Solving the Traveling Salesman Problem Using a Genetic Algorithm

Aya Abdellatif Mohammed
Enas Ikram Girgis

May 6, 2025

Abstract

The Traveling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem that seeks the shortest possible route visiting each city exactly once and returning to the starting point. Due to the exponential growth in route possibilities as the number of cities increases, solving TSP becomes computationally challenging [1]. This paper explores the use of the Genetic Algorithm (GA), a biologically inspired search heuristic, to find near-optimal solutions to TSP. The algorithm evolves a population of candidate routes through selection, crossover, and mutation operators. Our implementation demonstrates that GA provides high-quality routes with reduced computation time.

1 Introduction

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization problem that requires finding the shortest possible route that visits each city exactly once and returns to the starting point. This problem is not only theoretically significant in computer science and operations research but also has practical applications in logistics, manufacturing, and route planning systems [2]. TSP becomes increasingly complex as the number of cities grows, with the total number of possible routes increasing factorially, making brute-force methods computationally infeasible for large instances.

Numerous approaches have been developed to solve TSP efficiently, ranging from exact algorithms to approximation and heuristic-based methods:

- **Dynamic Programming**: One of the first approaches to solve TSP exactly, proposed by Richard Bellman, uses a recursive formulation to build up solutions but becomes impractical due to exponential time and space complexity [3].
- **Branch and Bound**: This method reduces the search space using bounds but still suffers from performance issues on large datasets.
- **Heuristic and Metaheuristic Algorithms**: These provide approximate solutions in reasonable timeframes and include Simulated Annealing, Ant Colony Optimization, and Genetic Algorithms.

Among these, the Genetic Algorithm (GA) stands out as a biologically inspired metaheuristic that efficiently explores large solution spaces by mimicking the process of natural selection. GA evolves a population of candidate solutions through operators such as selection, crossover, and mutation, gradually improving the quality of solutions across generations. Its flexibility and efficiency make it well-suited for solving NP-hard problems like TSP. Given its strong performance and adaptability, this paper focuses on using the Genetic Algorithm to find near-optimal solutions to the Traveling Salesman Problem.

2 Methodology

Genetic algorithm is a type of evolutionary algorithm inspired by the processes of natural selection and genetics. It is well-suited for solving the Traveling Salesman Problem (TSP). The algorithm operates as follows:

1. **Initialization**: Generate an initial population of candidate solutions, where each solution represents a possible ordering of cities to visit.
2. **Evaluation**: Evaluate the fitness of each candidate solution by calculating the total distance of the TSP route.
3. **Selection**: Select a set of parent solutions from the current population based on their fitness. Solutions with better fitness have a higher probability of being selected.
4. **Crossover**: Create new offspring solutions by combining the genetic material (city orderings) of the selected parents.
5. **Mutation**: Introduce random changes to the offspring solutions to maintain diversity and explore new regions of the solution space. Swap mutation and inversion mutation are commonly used mutation operators.
6. **Population Management**: Replace some solutions in the current population with the newly generated offspring solutions.
7. **Termination**: Repeat steps 2-6 for a certain number of generations or until a termination condition is met (e.g., reaching a maximum number of generations).
8. **Solution Extraction**: Once the algorithm terminates, extract the best solution found (i.e., the shortest route) from the final population.

2.1 Brief Analysis of Genetic Algorithm

- **Time Complexity**: Depends on the population size P, number generations G, and number of cities N: $(P \cdot G \cdot N)$.
- **Space Complexity**: $O(P \cdot N)$ storing the population of candidate solutions.

The actual runtime may vary based on implementation details, crossover/mutation strategies, and stopping conditions.

3 Implementation and Technical Results

The implemented solution uses a Genetic Algorithm (GA) to solve the Traveling Salesman Problem (TSP) over a set of ten cities with predefined coordinates. The algorithm was developed using Python and executed in Jupyter Notebook. The implementation can be found here.

The TSP aims to determine the shortest possible route that visits all cities exactly once and returns to the starting point. In our implementation, cities are represented by their coordinates, and distances are computed using the Euclidean formula. The GA incorporates standard components:

- Selection: Roulette Wheel
- Crossover: Order-based
- Mutation: Swap-based

The algorithm was initially run with the following parameters:

- Population Size: 250

- Crossover Rate: 0.8
- Mutation Rate: 0.2
- Number of Generations: 200

These parameters were chosen to balance exploration (via mutation) and exploitation (via selection and crossover), aiming to guide the search toward high-quality solutions without premature convergence. Figure 1 shows an example of a generated tour.

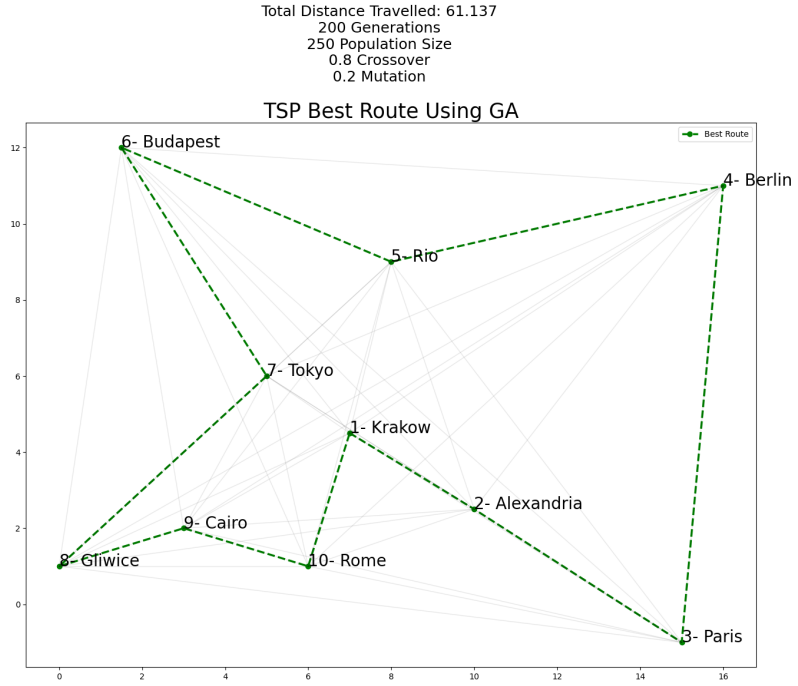

Figure 1: Solution Example

3.1 Comparison with Exact Methods

Although exact algorithms such as brute-force, dynamic programming, and branch and bound guarantee the optimal TSP solution, they are impractical for larger instances due to their factorial or exponential time complexity. In contrast, the Genetic Algorithm, as shown in Table 1, offers a heuristic approach that produces high-quality solutions efficiently, making it suitable for larger-scale problems where exact solutions are computationally infeasible.

Table 1: Comparison of Algorithms

Algorithm	**Complexity**	**Approach**
Brute-force	$O(n!)$	Exact
Dynamic Programming	$O(n^2 \cdot 2^n)$	Exact
Branch and Bound	$O(n!)$	Exact
Genetic Algorithm	Depends on params	Heuristic

4 Conclusion

In conclusion, the Genetic Algorithm offers an effective and flexible heuristic for solving the Traveling Salesman Problem. By simulating natural selection through crossover, mutation, and selec-

tion, the GA efficiently explores the search space and identifies near-optimal routes. While it does not guarantee the exact optimal solution, our results show that it consistently finds high-quality solutions close to the best possible. However, GA performance is highly sensitive to parameter settings—particularly population size and crossover rate—which must be carefully tuned to balance exploration and exploitation.

References

1. TutorialsPoint. *Travelling Salesman Problem using Genetic Algorithm* Accessed: 2025-05-05. https://www.tutorialspoint.com/travelling-salesman-problem-using-genetic-algorithm.
2. Applegate, D. L., Bixby, R. E., Chvátal, V. & Cook, W. J. *The Traveling Salesman Problem: A Computational Study* (Princeton University Press, Princeton, NJ, 2006).
3. Bellman, R. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM (JACM)* **9,** 61–63 (1962).