Homework 1 (Genetic Algorithm on TSP)

Introduction:

This report details the implementation and performance of a genetic algorithm designed to solve the Traveling Salesman Problem (TSP). The algorithm was evaluated on datasets of varying sizes, with a focus on optimizing parameters such as population size, crossover, mutation rates, and elitism to achieve efficient solutions.

Methodology:

Algorithm Overview: (Implementation Link)

The genetic algorithm was implemented in Python and aimed to minimize the total distance of a tour through a set of cities. Key components included:

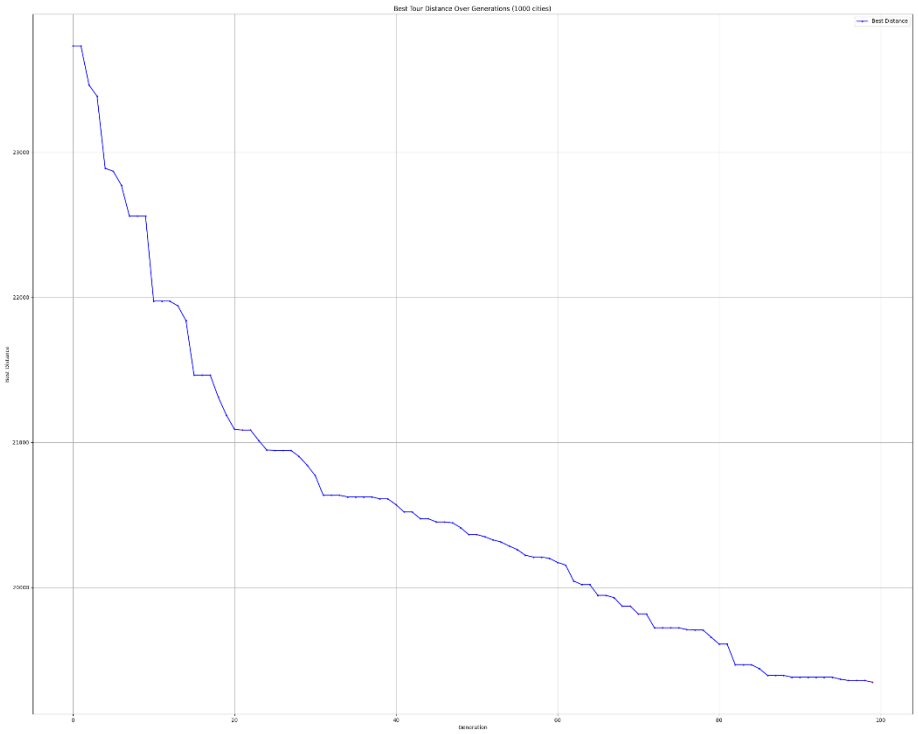
- Fitness Evaluation: **Calculated based on the inverse of tour distance.**
- Selection: Used diversified elitism to select parents, combining top-performing tours with randomly chosen ones to maintain diversity. Default I am taking 25% but have shown different metrices as well.
- Crossover: Employed ordered crossover to generate new offspring.
- Mutation: Introduced variations with an adaptive mutation rate. I have modified the initial mutation rate with a seed variation.

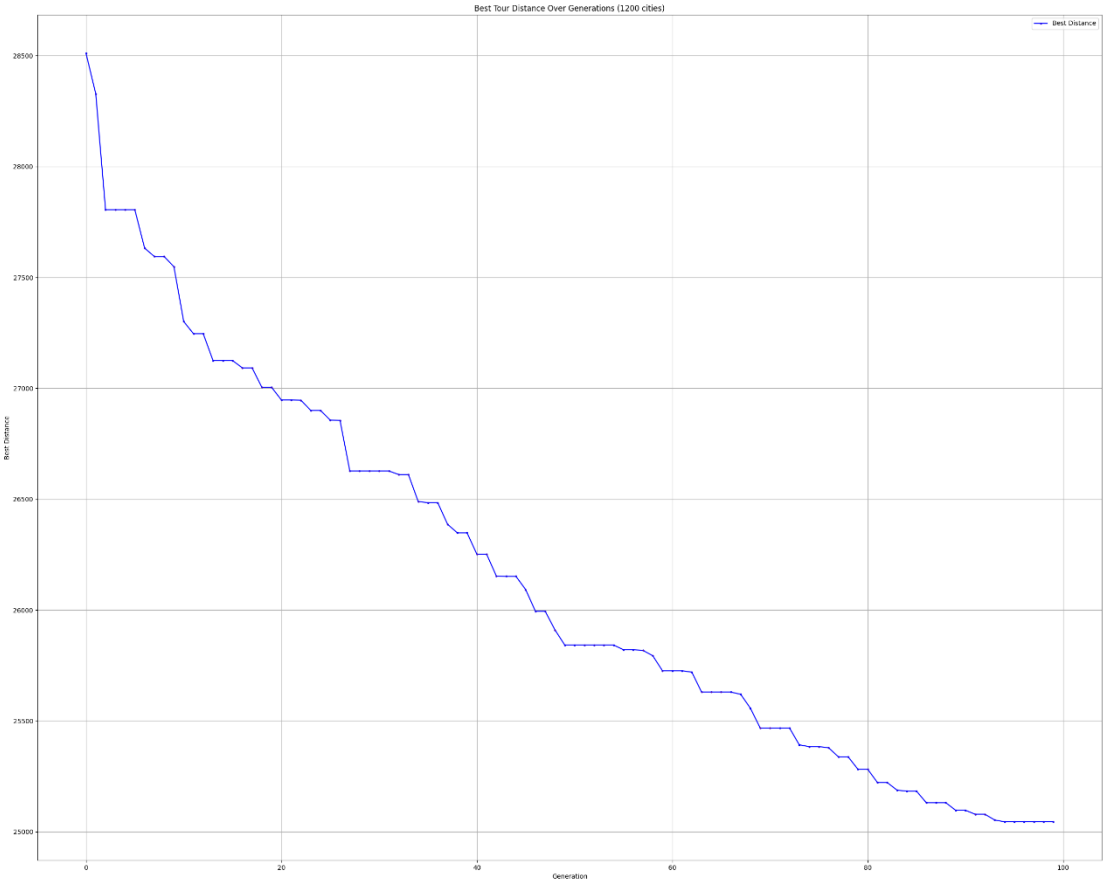
Observations:

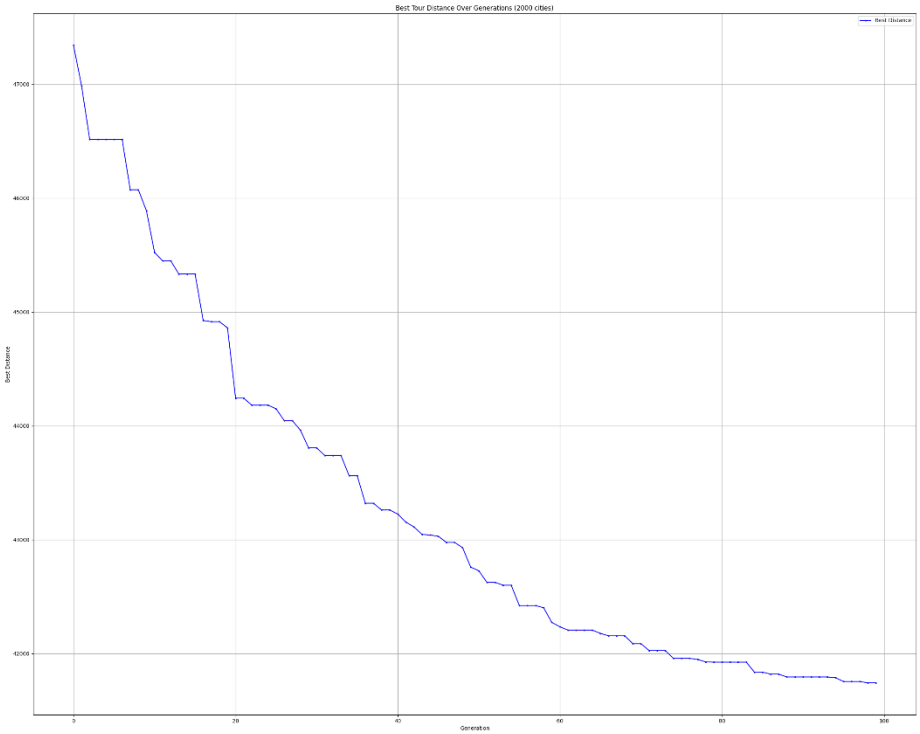
Below figure (figure 1) shows the running time differences given 50 population size with an increasing number of cities. After that I have shown the respective plots to show successive generations as well for these (was not able to generate plot for 5000 cities due to limitation in resources. Also, might have to zoom into the plots as the scale was much higher):

Results with Initial Mu-Rate 10% and 25% Eliticism	
Total Number of Cities	**Time Taken**
1000	47.87 seconds
1200	67.14 seconds
1500	108.91 seconds
2000	3.23 minutes
5000	18.34 minutes

Figure 1: Running time difference in number of cities.


Best Tour Distance Over Generations (1000 cities)
Best Distance
Best Distance
Generation
23000
22000
21000
20000
0
20
40
60
80
100


Best Tour Distance Over Generations (1200 cities)
Best Distance
Best Distance
Generation
28500
28000
27500
27000
26500
26000
25500
25000
0
20
40
60
80
100



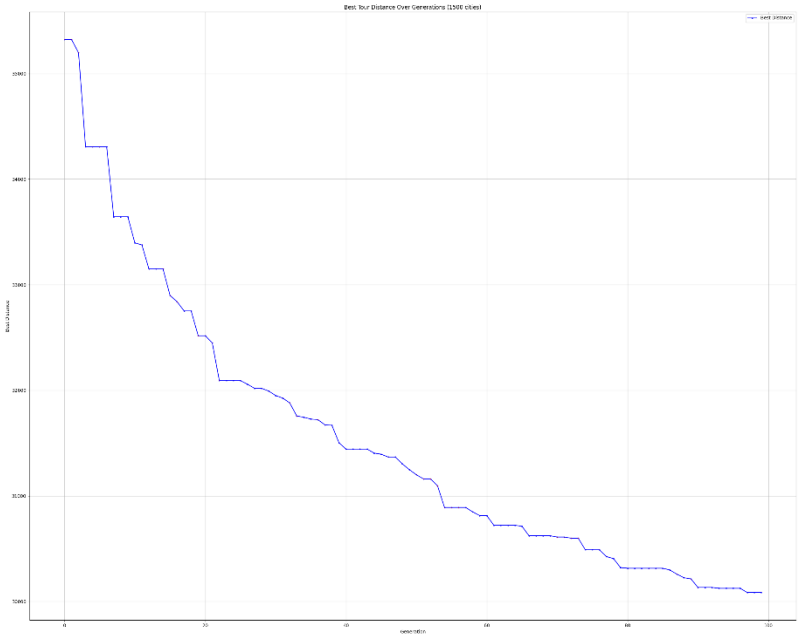


Figure 2: Plotting 50 population with mutation rate 10% and Elitism 25% for increasing number of cities.

These results helped pick **2000 cities** as a benchmark, as it had an optimized running time compared to lower numbers (low running time) and higher numbers (too high running time). This also helped us to understand the increase of running time with increasing number of cities, which is exponential, and expected out of the genetic algorithm.

On the basis of 2000 cities, I further ran variations on the different parameters like mutation rate, elitism on different population size to further get results closer to outcome. Figure 3 shows the results in a tabular way. The metric I chose to compare is likeness, which I have taken to be relative distance between the outcome of the algorithm for cost and actual cost given the same data (I have used a parallel code with Bellman-Ford Algorithm which tested to be giving the closest to actual outcome to this problem).

Results with Fixed 2000 Cities (above 3 minutes)					
Population Size	**Mutation Rate**	**Elitism**	**Cost Generated**	**Cost Actual**	**Likeness**
50	1%	5%	44414	45634	2.709665956
	9%	8%	43985	45634	3.680023209
	3%	3%	45486	45634	0.324846356
100	1%	5%	44077	45634	3.471146236
	9%	8%	43427	45634	4.956153648
	3%	3%	45197	45634	0.962226553

Figure 3: Result for fixed number of cities with parameter variation

I have highlighted the best result that I got. I have also attached the plot for each variation in the appendix section for a better understanding of each outcome.

Observations:

Based on the results, certain observations can be made:

1. I got the best results on a population size 50%, mutation rate 3% and elitism 3%, which is somewhat dependent on the dataset I used. The results can be different on a completely different dataset.
2. The results show too much, or too little mutation shows a drastic difference in the outcome. I have used 1% and 9% mutation rates as well, which shows huge differences in relative terms.
3. The impact of elitism is relatively less impacting. In decreasing elitism, I got better likeness.
4. The impact of population size is also less in the results. Changing population from 50 to 100 did not yield in hugely different approximation here. One important thing I noticed here is that the pattern of likeness stayed the same, even with population size changes.

One important thing to note here is that I have used the inverse of distance as the fitness evaluation metric. Using a different fitness evaluation may yield better results, or different results altogether.

Appendix:

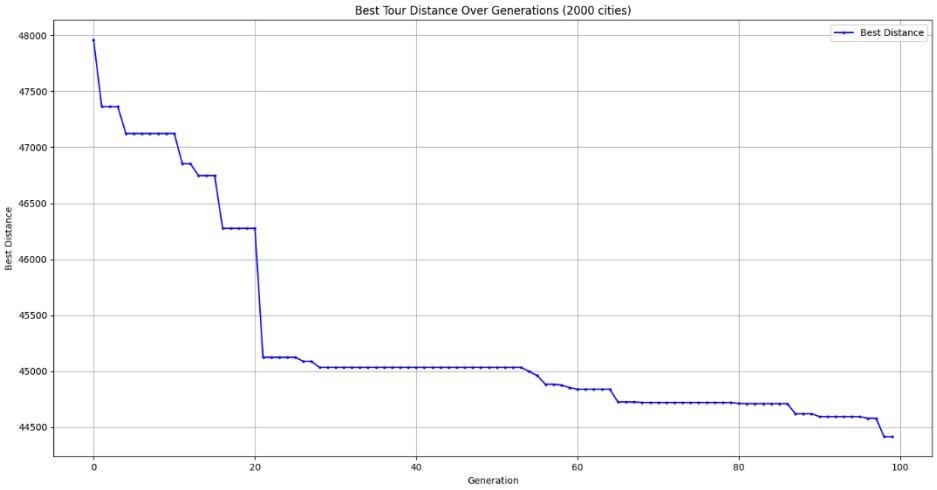


Figure: Population: 50, Mutation: 1%, Elitism: 5%

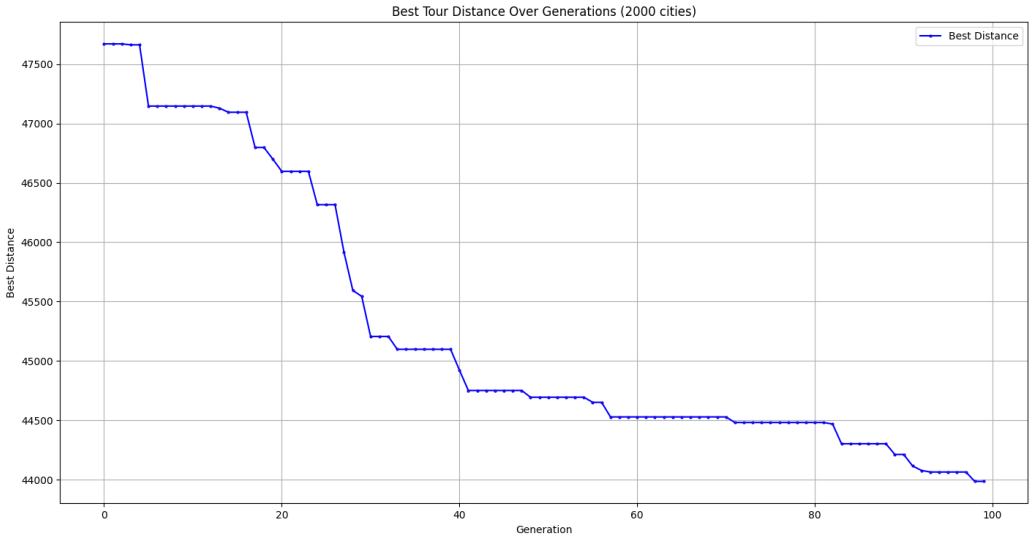


Figure: Population: 50, Mutation: 9%, Elitism: 8%

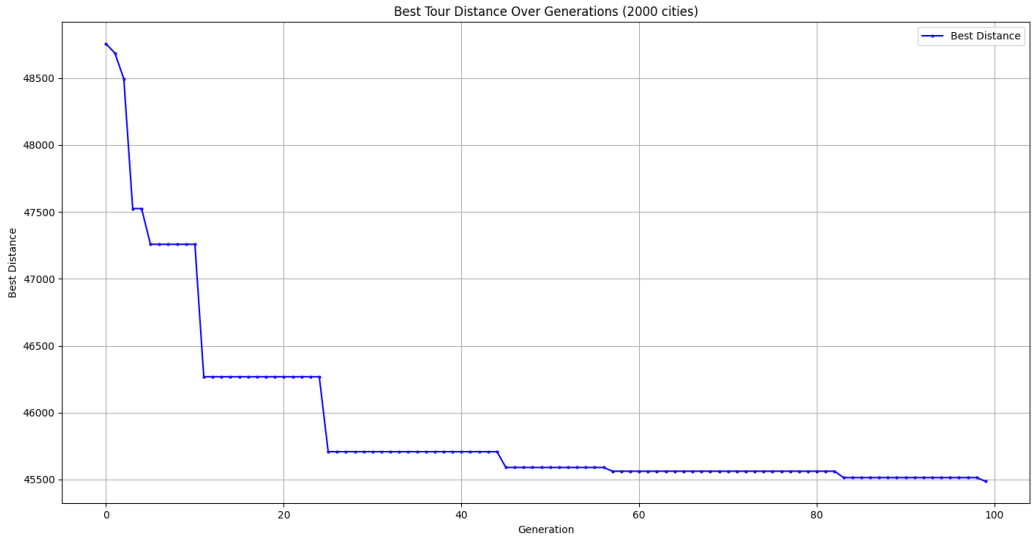


Figure: Population: 50, Mutation: 3%, Elitism: 3%

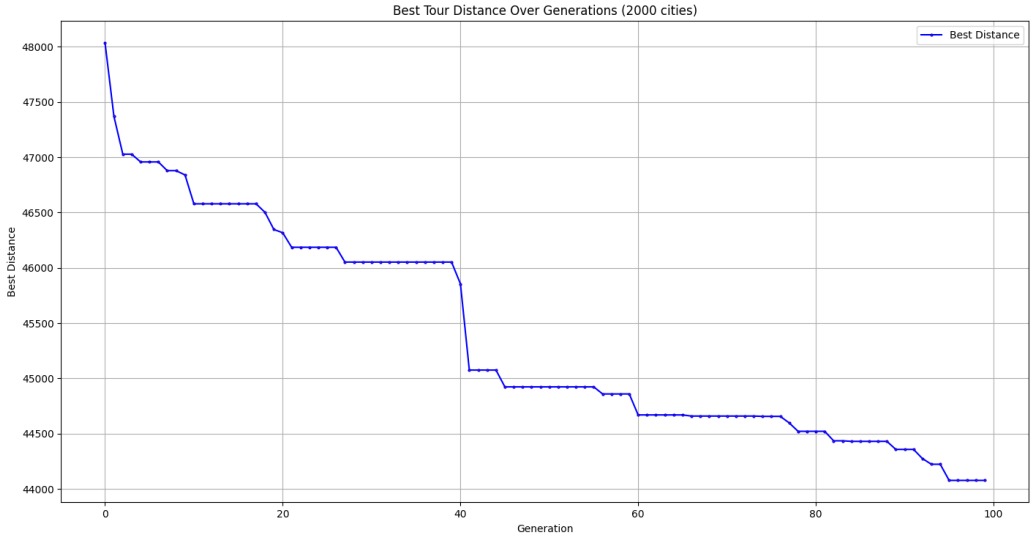


Figure: Population: 100, Mutation: 1%, Elitism: 5%

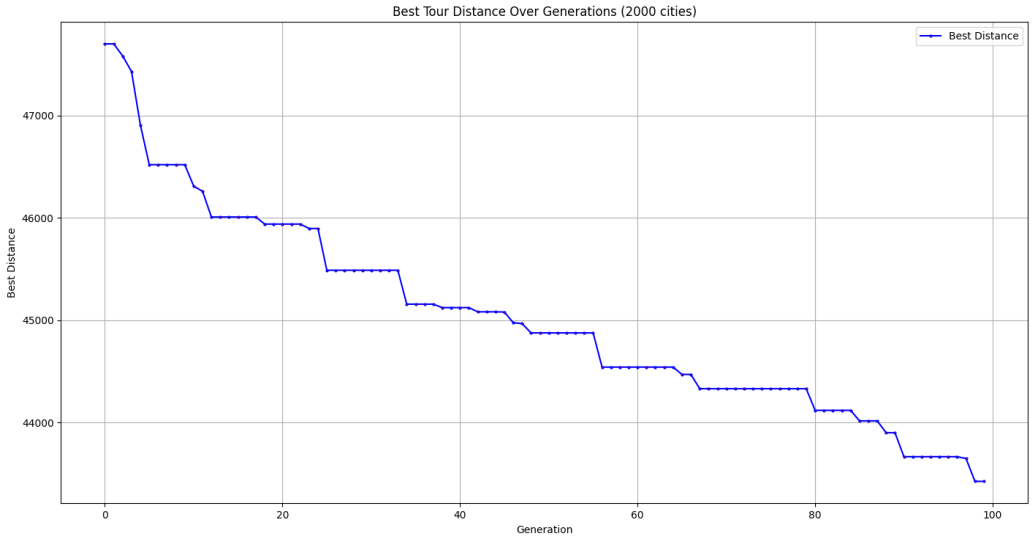


Figure: Population: 100, Mutation: 9%, Elitism: 8%

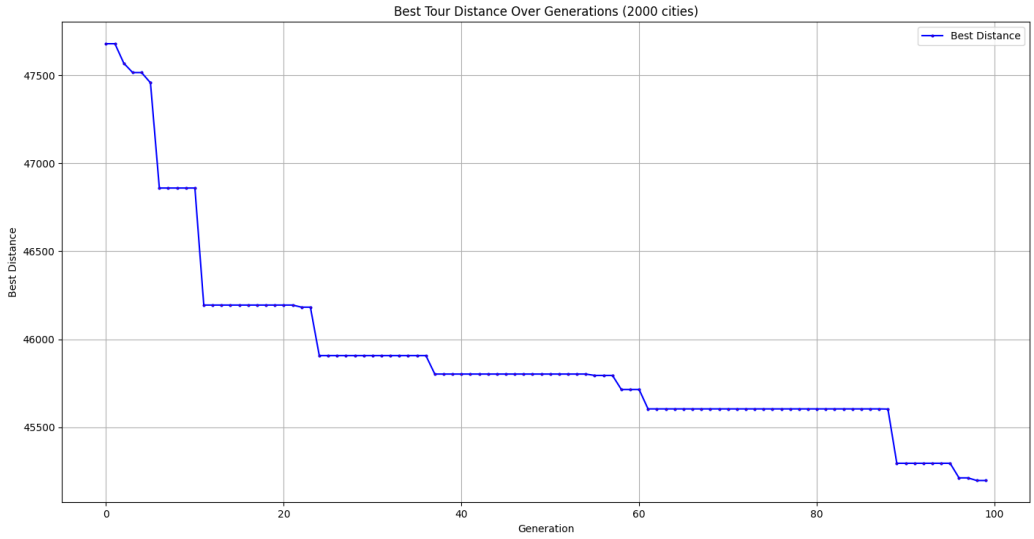


Figure: Population: 100, Mutation: 3%, Elitism: 3%