Test the performance of the GA using different combinations of population sizes and mutation probabilities to understand and analyse the effects of parameters on the GA.

Change the following parameter values to solve the corresponding problem:

a) Population size: 10, 20, 50, 100

b) Mutation Rate: 0.9, 0.6, 0.3, 0.1

Summarize, analyse, and discuss the results.

Genetic Algorithm:

Genetic algorithm is an optimization algorithm that mimics evolution/natural selection by following the principle of "survival of the fittest".

Genetic algorithm can be broken down and applied in few steps as follows:

- Representing Individuals.
- Generating initial and random population.
- Applying fitness function to check the fitness of each population.
- Selecting parents accordingly for mating/crossover by their fitness through random selection.
- Crossover of selected parents to produce a new generation.
- Mutation of new generation to bring diversity.
- Repeat until the solution is reached.

Solving Travelling Salesman problem using genetic algorithm:

Travelling Salesman Problem is a classic combinatorial optimization problem which finds the least-cost travelling route through a set of n cities so that each city is visited only once.

Classic combinatorial optimization problem refers to search of maxima and minima of a problem. While travelling salesman problem focuses on minimizing the "cost". It is a minimization problem.

Analysing and summarising the results after implementation

To apply the genetic algorithm framework, we need to covert a path state into a viable input for the algorithm. As we know that genetic algorithm involves initial population in similar to that of a DNA sequence. So, we can consider a sequence of cities visited as a string of numbers starting from the bottom left.

As mentioned in the problem, we are going to consider different population sizes from the given set (Population size : 10,20,50,100) initially. The simplest fitness function that can be considered here is to find the distance between two cities. Therefore, Pythagoras theorem can be used.

Later, we select parents for “selection” by computing the fitness for each solution and then performing crossover function to produce offspring.

As mentioned in the given code, The mixing number is “ρ = 1”, which can seen as asexual reproduction and uses stochastic beam search for simulating off springs.

After reproduction, a mutation rate is determined how often an offspring can have random mutations to their representation, so as to give diversity for the next generations.

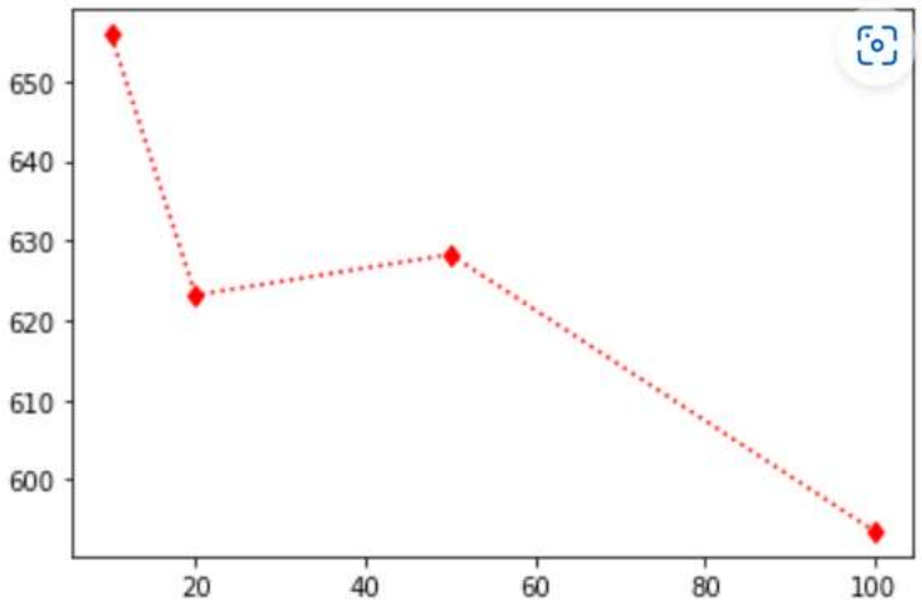
We will iterate this process until we converge to a solution which can be optimum.

Understanding the effect of population size on genetic algorithm:

As we run the algorithm 10 times for each population size within the set (10,20,50,100) and constant mutation rate of 0.3.

I have obtained the program performance for gives sizes as the following:

Population Size (Mutation rate: 0.3):	**Optimal Solution found at:**		**Distance Mean:**	**Mean time taken for computation:**
	Min	**Max**		
10	**108**	**9303**	**655.8660**	0.6962
20	**285**	**7500**	**623.1760**	1.4476
50	**949**	**9479**	**628.2629**	3.8101
100	**1981**	**9610**	**593.5621**	7.7280



As the population size increases, I observed that the speed and computational time of algorithm reduces.

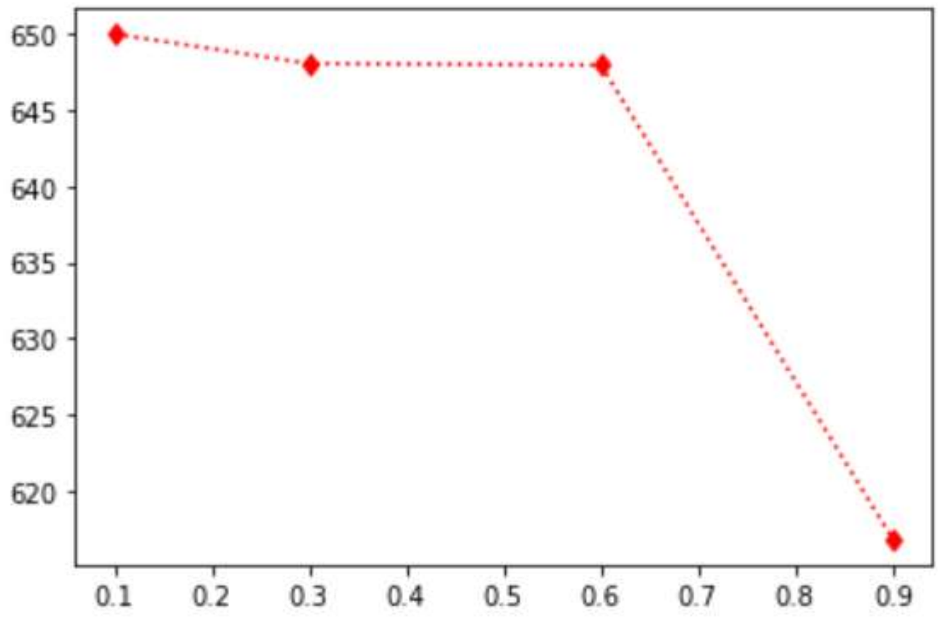
Since, smaller population might not be enough for a good mating pool, optimal population size can be found out by trail and error method through iteration. Looking at the above table and graph, population size 10 took least number of generations and least computational time to receive an optimal solution. While, a population size of 100 has the worst deviations from the means. We can conclude that Population size of 10 can be the most adaptive for “travelling sales man problem” to find poorer solutions in less computational time. Although population size “100” can be advantageous in terms of more diversity of the next generation. It can provide better optimal solutions compared to less population sizes, although it has less convergence rates.

Understanding the effect of Mutation rate on genetic algorithm:

As we run the algorithm 10 times for each mutation rate of set (0.1,0.3,0.6,0.9) with a population size of 20.

I have obtained the following statistics of algorithm performance:

Mutation rate: (Population size '20'):	**Optimal Solution found at:**		**Distance Mean:**	**Mean time taken for computation:**
	Min	**Max**		
0.1	**1180**	**9570**	649.9746606346062	1.4722
0.3	**1085**	**9952**	648.0378459968263	1.3963
0.6	**1171**	**9302**	647.9287902408375	1.4702
0.9	**301**	**9949**	616.8472012223939	1.4653



Mutation rate really helps the algorithm to diversify and explore the solution space for finding the best solution.

From the table above, we can observe that if the mutation rate is better, it helps in finding the optimal solution as quickly as possible. That is, we can state that, too high mutation rate increases the probability of searching more solution state and prevents converging to a solution quickly. On the other hand, small mutation rate can avoid preserving the diversity and result to premature convergence. Although, we can argue that, small mutation rate can hold parents with best fitness score and coverage to an optimal solution sooner. So, according to the statistic obtained, best value mutation rate for the problem can "0.3" because of its computational time and less deviation from the standard mean deviation.

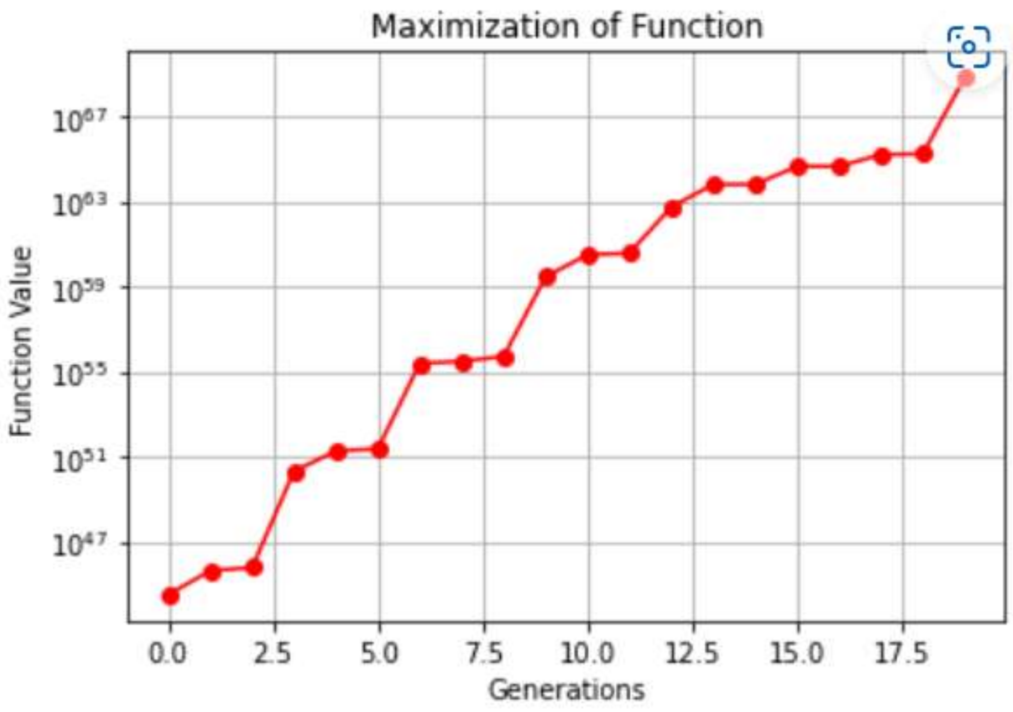
Thus, finally, I conclude that, regarding the exploration of search space for optimal solution, large population can be preferred compared to large mutation rate. The best efficiency of the algorithm can be obtained by large population and less mutation rate.

Task 2: To find maximization function for the given equation:

Fitness function is used to find the fitness score of a solution to find if the solution is closer to optimal solution. For optimization problems, basic functions such as sum of set of calculated parameters can be used to define the maxima or minima of a problem.

Since, a minimization problem involves reducing the cost, the sum should be equal to zero and inverse of the function. While, maximization function can be the function as it is requires higher values.

The graph shows as follows:


Maximization of Function
Function Value
10^{67}
10^{63}
10^{59}
10^{55}
10^{51}
10^{47}
Generations