BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

Rainbow ACO: Novel Ant Colony Optimization

Ivan Burlutskyi*

Abstract
The goal of this work is to propose a novel implementation of the Ant Colony Optimization (ACO) algorithm, termed **Rainbow ACO**. Although primarily designed to solve the Traveling Salesman Problem (TSP), the versatility of ACO allows the proposed method to be applied to a wide range of optimization problems. To achieve high efficiency and superior performance, Rainbow ACO integrates six distinct techniques aimed at enhancing both the exploration and exploitation capabilities of the algorithm. The algorithm was tested on complex TSP scenarios and demonstrated promising results.

*xburlu00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Ant Colony Optimization (ACO) is a powerful metaheuristic for solving complex combinatorial problems, including the Traveling Salesman Problem (TSP), vehicle routing, scheduling, network [1] optimization, and chip design [1]. Despite its versatility, traditional ACO suffers from many issues like premature convergence and a suboptimal exploration-exploitation balance. To address these limitations, we introduce **Rainbow ACO**, a novel implementation that integrates six distinct enhancement strategies. The proposed hybrid approach aims to boost the algorithm's exploitation capabilities while maintaining the exploration of promising solutions.

Rainbow ACO is primarily designed to solve the Traveling Salesman Problem (TSP). The TSP is a classic combinatorial optimization problem where the task is to find the shortest tour that visits each city exactly once and returns to the origin city. It is NP-hard, meaning that as the number of cities increases, the computational complexity grows exponentially, making it impractical to solve by brute-force methods for large instances.

The most well-known basic implementations of ACO are the Min-Max Ant System (MMAS) [2] and the Ant Colony System (ACS) [3]. Modern frameworks for solving average-sized TSP instances (ranging from 0 to 1,000 cities) include DeepACO, Symbiotic Organism Search ACO [4], Heuristic Smoothing ACO [5], and Lévy-flight ACO [6]. Additionally, for large to extra-large TSP instances (ranging from 1,000 to 1,000,000 cities), Partial ACO [7] and Restricted ACO [8] are commonly employed due to their ability to manage computational complexity and memory limitations.

Rainbow ACO includes six main features to enhance performance: a restricted pheromone matrix [8], the Lin-Kernighan heuristic [9], an improved Convex and Convey Two-Opt operator [5], Lambda-branching stagnation protection [2], dynamic lower and upper bounds for pheromone values [2], and a hybridization with the Grasshopper Optimization Algorithm (GOA) [10].

Rainbow ACO Achievements:

- Automatic tuning of the ACO parameters α and β, eliminating the need for manual adjustment.
- Adaptation of the state-of-the-art local search operator – the Lin-Kernighan Heuristic – to the pheromone context, along with the incorporation of advanced k-Opt operations.
- Implementation of stagnation protection mechanisms – recovering from premature convergence and continue effective exploration.
- Low memory footprint and adaptable performance — both time and memory complexities scale linearly with problem size.

2. Poster Commentary

All techniques employed in Rainbow ACO are illustrated in Fig. 1. In the following commentary, we

focus on discussing the most impactful components of Rainbow ACO.

2.1 Parameter Tuning

In traditional ACO implementations, the parameters α and β are typically set to constant values, requiring manual tuning for each specific TSP instance. To overcome this limitation, Rainbow ACO employs the Grasshopper Optimization Algorithm (GOA) to dynamically tune the α and β parameters during the optimization process.

In this approach, each grasshopper represents a candidate solution defined by a pair (α,β) for the ant colony. GOA simulates the natural behavior of grasshoppers, capturing social interactions, attraction toward optimal food sources, and repulsive forces near the best solutions to encourage exploration. An illustrative conceptual model of the grasshoppers' interactions is presented in Fig. 2. The evolution of α and β values over the course of the Rainbow ACO run is shown in Fig. 3.

2.2 Local searches

In approximately 90% of existing ACO implementations, 2-Opt or 3-Opt local searches are used to improve the ants' solutions. In Rainbow ACO, two local search operators are utilized: Convey and Convex 2-Opt and Lin Kernighan heuristic (LKH). The purpose of Convex 2-Opt is to enhance the paths of all ants, while an adapted LKH, powered by k-Opt, is designed to enhance the best local solution. The Convey and Convex 2-Opt operators aim to smooth the search space (Figure 6). The adapted LKH selects k suspicious edges based on their pheromone levels, prioritizing edges with the lowest pheromone values, and then performs the k-Opt operation (Figure 4).

2.3 Stagnation protection

Due to the evaporation factor ρ, the solution space in classic and modern ACO implementations can shrink too rapidly. As a result, ants are prone to being trapped in local optima, significantly limiting exploration. To mitigate this challenge, Rainbow ACO incorporates a stagnation protection mechanism. When stagnation is detected, Rainbow ACO reinitializes all edges with the maximum pheromone value, thereby encouraging renewed exploration and preventing premature convergence. This mechanism is illustrated in Figure 4.

2.4 Experiments

The experiments were conducted using the Art TSP dataset provided by the University of Waterloo. Rainbow ACO demonstrated competitive performance compared to existing methods such as Partial ACO and Restricted ACO (Figure 8). Moreover, the algorithm exhibited linear time complexity as the size of the TSP instances increased (Figure 9). Illustrative representations of the solutions obtained are presented in (Figure 10).

3. Conclusions

Although Rainbow ACO did not drastically outperform existing solutions, it demonstrated promising and competitive results.

We continue to enhance the performance of Rainbow ACO by implementing efficient data structures to accelerate local search operations and by developing a more complex and adaptable solution construction mechanism. Furthermore, we plan to extend Rainbow ACO to other optimization problems in order to evaluate its effectiveness across a broader range of scenarios.

References

[1] Bachir Benhala. An improved aco algorithm for the analog circuits design optimization. *INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING*, 10:128–133, 01 2016.

[2] Thomas Stützle and Holger H. Hoos. – ant system. *Future Generation Computer Systems*, 16(8):889–914, June 2000.

[3] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[4] Yong Wang and Zunpu Han. Ant colony optimization for traveling salesman problem based on parameters optimization. *Applied Soft Computing*, 107:107439, August 2021.

[5] Wei Li, Cancan Wang, Ying Huang, and Yiuming Cheung. Heuristic smoothing ant colony optimization with differential information for the traveling salesman problem. *Applied Soft Computing*, 133:109943, January 2023.

[6] Yahui Liu, Buyang Cao, and Hehua Li. Improving ant colony optimization algorithm with epsilon greedy and levy flight. *Complex amp; Intelligent Systems*, 7(4):1711–1722, March 2020.

[7] Darren M. Chitty. *Applying ACO to Large Scale TSP Instances*, page 104–118. Springer International Publishing, September 2017.

[8] Joshua Peake, Martyn Amos, Paraskevas Yiapanis, and Huw Lloyd. Scaling techniques for parallel ant colony optimization on large problem instances. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 47–54. ACM, July 2019.

[9] Keld Helsgaun. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, 1(2–3):119–163, July 2009.

[10] Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software*, 105:30–47, March 2017.