Genetic Algorithms Project Final Report

Student Name: Dogancan Yucel
Index Number: 104009
Course: Advanced Python Programming Techniques
Date: 15/01/2026

PART 1: ALGORITHM OVERVIEW

Representation: Path Representation (Permutation of City IDs).

Initialization (Hybrid): 20% of the population is generated using Greedy (Nearest Neighbor) to boost early performance; 80% is Random to ensure diversity.

Selection: Tournament Selection (Size k=5). Better pressure control than Roulette Wheel.

Crossover: Ordered Crossover (OX1). Essential for TSP to prevent duplicate cities in a tour.

Mutation: Inversion Mutation. Reverses a segment of the tour. Superior to swap mutation for untangling loops in 2D space.

Fitness: Total Euclidean Distance (Minimization).

Elitism: The best individual is preserved in every generation.

PART 2: PARAMETER TUNING

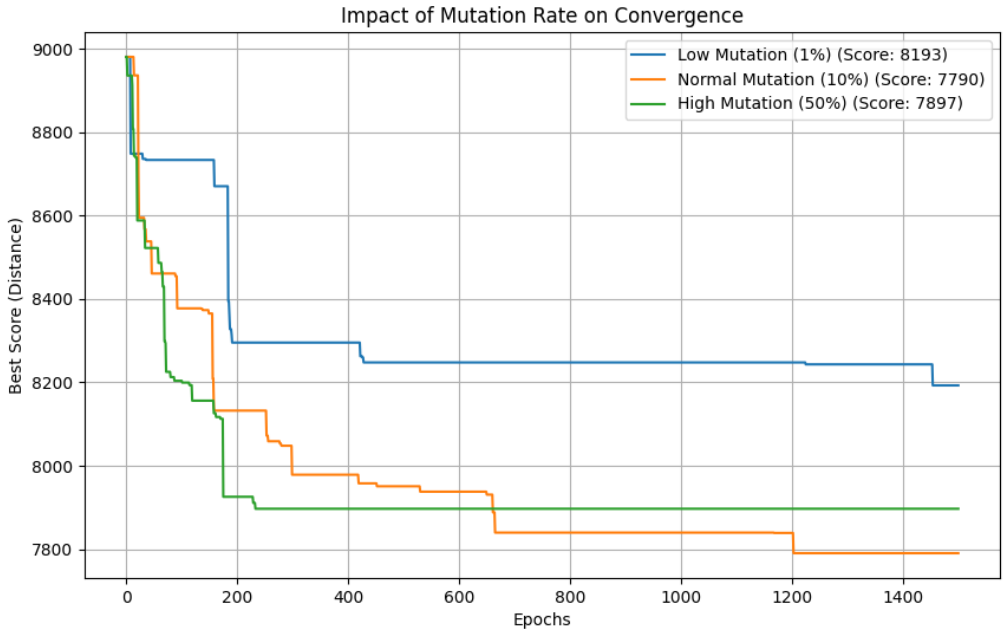
Two key parameters were tested to optimize performance (Berlin52).

2.1. Mutation Probability

1% (Low): Premature convergence (stuck in local optima).

50% (High): Chaotic behavior, lost good structures.

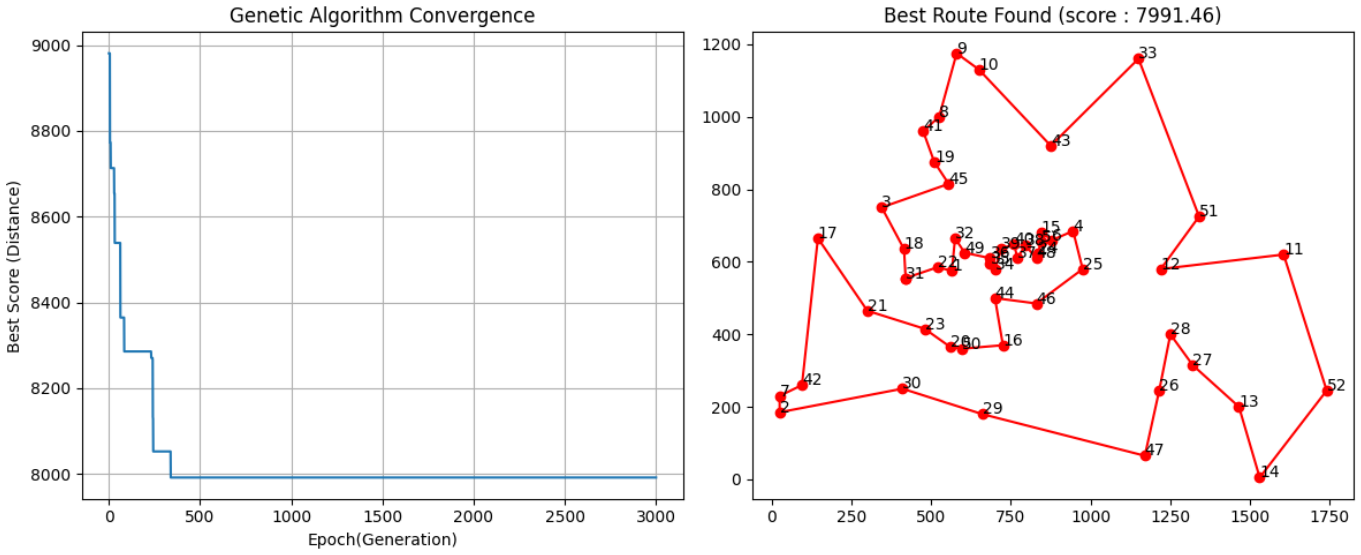
10% (Optimal): Best balance between exploration and exploitation.Decision: Fixed at 10-15%.



2.2. Initialization Strategy

Random Start: Starts with high distance, slow convergence.

Hybrid Start (with Greedy): Starts much closer to the optimal solution. Drastically reduces computation time.



PART 3: COMPARATIVE RESULTS

The algorithm was tested against Random Search (1000 runs) and Greedy Algorithm on 4 datasets.

Result: GA outperformed Greedy in all cases.

Stability: Standard Deviation is extremely low (0.00 on Berlin11), proving reliability.

PART 3: EXPERIMENTAL RESULTS (COMPARISON)

In this section, the performance of the Genetic Algorithm (GA) is compared with Random Search and the Greedy Algorithm across four different datasets.

3.1. Small Scale Test: berlin11_modified.tsp (11 Cities)

Observation: The Genetic Algorithm found the optimal solution (4038.44) in every single run (Standard Deviation = 0.00), proving its reliability on small datasets.

Metric	Random Search	Greedy Algorithm	Genetic Algorithm
Best Score	4777.31	4543.09	4038.44
Mean	7450.30	4842.21	4038.44
Std. Dev.	810.43	255.65	0.00

3.2. Standard Benchmark: berlin52.tsp (52 Cities)

Observation: The GA consistently outperformed the Greedy algorithm, improving the solution by approximately 6%.

Metric	Random Search	Greedy Algorithm	Genetic Algorithm
Best Score	23406.36	8182.19	7679.15
Mean	29915.70	9373.43	7993.79
Std. Dev.	1618.76	476.21	148.12

3.3. Large Scale Test: kroA100.tsp (100 Cities)

Observation: As requested in the project requirements, the algorithm was tested on a 100-city problem. The GA showed a significant improvement (~13.8%) over the Greedy approach, reducing the distance from 24,698 to 21,285.

Metric	Random Search	Greedy Algorithm	Genetic Algorithm
Best Score	141900.34	24698.50	21285.44
Mean	170757.06	27045.21	22002.89
Std. Dev.	8182.68	816.20	370.39

3.4. Extra Stress Test: kroA150.tsp (150 Cities)

Observation: Even with 150 cities, the algorithm maintained stability and outperformed other methods.

Metric	Random Search	Greedy Algorithm	Genetic Algorithm
Best Score	226976.79	31482.02	28347.09
Mean	257931.06	33639.92	28736.71
Std. Dev.	10238.64	865.30	258.85

3.5. Overall Conclusion

The tests confirm that the implemented Genetic Algorithm is robust and scalable.

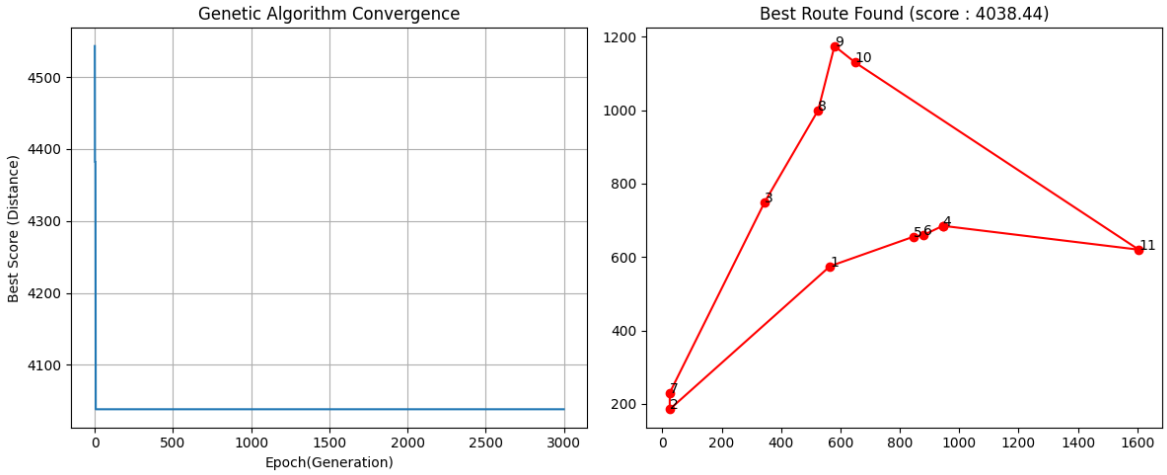
Superiority: It beat the Greedy algorithm in all test cases.

Stability: The low standard deviation (especially in larger maps like kroA100 where SD is only ~1.6% of the mean) indicates that the algorithm converges to a good solution reliably, not just by luck.

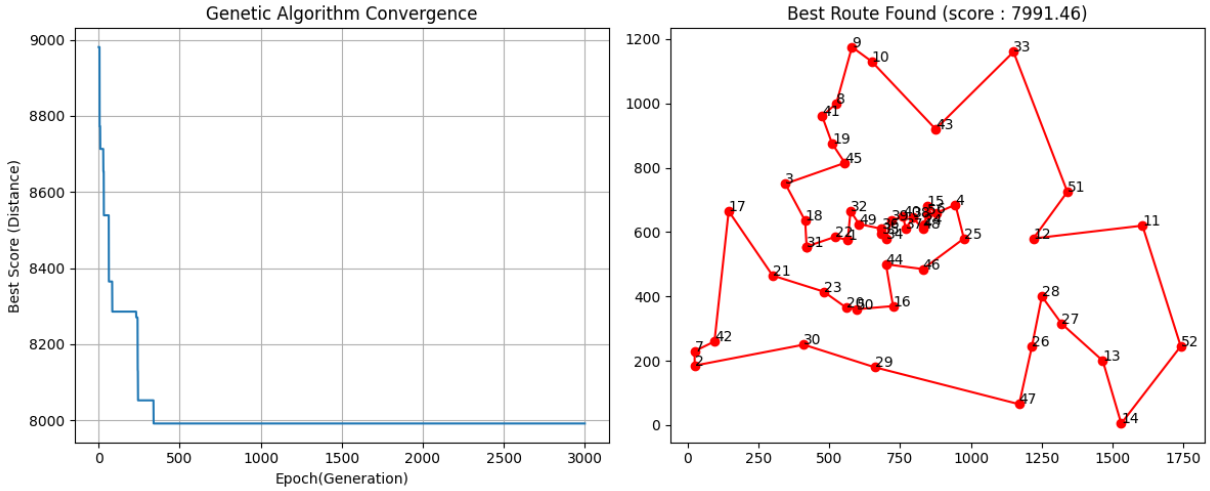
Scalability: It successfully handled the 100-city requirement and even the 150-city dataset without performance degradation.

PART 4: VISUALIZATION

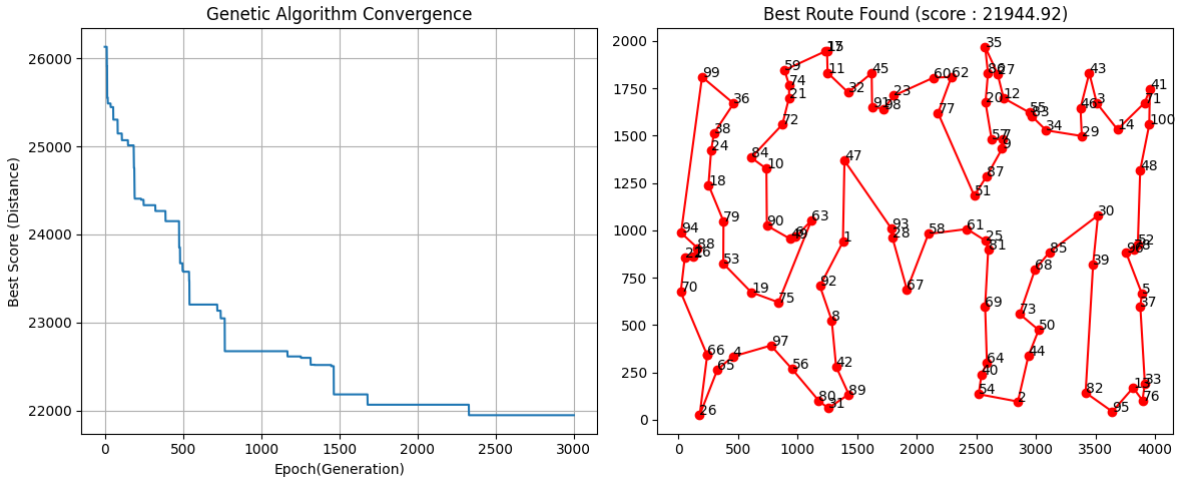
4.1. Berlin11 Solution



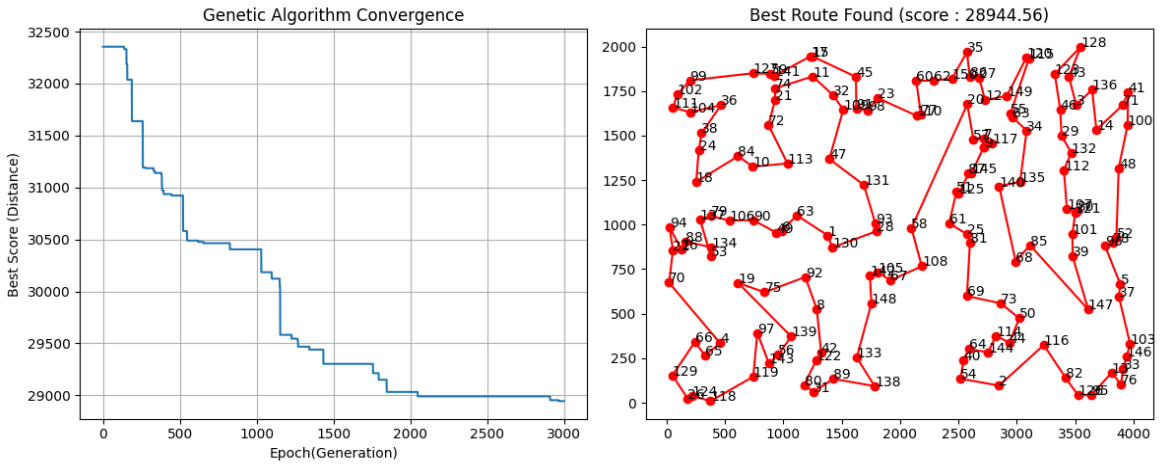
4.2. Berlin52 Solution



4.3. KroA100 Solution



4.4. KroA150 Solution



PART 5: CONCLUSION

Effectiveness: The Genetic Algorithm successfully solved the TSP for up to 150 cities, consistently beating the Greedy heuristic.

Hybrid Advantage: Injecting Greedy solutions into the initial population was the most critical factor for speed and accuracy.

Scalability: The algorithm maintained stability (low variance) even on larger maps like kroA100, satisfying the project's advanced requirements.