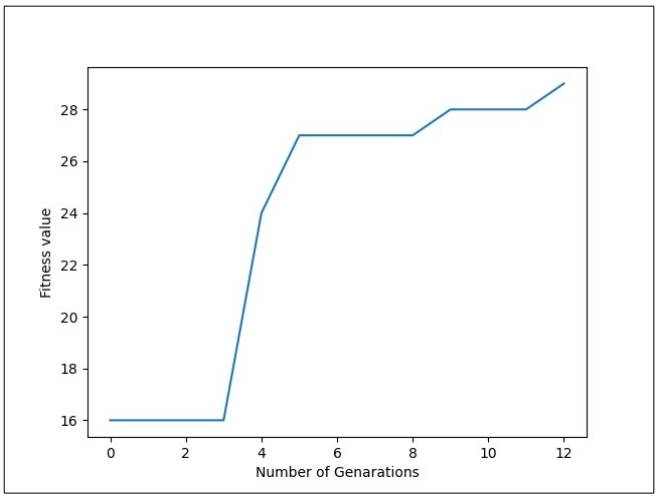
Using normal Genetic Algorithm, for one run we got optimum answer as [6, 3, 5, 8, 1, 4, 2, 7]. which took **140,000** iterations.

Meanwhile using Improved Genetic Algorithm for 8 queens,
On average for 100 runs, took **18.51** generations.
On minimum for 100 runs, took **8** generations.
On maximum for 100 runs, took **43** generations.

Using improved Genetic algorithm, for one run we got optimum answer in **14** generations only.
[5, 3, 0, 4, 7, 1, 6, 2]



Fitness vs Generations graphs for GA naive



Fitness vs Generations for GA optimized

Q2. *Use genetic algorithm to solve the travelling salesman problem (TSP). There are 14 cities labelled A to N.*

Here each city maps to an integer from 0 to 13.
We calculate the distance as we go from first city to the last, and the return distance is implicit.

The fitness function is taken as 10^6 divided bt the distance calculated from starting city to last city (1000000/distance from Start to End).

Here crossover function must be carefully chosen so that we dont obtain invalid states. So we select a subset of first parent. Then remaining elements are taken from second parent. While taking elements from the second parent, it is ensured that the elements appear in the same order as they did in the second part. Also it is ensured that no elements gets duplicated. This is similar to the optimized crossover we did in previous question.

Also in the mutation function, make sure to swap any two selected positions rather than modifying any position. Not doing this can produce an invalid state.

Using Normal genetic algorithm, it took a lot of generations to reach a good answer, which is still not the best answer.

Fitness started to become constant after **45785** generations
Best fitness value: 287.35632183908046
Distance covered: **3480**.0
Best entity: [6, 0, 11, 12, 10, 3, 5, 2, 4, 8, 1, 7, 13, 9]

As an optimization, we do the following,

1. Introduce double Mutation at probabilty 0.02.

2. Modify the reproduce function to get the best child out of a number of children(here 10) produced from two chosen parents. This will get the best child out of the 10 children, which may or may not be better than the parents. This is a local search optimization to the reproduction.

3. In the reproduce function, while selecting a random section out of the first parent, check if reversing the section can give better fitness value.

The same modification was also done in the first question.

It turns out these last two improvements(specially the last one) reduces number of generations taken to reach optimal state drastically.
It even gives the best possible answer at **3159.99 km**, in just around 100 generations. The optimal path turns out to be:

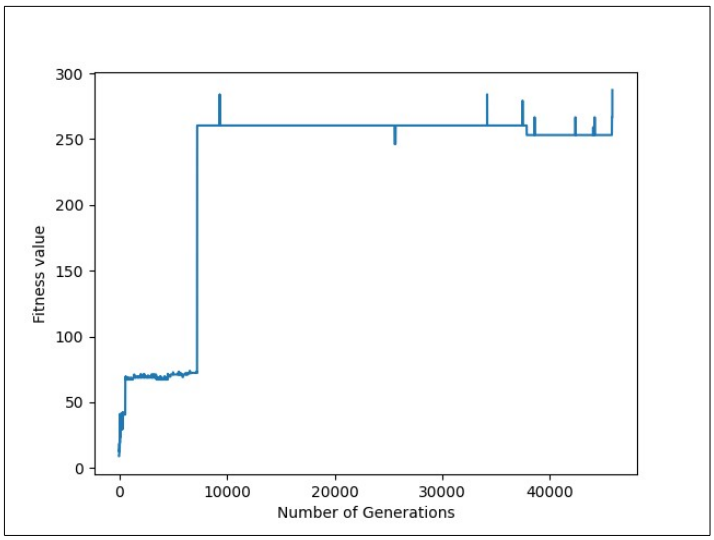
[6, 0, 9, 11, 12, 10, 3, 5, 2, 4, 8, 1, 13, 7]

The **stopping criteria** is a heuristic function, where we look in the past 10 generations and find difference between the maximum fitness and minimum fitness. If the difference started to become very less and is less than 0.000001, we stop. Using this heuristic function, we tend to stop close to the optimum answer.

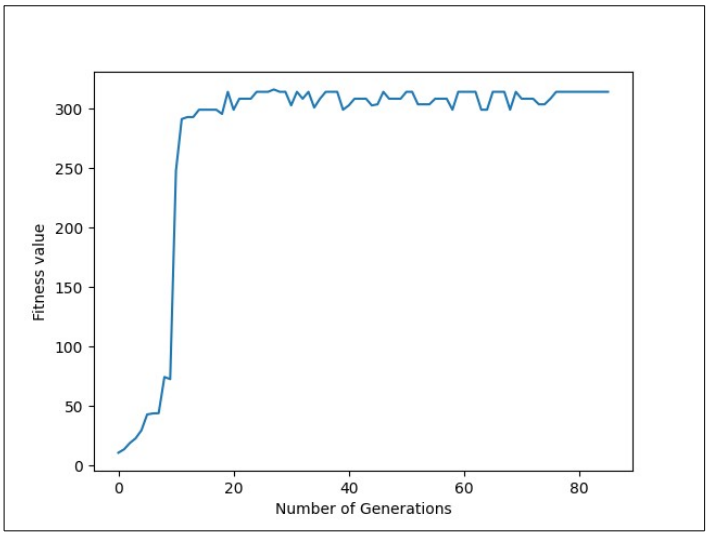
Below are the statistics using the heuristic function.

On average for 100 runs, took **193.9** generations.
On minimum for 100 runs, took **23** generations.
On maximum for 100 runs, took **772** generations.

For one of the run,
Fitness started to become constant after **86** generations
Best fitness value: 314.4654088050315
Distance covered: 3179.9999999999995
Best entity: [9, 0, 6, 11, 12, 10, 3, 5, 2, 4, 8, 1, 13, 7]



Fitness vs Number of generations
in case of Normal Genetic algorithm



Fitness vs Number of Generations in
case of Optimized Genetic algorithm