1 Problem Formulation

We represent a solution as a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of the n cities, where city indices come from the instance file (e.g., `tsp_100_1`).

1.1 Decision Variables

Let

$$\pi_k \in \{1, 2, \ldots, n\}, \quad \text{denote the city visited at step } k.$$

Each city appears exactly once in π, i.e.

$$\{\pi_1, \pi_2, \ldots, \pi_n\} = \{1, 2, \ldots, n\}.$$

1.2 Objective Function

Minimize the total tour length:

$$\min_{\pi} f(\pi) = \sum_{k=1}^{n-1} d_{\pi_k, \pi_{k+1}} + d_{\pi_n, \pi_1}.$$

1.3 Constraints

- **Uniqueness:** No city is visited more than once: $\pi_i \neq \pi_j$ for $i \neq j$.
- **Closed tour:** The tour returns to its start city (enforced by d_{π_n, π_1}).

2 Genetic Algorithm Design

This section describes our GA architecture for the TSP.

2.1 Chromosome Representation

A tour is encoded as a permutation

$$\pi = (\pi_1, \ldots, \pi_n)$$

of the n cities.

2.2 Population Initialization

The initial population P_0 consists of N random permutations:

$$P_0 = \{\pi^{(1)}, \ldots, \pi^{(N)}\}, \quad \pi^{(i)} \sim \text{Uniform}(\{1, \ldots, n\}).$$

2.3 Genetic Operators

- **Selection:** Tournament selection of size k.
- **Crossover:** Partially Mapped Crossover (PMX).
- **Mutation:** Swap mutation (swap two random positions).
- **Replacement:** Elitism—carry the top 1–2 individuals forward.

2.4 Fitness Function

The fitness is defined as the tour length:

$$f(\pi) = \sum_{k=1}^{n-1} d_{\pi_k, \pi_{k+1}} + d_{\pi_n, \pi_1}.$$

2.5 Stopping Criteria

- Maximum generations $G_{\max}$ reached.
- No improvement in the best fitness for T consecutive generations.

2.6 Advanced Features

We implemented two enhancements beyond the basic GA:

Adaptive Mutation Rate The mutation probability μ decreases linearly from its initial value μ_0 to nearly zero over G generations:

$$\mu(g) = \mu_0 \times \left(1 - \frac{g}{G - 1}\right).$$

2-Opt Local Search After each mutation, a 2-opt local search is applied to remove crossing edges and polish the tour. This memetic hybridization improves solution quality and accelerates convergence.

3 Results and Analysis

We conducted experiments by varying population size and mutation rate under two budgets (50 and 100 generations).

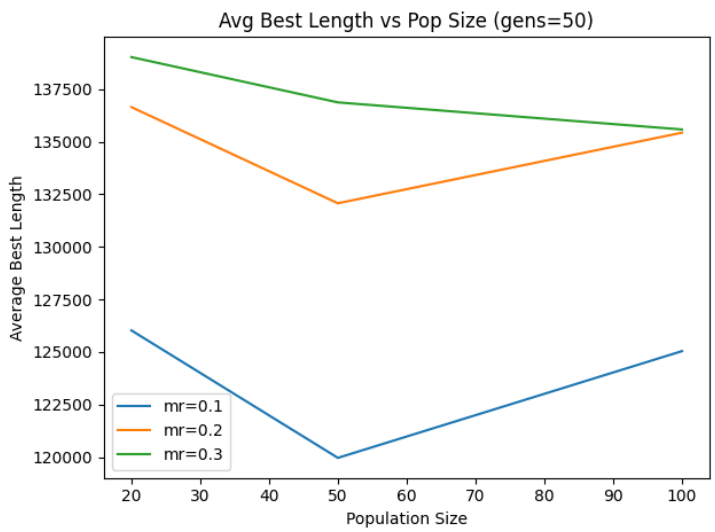

Figure 1: Average Best Tour Length vs Population Size (50 generations; mutation rates = 0.1, 0.2, 0.3).

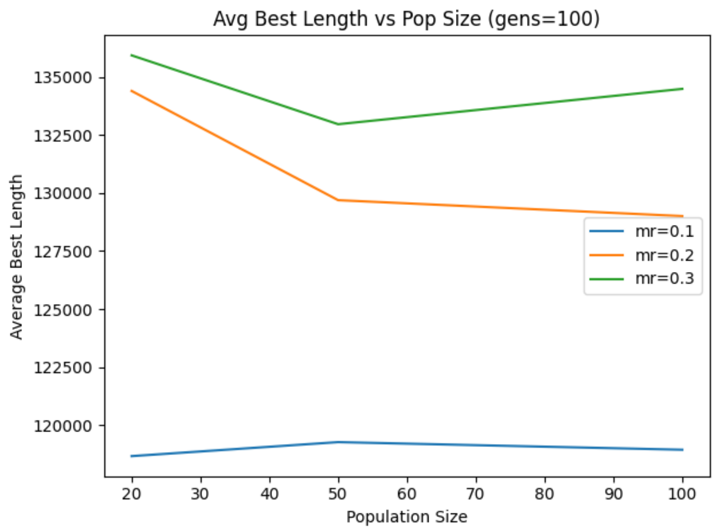

Figure 2: Average Best Tour Length vs Population Size (100 generations; mutation rates = 0.1, 0.2, 0.3).

3.1 Parameter Selection

The best average performance (119 267.4) was achieved with:

$$\text{pop\_size} = 50, \quad \text{generations} = 100, \quad \text{mutation\_rate} = 0.1.$$

A final GA run with these settings yields:

```
Best tour length: 122392.68
Best tour: [91, 48, 16, 12, ..., 72]
```

4 Conclusion

We implemented a GA for the TSP and performed systematic parameter studies. The combination of population size 50, mutation rate 0.1, and 100 generations produced the best results (average length 119 267.4).

Future work may include adaptive parameter control, island-model parallel GAs, or integration of stronger local searches (e.g., 3-opt).

5 References

References

[1] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA J. Comput.*, 3(4):376–384, 1991.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

A Appendix: GA Pseudocode

Algorithm 1 Genetic Algorithm for TSP

Initialize population P of size N with random permutations
for $g = 1$ to G **do**
    Evaluate fitness $f(\pi)$ for each $\pi \in P$
    Select top E elites to carry over
    Initialize new population $P' \leftarrow \{\}$
    while $|P'| < N$ **do**
        $p_1, p_2 \leftarrow$ TournamentSelect(P, k)
        $(c_1, c_2) \leftarrow$ PMX(p_1, p_2)
        Mutate(c_1, rate μ); Mutate(c_2, rate μ)
        $P' \leftarrow P' \cup \{c_1, c_2\}$
    end while
    $P \leftarrow$ SelectBest($P' \cup$ elites, N)
end for
return best tour in P