

# Metaheuristic Approaches for Optimizing the MinMax Multiple Traveling Salesman Problem

Ioana-Ștefana Hanus  
 Master's Student  
 Alexandru Ioan Cuza University  
 Iași, Romania  
 ioana.hanus@student.uaic.ro

## I. INTRODUCTION

The MinMax Multiple Traveling Salesman Problem (mTSP) is a complex combinatorial optimization challenge in which multiple salesmen must visit a set of cities exactly once, returning to a central depot, with the objective of minimizing the maximum tour length among all routes. This NP-hard variant of the classical Traveling Salesman Problem (TSP) requires advanced algorithmic strategies to compute near-optimal solutions efficiently. In this study, we explore and compare two bio-inspired metaheuristics—Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO)—through custom implementations developed in Python.

## II. RELATED WORK

Metaheuristics such as ACO and PSO have been widely studied for solving TSP variants. ACO, inspired by the foraging behavior of ants, has been adapted to multi-agent routing problems [3], [4]. PSO simulates social behavior among swarming agents and has also shown potential in TSP contexts [5], albeit with higher variability. The reference work [1] served as a benchmark source for performance comparison, while additional inspiration for implementation details was drawn from algorithm tutorials, including a visual simulation of ACO [2].

## III. METHODS

### A. Ant Colony Optimization (ACO)

The implemented Ant Colony Optimization algorithm employs a population of 50 ants and runs over 1000 iterations. Initially, all edges are assigned a uniform pheromone level. During each iteration, ants construct solutions by probabilistically selecting the next city based on a combination of pheromone intensity ( $\tau_{ij}$ ) and heuristic visibility ( $\eta_{ij}$ ), governed by parameters  $\alpha$  and  $\beta$ :

$$p_{ij} = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{k \in N_i} (\tau_{ik})^\alpha \cdot (\eta_{ik})^\beta}$$

where  $\eta_{ij}$  is the inverse of the distance between cities  $i$  and  $j$ , and  $N_i$  is the set of unvisited neighboring cities.

Each tour undergoes a 2-opt local search, with a 5% chance of perturbing the solution via random swaps. Every 50 iterations, a 3-opt-like procedure further refines each tour by exploring edge-rewiring among triplets of cities, improving solution quality and mitigating premature convergence.

Pheromone trails are updated based on both individual ant performance and the global best solution, following:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^n \Delta \tau_{ij}^k, \quad \text{where} \quad \Delta \tau_{ij}^k = \frac{Q}{L_k}$$

Here,  $L_k$  denotes the length of ant  $k$ 's tour and  $Q$  is a constant representing the total amount of pheromone deposited. Parameters  $\alpha$  and  $\beta$  are dynamically tuned:  $\alpha$  increases over time to enhance exploitation, while  $\beta$  decreases to reduce initial greediness. This dynamic adjustment proved essential in balancing exploration and convergence.

### B. Particle Swarm Optimization (PSO)

The implemented Particle Swarm Optimization algorithm consists of five swarms, each containing 50 individuals. Each individual represents a candidate tour as a permutation of the cities (excluding the depot), and evolves by applying a velocity composed of city swaps derived from three influences: inertia, cognitive memory, and social interaction.

During each of the 1000 iterations, individuals update their positions by combining:

- **Inertia vector**, preserving part of the current velocity;
- **Cognitive vector**, based on the best solution found by the individual itself;
- **Social vector**, based on the best-known solution in the entire swarm.

The velocity is formed by selecting swap operations from each vector, scaled by weights (inertia, cognitive,

and social), and then shuffled to encourage diversity. The resulting swaps are applied to update the individual’s position.

Each individual updates its personal best if a new better solution is found. The global best is updated accordingly. When a swarm stagnates—i.e., no improvement occurs over a fixed number of iterations—a hill-climbing procedure is triggered on a portion of its individuals. This heuristic performs targeted swaps to refine the solution locally.

The inertia weight is gradually decreased from 1.0 to a minimum threshold of 0.4 to balance exploration and exploitation. Parameters such as the cognitive and social weights were empirically tuned to 1.5 each, following extensive experimentation.

#### IV. RESULTS

The algorithms were tested on four standard TSP instances: *eil51*, *berlin52*, *eil76*, and *rat99*, each with 2, 3, 5, and 7 salesmen configurations. Each run was repeated 30 times.

ACO consistently produced solutions closer to known reference values [1], with lower standard deviation. However, its execution time was higher due to the complexity of the 3-opt-like search. PSO executed faster but displayed greater solution variability and was competitive only in some cases.

Among the four TSP instances tested, we present detailed results for the *berlin52* dataset, as it is representative in terms of instance complexity and outcome variability across methods.

TABLE I: Performance summary on the *berlin52* test instance

#Sm.	Algorithm	Mean	SD	Min	Max
2	ACO	4342.84	27.35	4288.36	4361.19
	PSO	4439.50	328.38	4128.93	4893.80
	Reference	[4049.05, 4110.21]	—	—	—
3	ACO	2863.12	22.69	2842.18	2877.79
	PSO	3027.85	78.27	2900.36	3112.93
	Reference	[2753.63, 3244.37]	—	—	—
5	ACO	2198.89	75.67	2112.08	2296.49
	PSO	2015.23	220.41	1791.31	2295.35
	Reference	[1671.69, 2441.39]	—	—	—
7	ACO	2508.62	4.48	2504.96	2514.17
	PSO	1595.23	108.32	1452.56	1737.60
	Reference	[1272.06, 2440.92]	—	—	—

On this instance, ACO generally produced more stable solutions with lower standard deviations, especially as the number of salesmen increased. However, PSO showed stronger performance in specific configurations (e.g., 5 and 7 salesmen), albeit with higher variability.

The ACO results approached known reference bounds more consistently, while PSO yielded faster but less reliable convergence. An unexpected dip in PSO performance was observed for the 3-salesmen case on the *berlin52* instance, with worse average results than in the 2- and 5-salesmen cases. This suggests possible sensitivity to parameter settings or early stagnation in that configuration, despite consistent behavior across runs.

#### V. DISCUSSION AND CONCLUSIONS

The presented experiments show that ACO generally outperforms PSO in terms of solution quality and stability, though it requires longer runtimes. PSO offers faster execution, making it appealing for applications where quick, approximate solutions are acceptable.

Beyond performance, a key contribution of this work lies in its educational potential. Both algorithms are implemented in Python with clear, modular code designed for ease of understanding and adaptation. This makes them valuable teaching tools for courses in artificial intelligence, optimization, and algorithm design. The Traveling Salesman Problem (TSP) and its variants remain fundamental examples in combinatorial optimization education, and our implementations provide hands-on resources that bridge theory and practice, directly supporting the conference theme, *Computer Science: a Catalyst for Educational Change*.

Future work will focus on:

- Developing hybrid methods that combine ACO’s solution quality with PSO’s efficiency;
- Improving ACO’s runtime by optimizing the 3-opt local search procedure;
- Investigating alternative neighborhood structures to enhance PSO performance;
- Packaging the codebase into reusable, well-documented modules to facilitate adoption in educational settings.

#### REFERENCES

- [1] R. Necula, M. Breaban and M. Raschip, “Tackling the Bi-criteria Facet of Multiple Traveling Salesman Problem with Ant Colony Systems,” *2015 IEEE ICTAI*, Vietri sul Mare, 2015, pp. 873–880.
- [2] Simulife Hub, “Ant colony optimization algorithm,” YouTube, <https://www.youtube.com/watch?v=u7bQomllcJw>
- [3] J. Pan and D. Wang, “An ant colony optimization algorithm for multiple travelling salesman problem,” *IEEE ICICIC*, 2006.
- [4] L.-C. Lu and T.-W. Yue, “Mission-oriented ant-team ACO for min–max MTSP,” *Applied Soft Computing*, vol. 76, pp. 436–444, 2019.
- [5] H. Zhou, M. Song, and W. Pedrycz, “A comparative study of improved GA and PSO in solving multiple traveling salesman problem,” *Applied Soft Computing*, vol. 64, pp. 564–580, 2018.