**GA\_2023\_2024\_LastName\_FirstName**

**Genetic Algorithm Report**

**Part 1: Algorithm Description**

The implemented algorithm is a Genetic Algorithm (GA) designed to solve the Traveling Salesman Problem (TSP). The following components were used:

* **Mutation**: A simple swap mutation was used, where two random cities in a route are swapped with a given mutation probability.
* **Selection**: Tournament selection was implemented, where a subset of solutions competes, and the best one is chosen.
* **Crossover**: A simple crossover was used where the first half of one parent is taken, and the remaining cities are filled from the other parent.
* **Initial Population**: The initial population was created using random permutations of city orders.
* **Solution Representation**: Each solution is represented as a sequence of city IDs, defining the order in which they are visited.

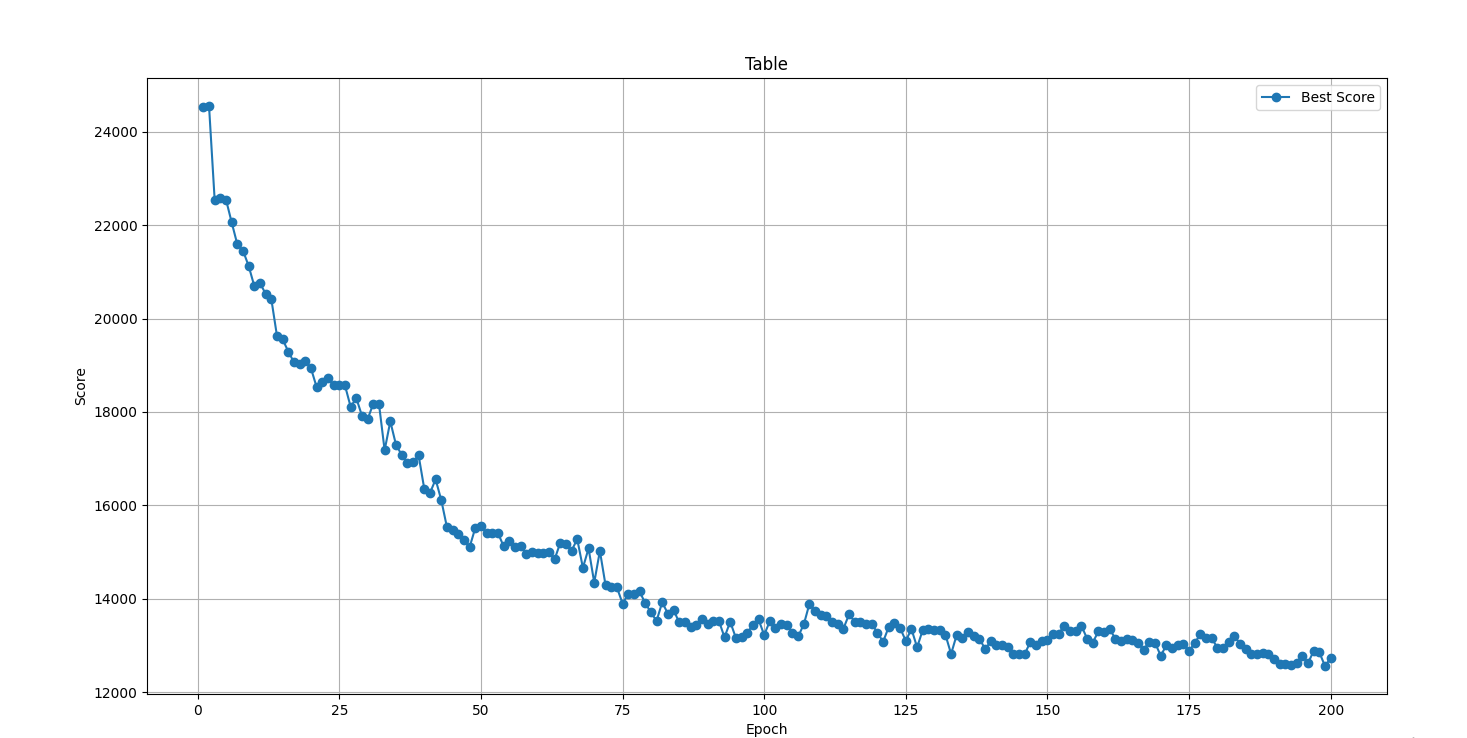
**Part 2: Parameter Testing and Comparison**

Three key parameters were tested for their effect on the final result:

1. **Mutation Probability** (0.2, 0.5, 0.8)

çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, metin içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram, paralel içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

1. **Initial Population Size** (50, 100, 200)çizgi, diyagram, metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

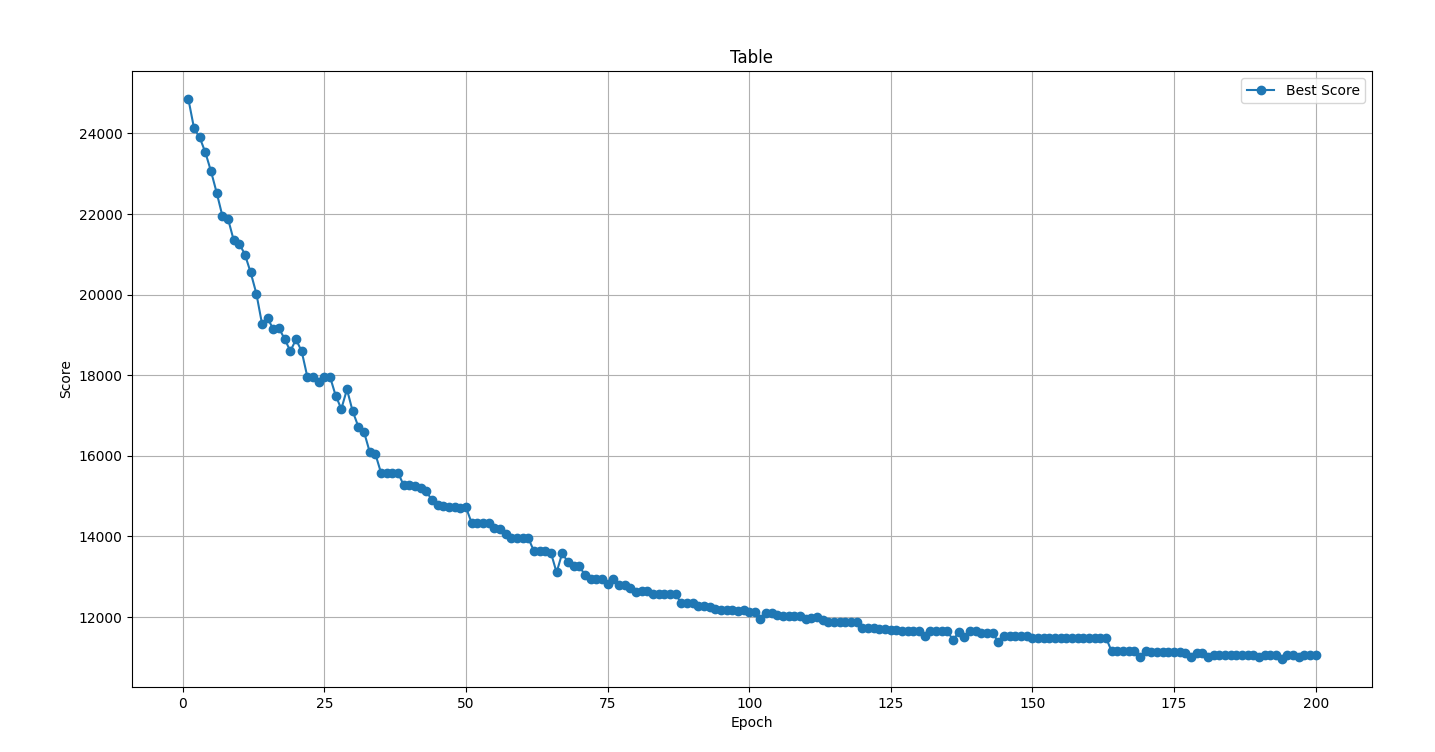
   Yapay zeka tarafından oluşturulan içerik yanlış olabilir.çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram, metin içeren bir resim

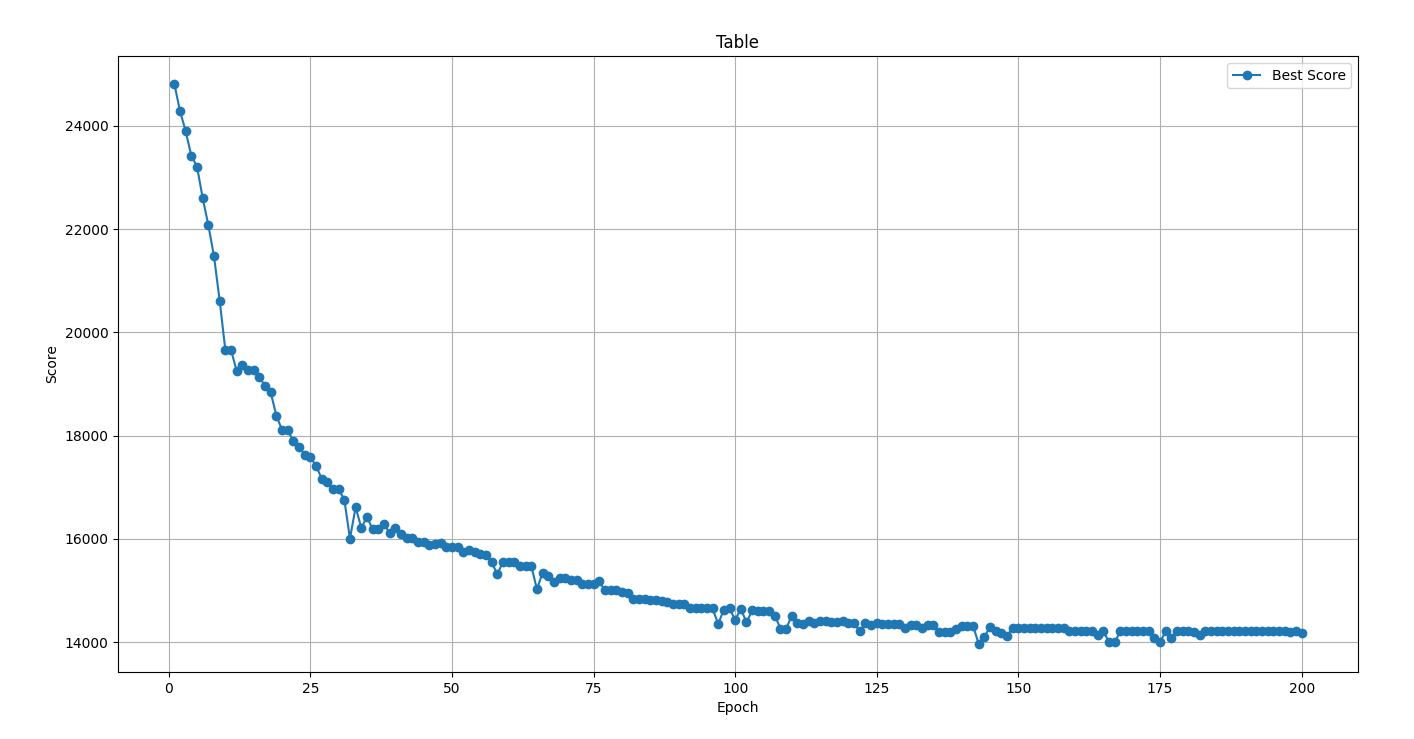
   Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

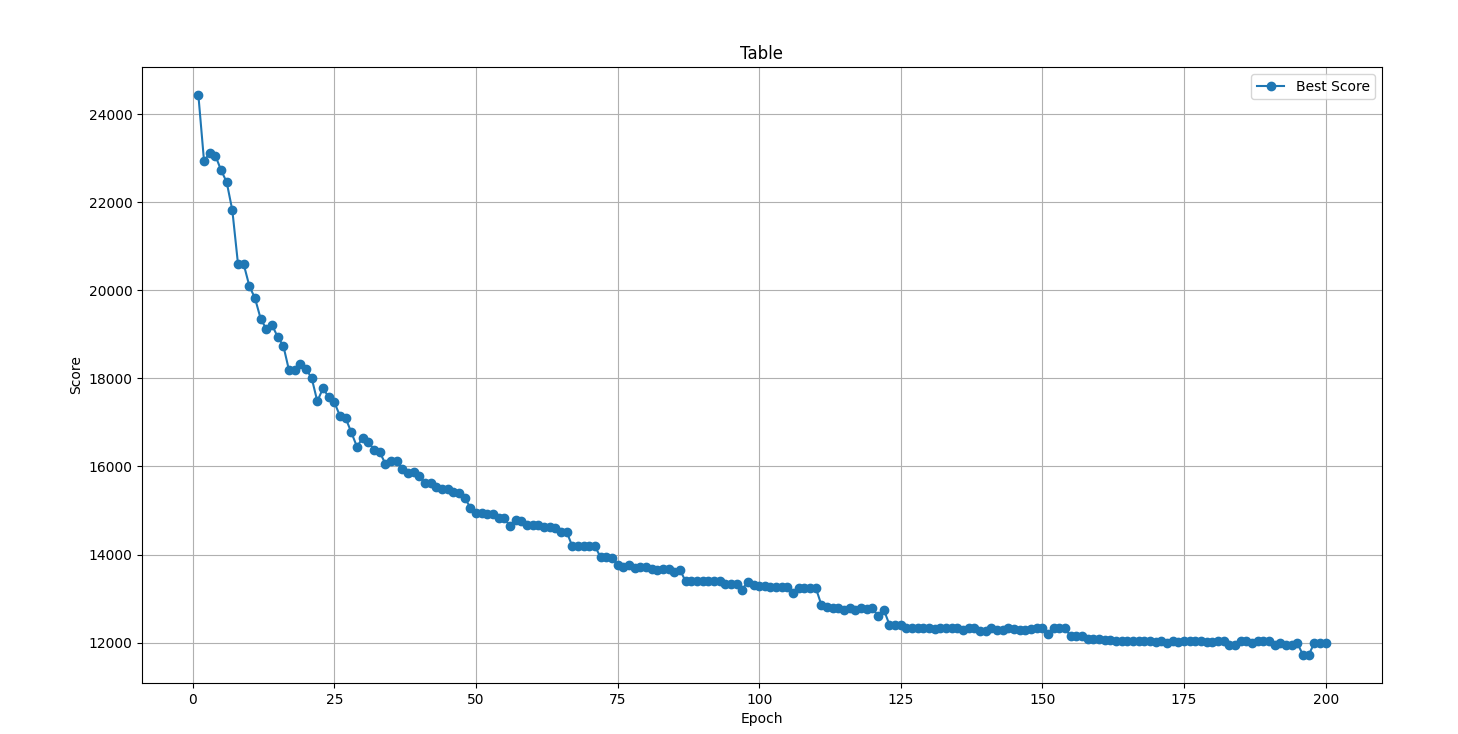
çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

1. **Tournament Size** (3, 5, 7)







The tests were conducted over multiple runs with a fixed seed to ensure fair comparisons. Below are the results:

* **Mutation Probability Effect**: Higher mutation probability allowed for better exploration but too high values led to instability.
* **Population Size Effect**: A larger population resulted in better solutions but increased computational time.
* **Tournament Size Effect**: A moderate tournament size provided the best balance between selection pressure and diversity.

Graphs showing the best score per generation for different parameter values are included.

**Part 3: Comparison with Other Methods**

The genetic algorithm’s performance was compared against the greedy and random approaches:

* **GA Results**: The best solution was obtained in 10 runs, with statistical measures including mean, standard deviation, and variance.
* **Greedy Algorithm**: The best 5 solutions were recorded, along with statistical data.
* **Random Algorithm**: 1000 runs were performed, and the best result along with statistical data was presented.

Results are summarized in the following (i tested them on berlin11.tsp file to get results quickly

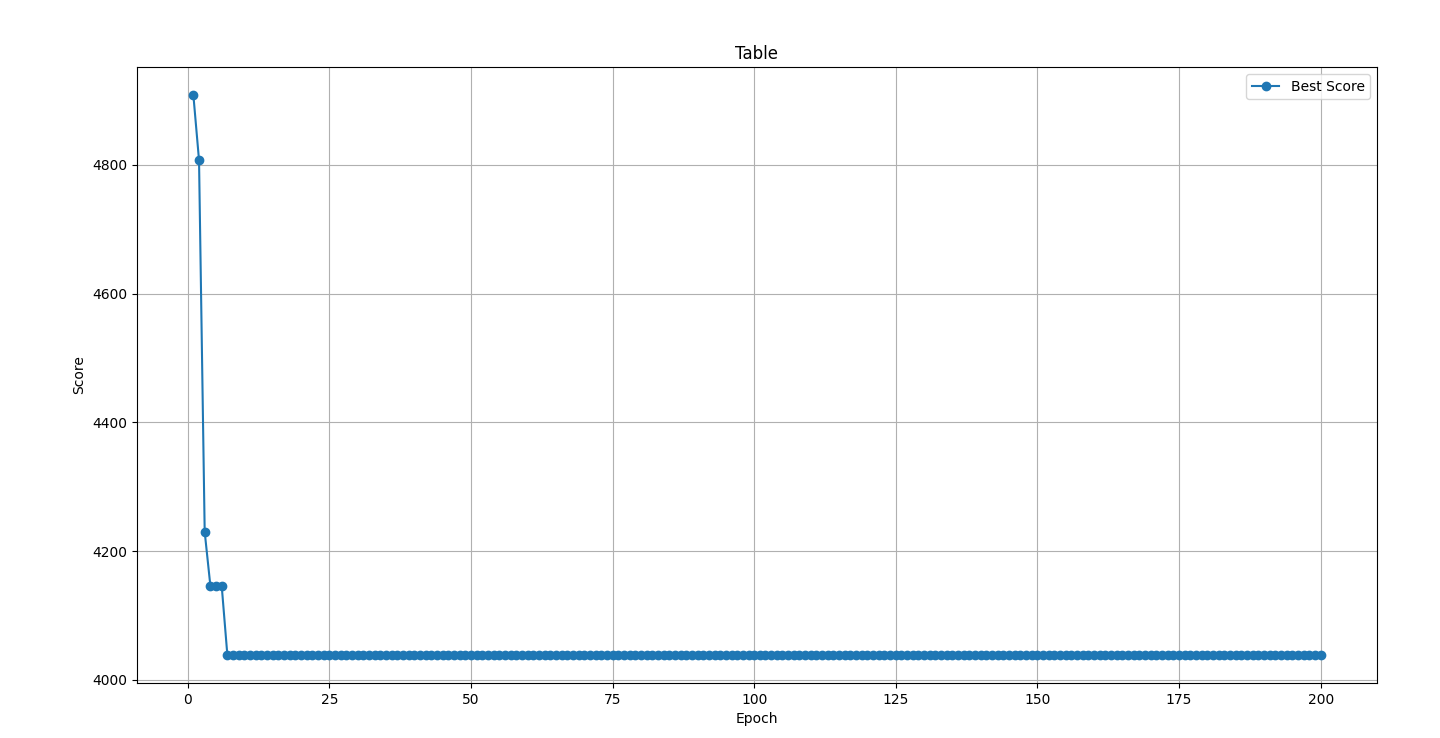
GA 4038.4379127952357

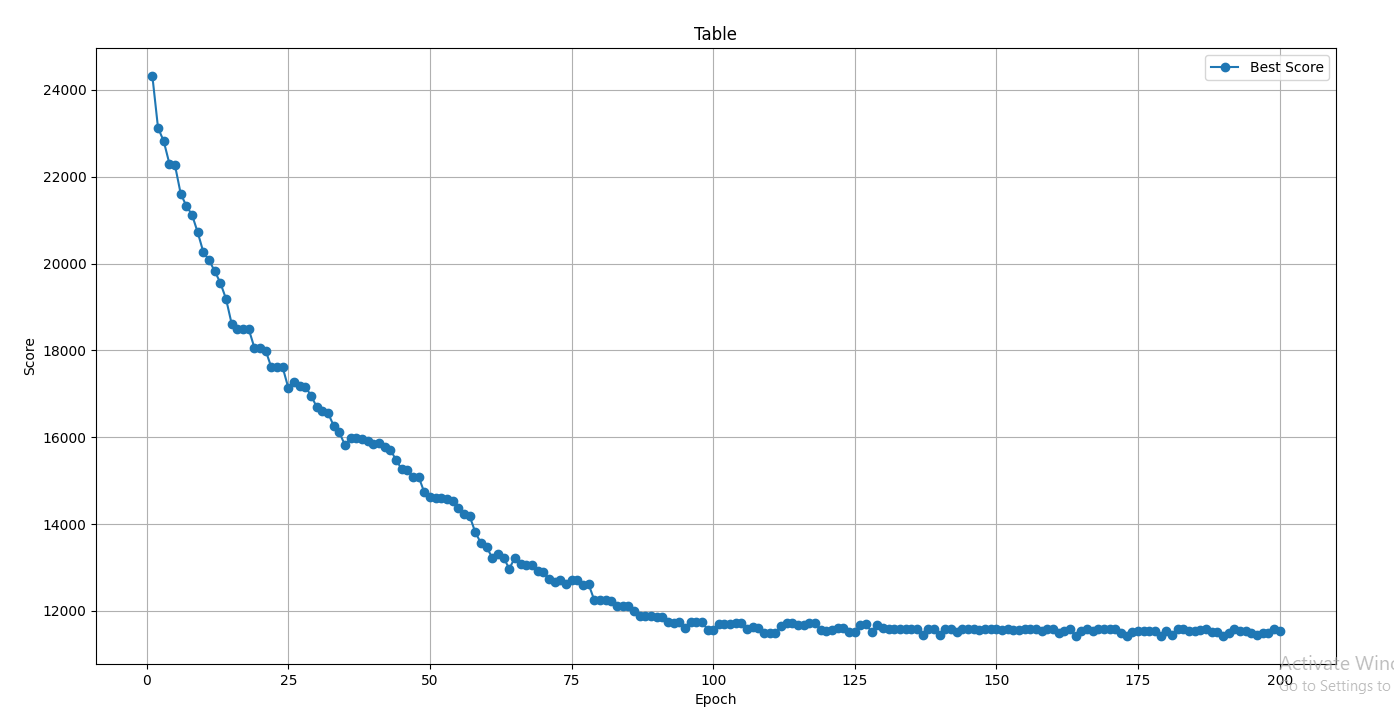
Greedy 5512.279530223165

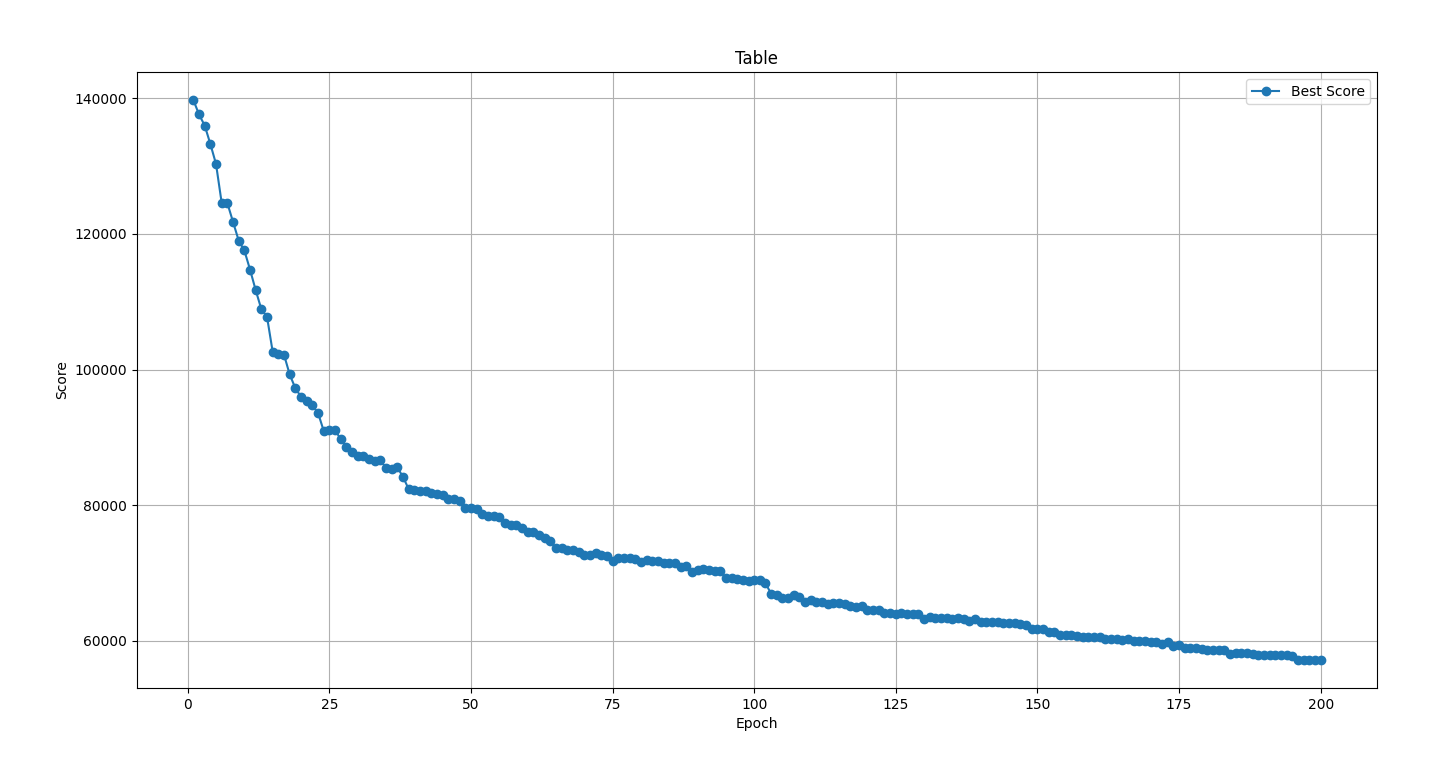
Random Solution 8148.253489539865

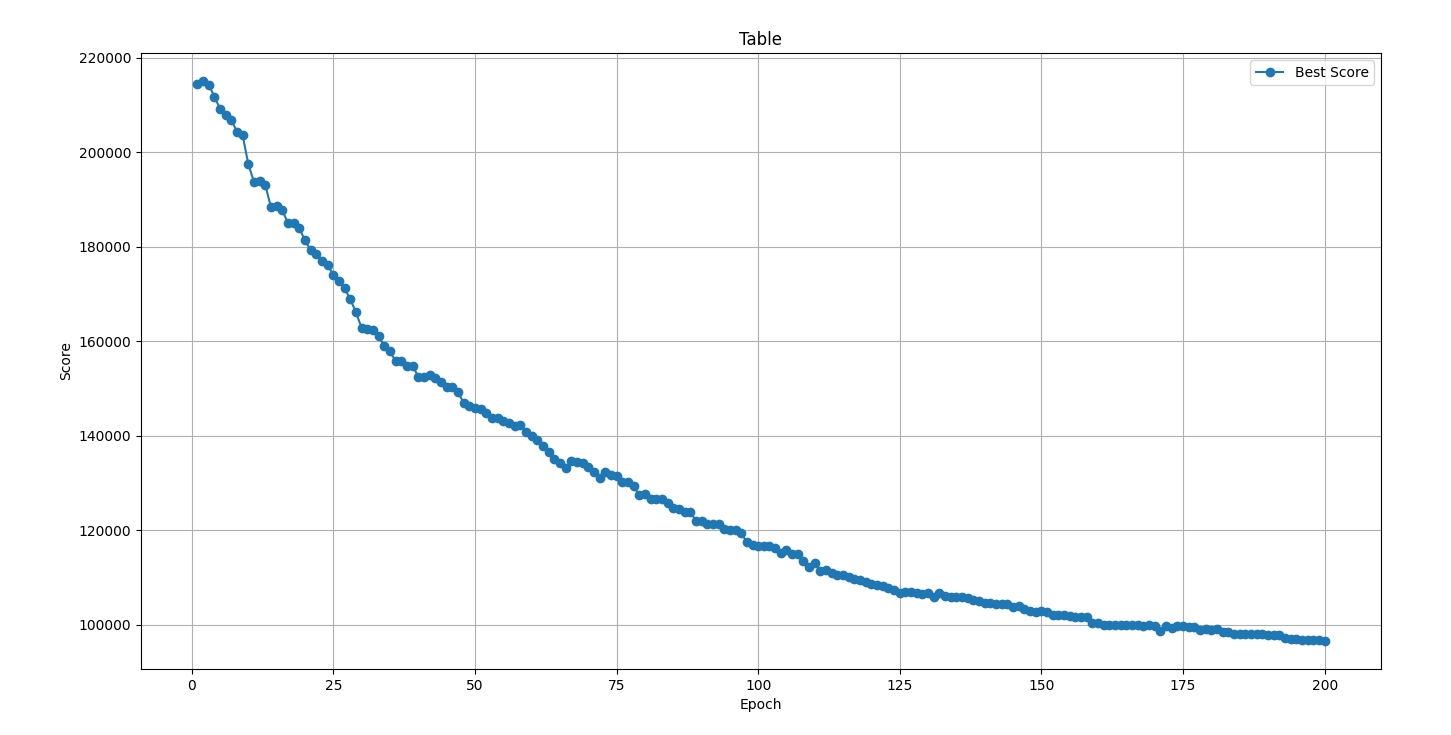
**Part 4: Testing on Different Data Sets**

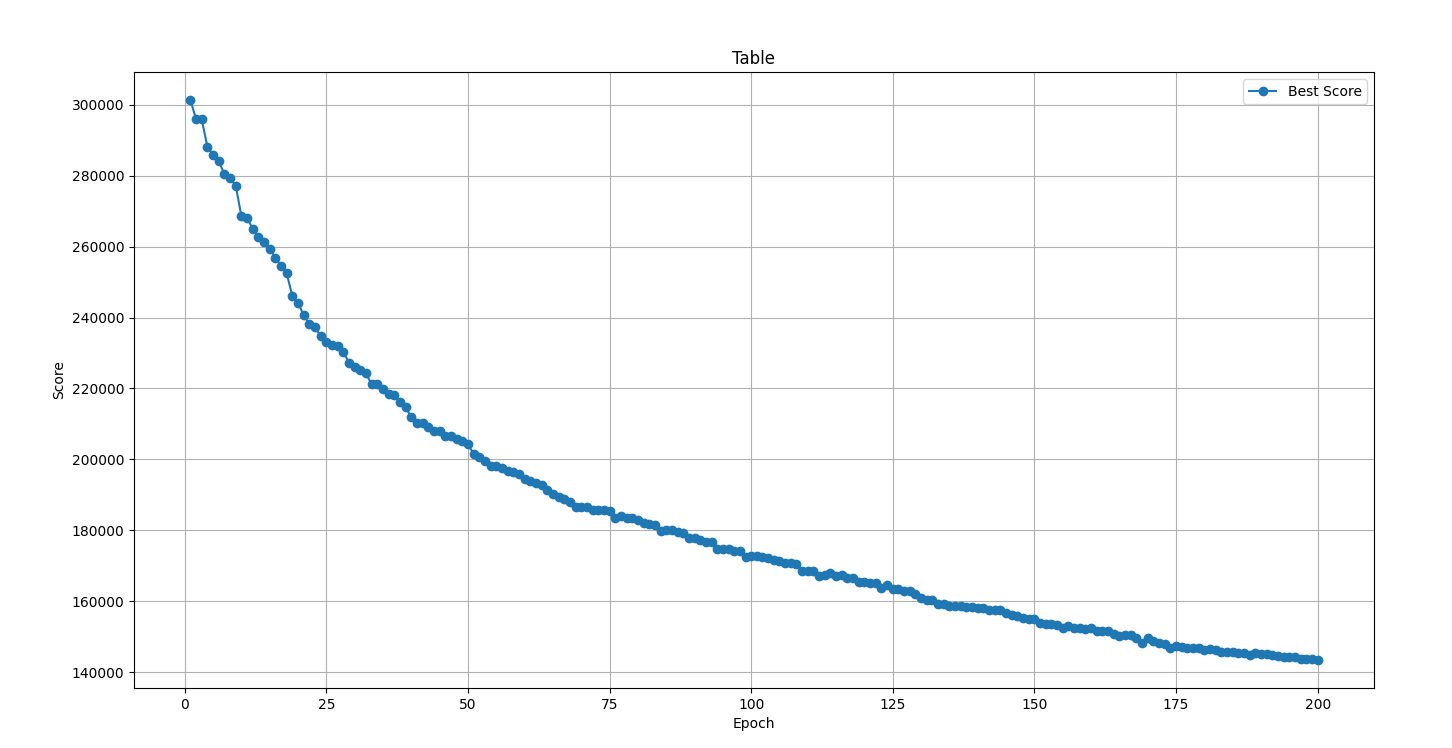
The algorithm was tested on problem sizes of 11, 52, 100, and 150 cities. The best solutions were visualized using Matplotlib for two selected cases.











**Part 5: Conclusion**

* The genetic algorithm outperformed the random and greedy methods.
* A balanced mutation rate and population size are key to obtaining good results.
* Future improvements could include more advanced crossover techniques and adaptive mutation rates.