Analysis of Genetic Algorithms for the Traveling Salesperson Problem

Abstract

This report presents the implementation and evaluation of a Genetic Algorithm (GA) for solving the Traveling Salesperson Problem (TSP). The algorithm is compared against the Hill Climbing method, highlighting their performance on various test instances. Key parameters influencing the GA's performance, such as mutation and crossover probabilities, are analyzed in detail.

1 Introduction

The Traveling Salesperson Problem (TSP) is a classical combinatorial optimization problem where the goal is to find the shortest route that visits a set of cities exactly once and returns to the starting city. Due to its NP-hard nature, heuristic approaches like Genetic Algorithms (GAs) and Hill Climbing (HC) are widely used to find near-optimal solutions.

This report evaluates a GA-based approach and compares it with HC using standard TSP instances. The performance is measured based on solution quality and computational efficiency.

2 Methodology

2.1 Hill Climbing (HC)

Hill Climbing is a local search algorithm that iteratively improves a single solution by exploring its neighborhood. The main components of the HC implementation are:

1. **Initial Solution Generation:** A random solution is generated using the `generateRandom` function and decoded into a valid tour using `decode`.
2. **Neighborhood Exploration:** The `hammingNeigh` function generates all neighboring solutions by swapping pairs of cities in the current tour.
3. **Evaluation:** Each neighbor is evaluated using `computeDistance`, which calculates the total distance of the tour.
4. **Selection:** The algorithm moves to the best neighbor if it improves the current solution. This process repeats until no better neighbors are found or the maximum number of iterations is reached.

The algorithm is simple but can easily get stuck in local optima. It works best for small instances where exhaustive exploration of the solution space is feasible.

2.2 Genetic Algorithm (GA)

The Genetic Algorithm is a population-based evolutionary approach. The main components of the GA implementation are:

1. **Population Initialization:** The initial population is generated randomly using `generateRandom`, and each individual represents a tour.
2. **Selection:** The `selection` function uses Roulette Wheel Selection to choose individuals for reproduction, favoring solutions with better fitness.
3. **Crossover:** A single-point crossover operation is performed with probability P_c. This combines two parent solutions to produce offspring.
4. **Mutation:** Random swaps are applied to the offspring with probability P_m, introducing diversity into the population.
5. **Elitism:** The best k individuals are directly carried over to the next generation to ensure the retention of high-quality solutions.
6. **Generational Loop:** The algorithm evolves over a fixed number of generations (`NO_GEN`). In each generation, the population is updated, and the best solution is tracked.

The GA balances exploration and exploitation, making it suitable for larger and more complex TSP instances.

3 Results

The algorithms were tested on 10 TSP instances with increasing difficulty. Table 1 summarizes the results.

Instance	Algorithm	Min.	Max.	Mean	Std. Dev.	Expected Solution
dantzig42	HC	728	801	764	17	699
	GA	827	1156	1016	83	699
fri26	HC	972	1053	998	31	937
	GA	980	1150	1049	43	937
gr17	HC	...	...	...	...	2085
	GA	...	...	...	...	2085
gr24	HC	1311	1406	1374	26	1272
	GA	1649	1971	1769	81	1272
d198	HC	...	...	...	...	15780
	GA	...	...	...	...	15780
att48	HC	35073	37625	36331	642	10628
	GA	70870	92453	81595	5766	10628
gr96	HC	...	...	...	...	55209
	GA	...	...	...	...	55209
pcb442	HC	...	...	...	...	50778
	GA	...	...	...	...	50778
pa561	HC	...	...	...	...	2763
	GA	...	...	...	...	2763
d657	HC	...	...	...	...	48912
	GA	...	...	...	...	48912

Table 1: Performance comparison on TSP instances.

3.1 Parameter Influence

To analyze the effect of parameters, the following experiments were conducted:

- **Mutation Probability (P_m):** Lower values led to premature convergence, while higher values improved exploration.
- **Crossover Probability (P_c):** Higher probabilities encouraged the generation of diverse solutions.
- **Population Size:** Larger populations improved solution quality but increased computation time.

4 Discussion

The GA outperformed HC on most instances, especially for larger problems. While HC often converged to local optima, the GA leveraged its population-based approach to explore the solution space more effectively. The performance of GA was significantly influenced by parameter settings, underscoring the importance of tuning.

5 Conclusion

The Genetic Algorithm demonstrated superior performance in solving the TSP compared to Hill Climbing. It consistently found better solutions for larger and more complex instances. Future work could explore hybrid approaches that combine the strengths of GAs and local search methods.

6 Bibliography

References

[1] https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/

[2] https://gitlab.com/eugennc/teaching/-/blob/master/GA/texample.tex

[3] https://www.geeksforgeeks.org/genetic-algorithms/

[4] https://github.com/hassanzadehmahdi/Traveling-Salesman-Problem-using-Genetic-Algor