Ant Colony Optimization for the Mutiobjective Shortest Paths Problem

Luis Burgueño[1] and Héctor Gutiérrez[1]

Tecnológico de Monterrey

Abstract. Ant Colony Optimization (ACO) has established itself as a powerful metaheuristic for solving combinatorial optimization problems, including applications in Multi-Objective Optimization (MOO). The Multiobjective Shortest Path Problem (MSPP) extends the classic shortest path problem by optimizing multiple conflicting objectives simultaneously, such as minimizing travel time and cost or maximizing reliability. This work presents an Ant Colony Optimization algorithm tailored for MSPP. The algorithm utilizes pheromone trails and heuristic values to guide ants toward optimal solutions while balancing multiple objectives. Experimental results demonstrate the effectiveness of the proposed ACO approach in generating high-quality Pareto-optimal solutions. Compared to a baseline random search algorithm, ACO consistently produced a larger number of high-quality, non-dominated solutions. While ACO required slightly more computational time (averaging 23 seconds per run), the trade-off was justified by the substantial improvement in solution quality. The generated Pareto front effectively minimized all objectives, showcasing ACO's superiority in handling the complexities of MSPP.

Keywords: Multi Objective Optimization · Multiple Shortest Paths · Ant Colony Optimization · Swarm Intelligence

1 Introduction

Ant Colony Optimization (ACO) algorithms, as described by [1], have proven to be effective strategies for solving a variety of problems, including those in Multi-Objective Optimization (MOO). Since population-based algorithms are well-suited to problem areas like MOO, it is understandable that the research community has been interested in them. Generally, the MOO, used to find a set of optimal solutions, is presented as follows:

$$\begin{aligned}
&y = f(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\},\\
&e(x) = \{e_1(x), e_2(x), \ldots, e_k(x)\} \geq 0,\\
&x = (x_1, x_2, \ldots, x_n) \in X,\\
&y = (y_1, y_2, \ldots, y_m) \in Y
\end{aligned}$$

Where X represents the decision space consisting of a set of n decision variables, and Y denotes the objective space comprising a set of mm objective functions with k constraints.

García-Martínez et al. [2] provided a comprehensive review and analysis of various Multi-Objective Ant Colony Optimization (MOACO) methods. They summarized all known MOACO algorithms and introduced a straightforward taxonomy, classifying them based on the number of pheromone and heuristic matrices. Their study focused particularly on an empirical evaluation of these algorithms using the Multi-Objective Traveling Salesman Problem (MOTSP). This analysis removed many MOACO algorithms from their original contexts, concentrating instead on adapting them to the MOTSP.

Similar to the single-objective shortest path problem, the multi-objective shortest path problem (MSPP) has been extensively explored by researchers across fields such as optimization, traffic and transportation route planning [3], and information and communication network design. The MSPP expands on the classic shortest path problem by seeking efficient paths that balance two or more often conflicting objectives. In communication networks, the goal can be to minimize delay while maximizing throughput, or in transportation planning, to find routes that reduce travel cost, path length, and travel time. Unlike single-objective optimization, multi-objective optimization seeks not a single optimal solution but rather a set of solutions that effectively balance all objectives.

Because Ant Colony Optimization (ACO) performs so well in finding solutions, it has gained a lot of interest from researchers and is now widely used in various areas, including finding the shortest route. ACO has become a popular focus in the study of intelligent optimization algorithms, with two main mechanisms: the probabilistic decision rule and the pheromone update rule. The decision rule helps guide the "ants" to create solutions in a probabilistic way, while the pheromone update rule allows the ants to gather knowledge about the problem. Recently, some researchers have developed ACO algorithms specifically for multi-objective problems, mostly focusing on specific tasks like scheduling, vehicle routing, or portfolio selection, among others.

This work uses the ACO method to solve the multiobjective shortest paths problem and it is structured as follows. Section 2 defines what the Multiobjective Shortest Path Problem and Ant Colony Optimization are, along with some work done in the subjects. Section 3 explains the approach used. Section 4 presents the results obtained. Finally, Section 5 draws conclusions and introduces future work to be done on the subject.

2 Background

2.1 Multiobjective Shortest Path

The Multiobjective Shortest Path Problem (MSPP) falls within a category of problems that have polynomial-time algorithms for their original, single-objective forms. This characteristic enables optimizers to adopt a two-phase strategy, which has demonstrated efficient performance in a bi-objective context [4]. This strategy leverages the fact that some solutions within the Pareto set can be obtained by scalarizing the objectives, assigning weights to each. As a result, part of the Pareto set can be identified using polynomial time. An optimizer can narrow down the search space when seeking non-supported solutions by having these solutions (or a subset of them) beforehand. A version of these was defined in [5] in which:

$$\min_{r \in R_{s,t}} \left(z^1(r), \ldots, z^K(r)\right) \quad \forall t \in V \setminus \{s\}$$

$E(R_{s,\cdot})$ denotes the *maximal complete set of efficient paths* in $R_{s,\cdot}$, for a given source vertex s, i.e., the set of all paths of $R_{s,\cdot}$ in which each path corresponds to a non-dominated point. Several distinct efficient paths r_1^*, r_2^*, r_3^* can correspond to the same non-dominated point $z(r_1^*) = z(r_2^*) = z(r_3^*)$ in the objective space. The paths r_1^*, r_2^*, r_3^* are said to be *equivalent* in the objective space.

The *minimal complete set of efficient paths* is a subset of $E(R_{s,\cdot})$ that contains no equivalent solutions, and for any $r \in E(R_{s,\cdot})$, there exists r' in the minimal complete set such that r and r' are equivalent.

Research on this problem includes both exact and heuristic methods, including evolutionary and ant colony optimization techniques. Raith and Ehrgott [6] recently compared various exact methods for the Bi-objective Shortest Path (BSP) problem, demonstrating that the two-phase method can solve instances with up to 21,000 nodes in under 30 seconds and up to 300,000 nodes in less than 40. However, this algorithm has yet to be extended to tackle problems with more than two objectives.

Several Evolutionary Multiobjective Optimization (EMO) algorithms have been proposed for the Multiobjective Shortest Path Problem (MSPP) [7], and they share some common features. Firstly, all these algorithms incorporate an elitist approach, maintaining two separate populations throughout their execution. Secondly, they all apply binary tournament selection. Lastly, while their crossover operators vary in how they exchange genes, each algorithm uses a one-point crossover method.

2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic inspired by biological systems, particularly the concept of swarm intelligence—where groups communicate and coordinate actions without a central authority, using local, physical signals. This type of indirect communication, known as *stigmergy* [8], is seen in ants, where the signal takes the form of pheromone trails. As ants search for food, they deposit pheromones along their paths, creating a trail for others to follow. Since the pheromone gradually evaporates, shorter paths retain stronger signals, encouraging a natural convergence toward the most efficient routes.

To replicate this natural behavior, the Ant System (AS) metaheuristic was introduced [9], and later enhanced into the Ant Colony System (ACS) [10] and the MAX-MIN Ant System (MMAS). In these algorithms, agents known as ants iteratively construct solutions. Each ant builds a solution by evaluating its current position and calculating the objective value of its path so far. The ant makes decisions based on available information about possible paths at each step. This information comes both from other ants, represented by the pheromone levels on each path, and from its heuristic knowledge.

Its main characteristic is that, at each iteration, the pheromone values are updated by all the ants that have built a solution in the iteration itself. The pheromone τ_{ij} associated with the edge joining cities i and j is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \sum_k \Delta\tau_{ij}^k$$

where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^k$ is the quantity of pheromone laid on edge (i,j) by ant k.

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases}$$

where Q is a constant, and L_k is the length of the tour constructed by ant k.

In a solution, ants select the following city to be visited through a stochastic mechanism. When an ant k is in city i, the probability of going to city j is given by:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N(s_i^k)} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{if } j \in N(s_i^k), \\ 0 & \text{otherwise,} \end{cases}$$

where $N(s_i^k)$ is the set of feasible components; that is, edges (i,l) where l is a city not yet visited by the ant k. The parameters α and β control the relative importance of the pheromone information τ_{ij} and the heuristic information η_{ij}, which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

where d_{ij} is the distance between cities i and j.

The MOACO algorithm introduced by Häckel et al. [11] employs a multi-colony strategy, where each colony works with separate weight intervals. A single, global pheromone matrix is used in the algorithm. It incorporates dominance-based selection and an update-by-origin method. Heuristic information is provided by a Dynamic Programming algorithm called the Look-Ahead Heuristic (LAH), developed by the authors. Their experiments demonstrate that incorporating LAH enhances overall performance and that their ACO algorithm produces solutions well-distributed across the three-dimensional objective space compared to an unspecified dynamic multi-objective algorithm.

3 Algorithmic Approach

The goal of the MSPP is to find a set of paths between two nodes in a graph ($G = (V, E)$ where V is the set of vertices or nodes and E is the set of Edges) that optimizes multiple objectives. The idea behind this approach is that ants search for the optimal paths by balancing multiple objectives with the help of a simple heuristic (the inverse of the average of the costs for each path). This algorithm creates a Pareto-optimal set that represents the best trade-offs among objectives.

3.1 Algorithm Overview

The following algorithms repeats for a set number of iterations.

- **Initialization:** Initialize the pheromone matrix to store the levels on all edges.
- **Path Building:** Each ant tries to build a path from the starting point to the end node. It probabilistically chooses the next node at each step based on pheromone intensity from the previous ants and heuristic values.
- **Pareto Dominance Check:** After all ants have made their path, check if any path is Pareto-optimal. Only those paths not worse in all objectives compared to others (non-dominated paths) are stored.
- **Pheromone Update:** Reduce pheromone levels on all edges by a fixed evaporation rate and increase pheromone levels on the non-dominated paths.

Here's a pseudocode of how the algorithm works:

```
function ACO(Graph, start_node, end_node, num_ants, num_iterations, alpha, beta,
    evaporation rate):

Initialize pheromone matrix
Initialize pareto_archive to store solutions

for iteration=1 to num_iterations:
    Create a set of num_ants Ants

    for each ant in set of Ants:
        Initialize path and cost
        while current_node != end_node:
            Choose next_node based on probabilities where for each neighbor
                prob=(pheromone_level ¨** alpha) * (heuristic ** beta)
            Update path and cost
        end while

        Check if ant's path is non-dominated:
        if non-dominated with respect to current Pareto archive:
            Remove dominated solutions if any
            Add ant's path to Pareto archive if it's not there

    end for
```

```
        Apply evaporation to all pheromone levels
        Reward shorter paths

    return pareto_archive
    end function
```

4 Experimental Results

To obtain results a random graph with 100 nodes and 3 objective costs (between 1 and 100) was built. It was tested for 1,000 iterations but the number and quality of solutions did not change past $\tilde{1}00$, therefore all experiments were done with 100 iterations. Parameters used for ACO can be seen on Table 1.

Parameter	Value
Number Of Ants	100
Iterations	100
Pheromone Influence	1
Heuristic Information Influence	2
Evaporation Rate	0.5
Pheromone Initial Level	0.1

Table 1. Parameters for ACO

In order to assess the performance of the model, a comparison was made with a random search algorithm with 30 runs. Figure 1 shows a boxplot comparing the number of solutions and time taken to obtain them with ACO and Random Search. As expected, Random Search takes a lot less time to obtain solutions but the number of non-dominated solutions is far smaller than with ACO. It can also be observed that ACO has some outliers in terms of computation time but normally stays around 23 seconds. UUsing the Wilcoxon rank-sum test, a p-value of $2.6x10^{-8}$ was obtained for the number of solutions and $1.86x10^{-9}$ for running time. These extremely low p-values indicate that the results are statistically significant.

Figure 2 better illustrates the difference between the quality of solutions. It can be observed that the solutions from the ACO minimize all objectives, while random search even though it gets solutions, they are really bad and solutions are on the complete opposite side of the pareto front. The rest of the plots, showing very similar behavior, can be seen in Appendix A.

5 Conclusions and Future Work

In this project, we introduced an Ant Colony Optimization (ACO) approach to solve the Multiobjective Shortest Path Problem (MSPP), demonstrating its ef-

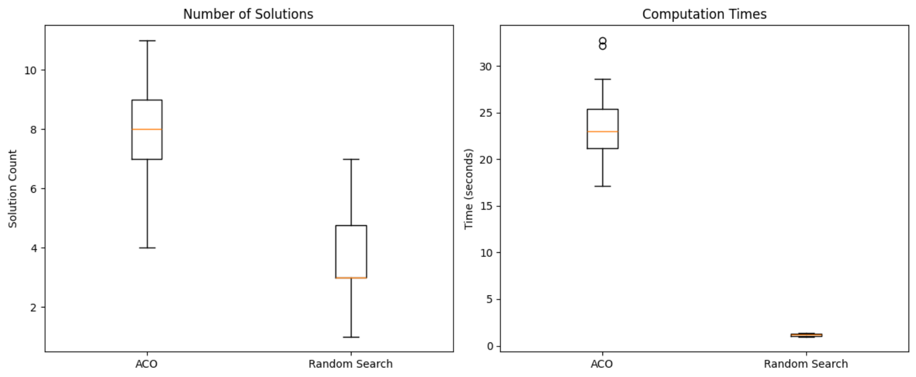

Fig. 1. Comparison between ACO and Random Search

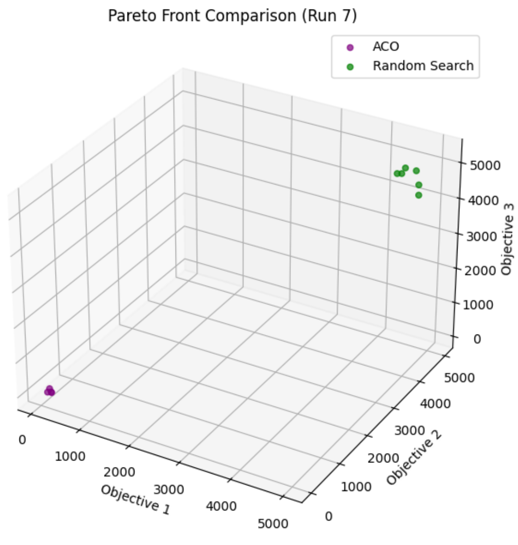

Fig. 2. Pareto Front Plot with run 7

fectiveness in generating high-quality Pareto-optimal solutions. Our experimental results highlight the strengths of ACO in comparison to a baseline random search algorithm. ACO consistently produced a larger number of high-quality, non-dominated solutions compared to random search. The Pareto front generated by ACO minimized all objectives effectively, while the solutions from random search were of significantly lower quality, positioned far from the Pareto front. Even though ACO takes a higher computational time, the trade-off is worth it because of the increase in quantity and quality of solutions.

Future research in this area with more computational resources available, could test this method on more robust scenarios with more nodes in the graph and number of objectives to minimize. This work could also be applied to a real-world scenario to test for viability in these scenarios. Finally, an adaptive strategy for adjusting the model's parameters during runtime could prove really useful.

References

1. Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.
2. Carlos García-Martínez, Oscar Cordon, and Francisco Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180:116–148, 07 2007.
3. Janusz Granat and Francesca Guerriero. The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1):103–118, 2003.
4. Luis Paquete and Thomas Stützle. A two-phase local search for the biobjective traveling salesman problem. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 479–493. Springer, 2003.
5. Xavier Gandibleux, Frédéric Beugnies, and Sabine Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR*, 4(1):47–59, 2006.
6. Andrea Raith and Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299–1331, 2009.
7. Fangguo He, Huan Qi, and Qiong Fan. An evolutionary algorithm for the multi-objective shortest path problem. In *International Conference on Intelligent Systems and Knowledge Engineering 2007*, pages 1276–1280. Atlantis Press, 2007.
8. Plerre-P Grassé. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6:41–80, 1959.
9. Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*, 1992.
10. Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

11. Sascha Häckel, Marco Fischer, David Zechel, and Tobias Teich. A multi-objective ant colony approach for pareto-optimization using dynamic programming. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 33–40, 2008.

A 3D Plots

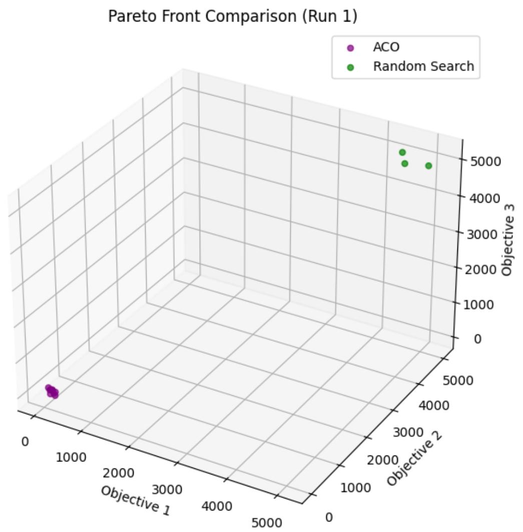

Fig. 3. Run 1

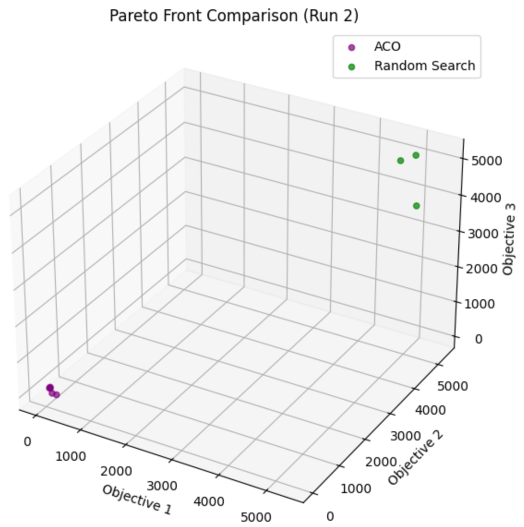

Fig. 4. Run 2

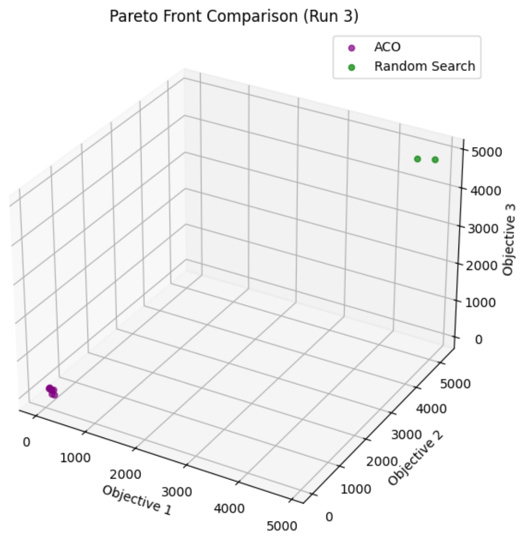

Fig. 5. Run 3

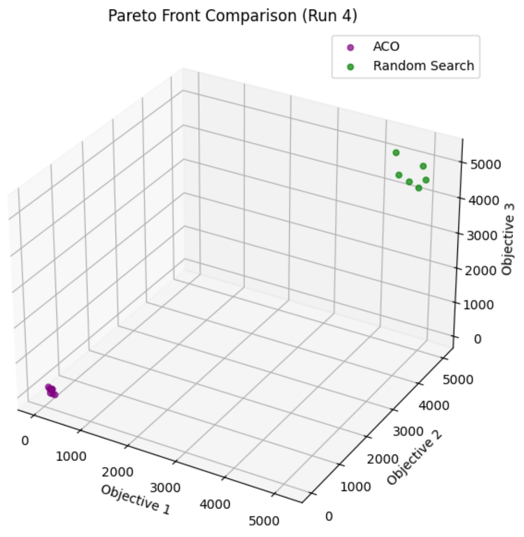

Fig. 6. Run 4

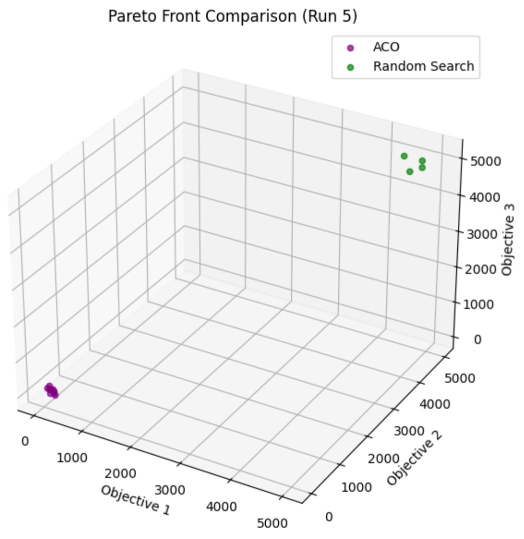

Fig. 7. Run 5

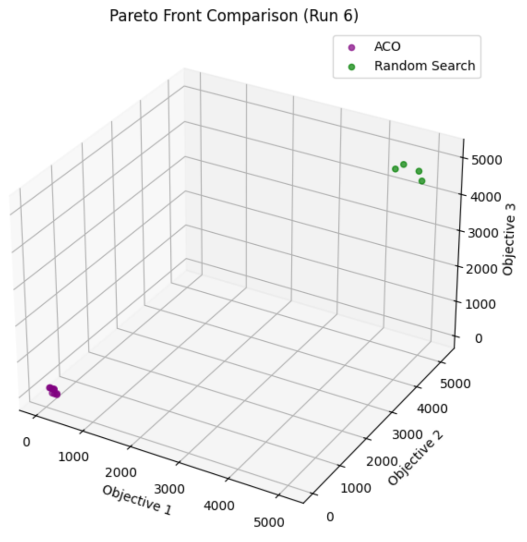

Fig. 8. Run 6

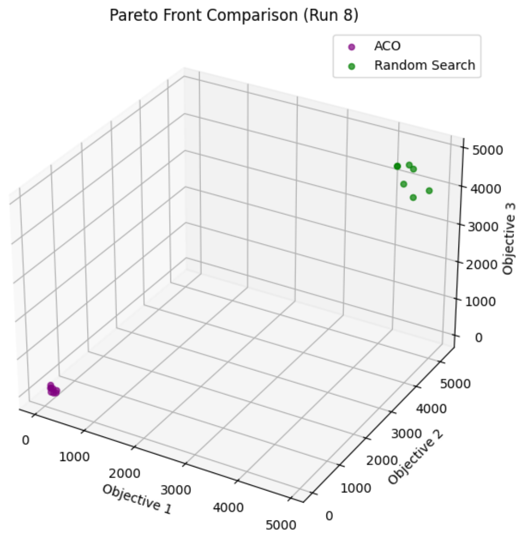

Fig. 9. Run 8

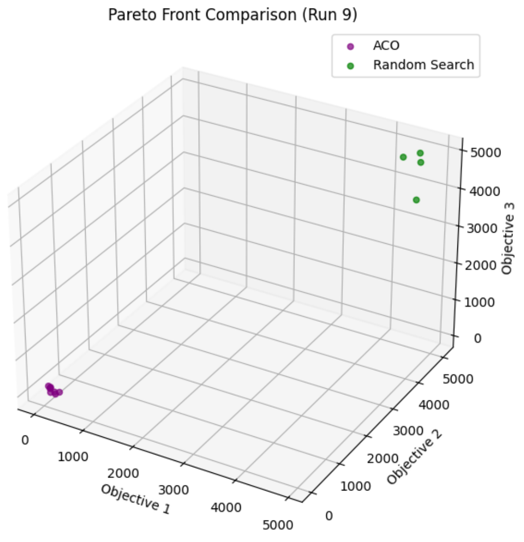

Fig. 10. Run 9

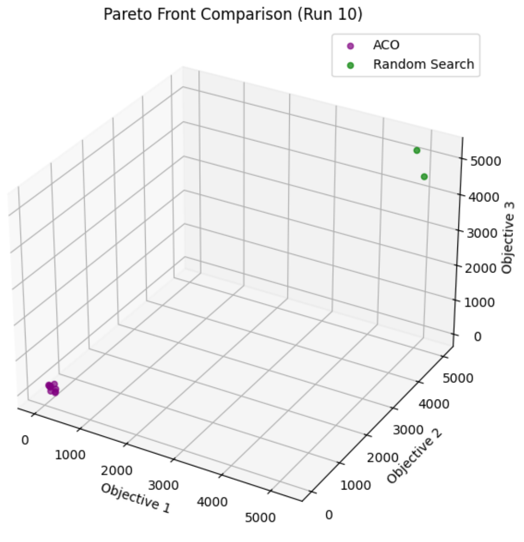

Fig. 11. Run 10

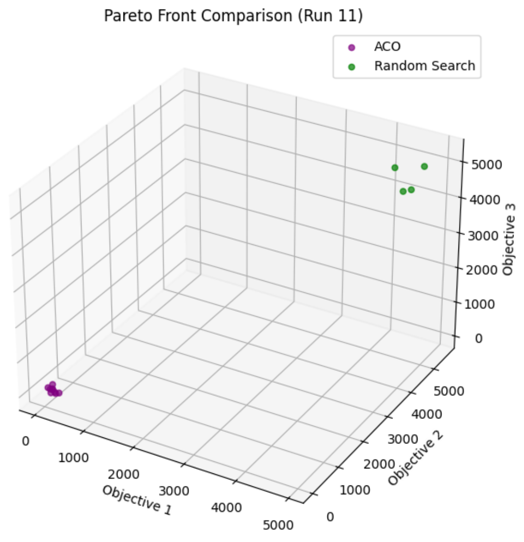

Fig. 12. Run 11

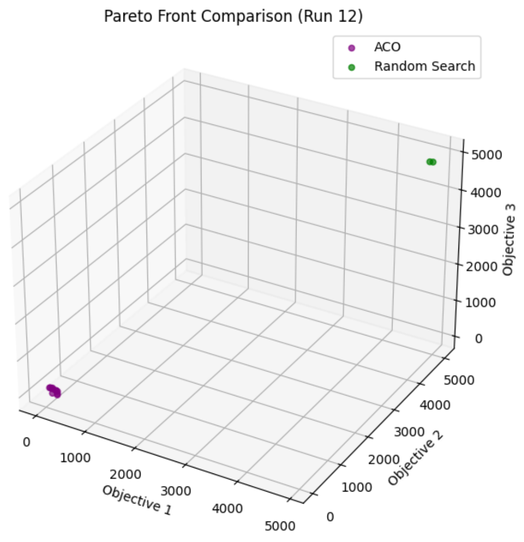

Fig. 13. Run 12

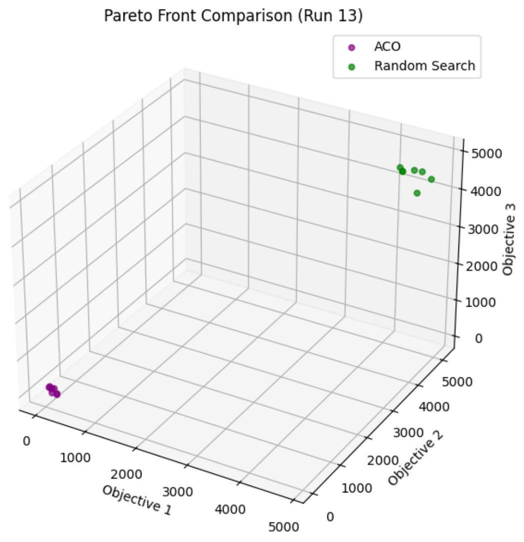

Fig. 14. Run 13

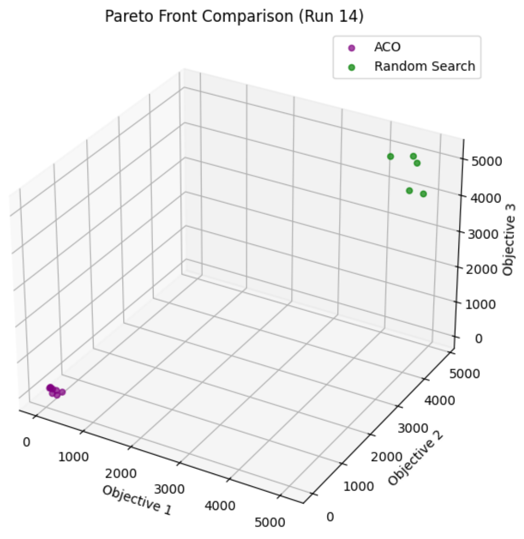

Fig. 15. Run 14

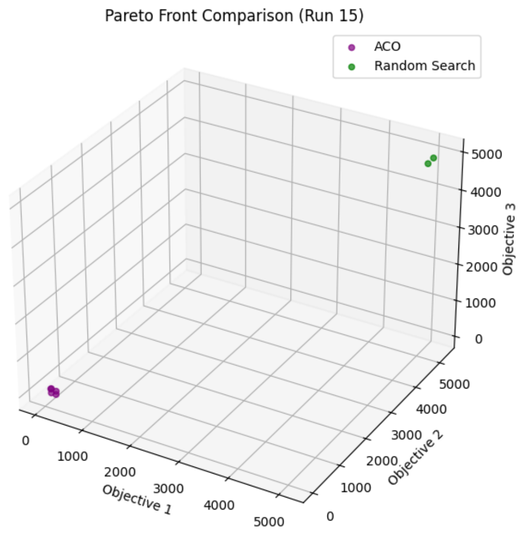

Fig. 16. Run 15

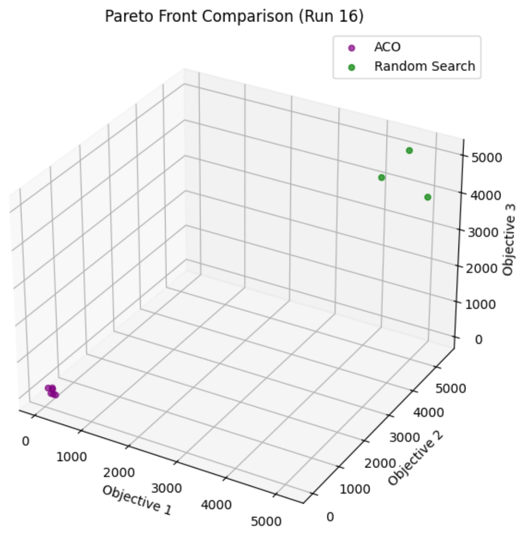

Fig. 17. Run 16

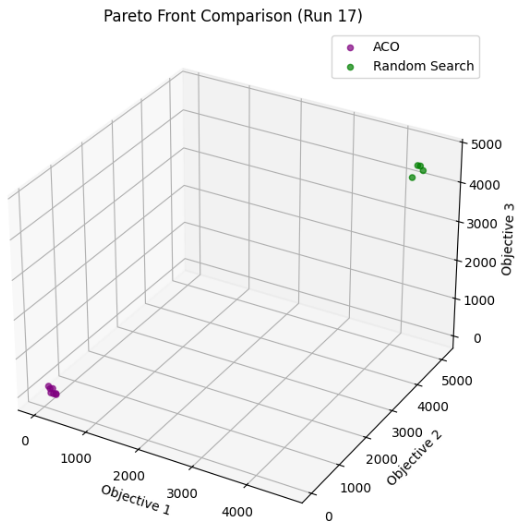

Fig. 18. Run 17

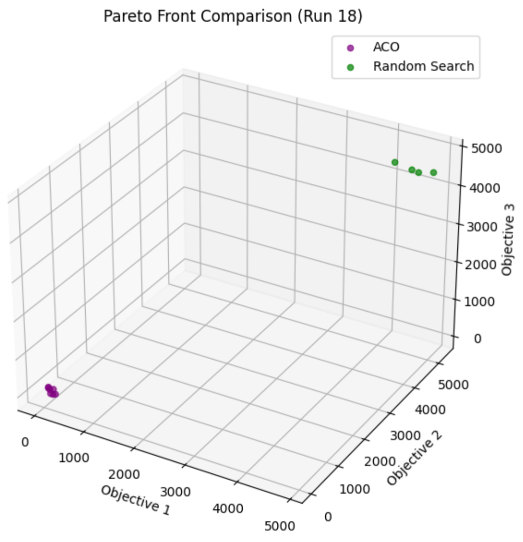

Fig. 19. Run 18

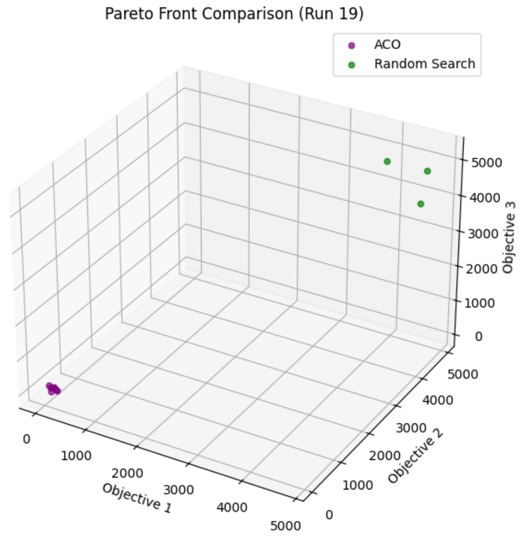

Fig. 20. Run 19

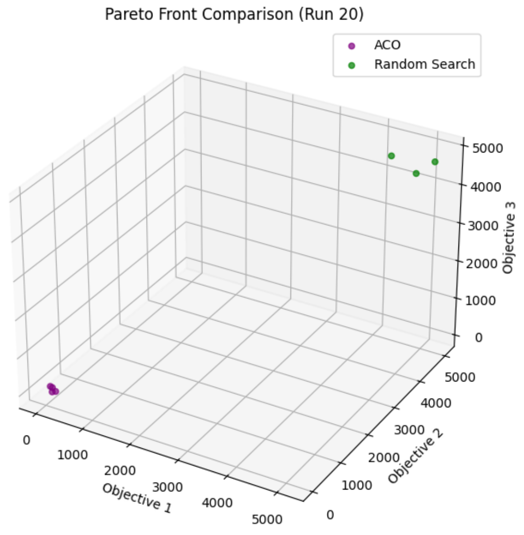

Fig. 21. Run 20

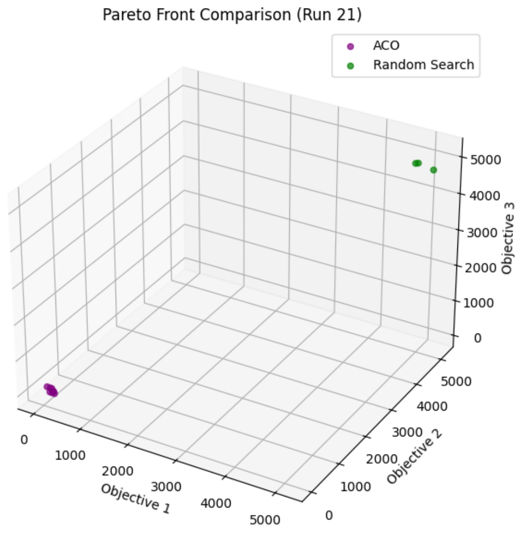

Fig. 22. Run 21

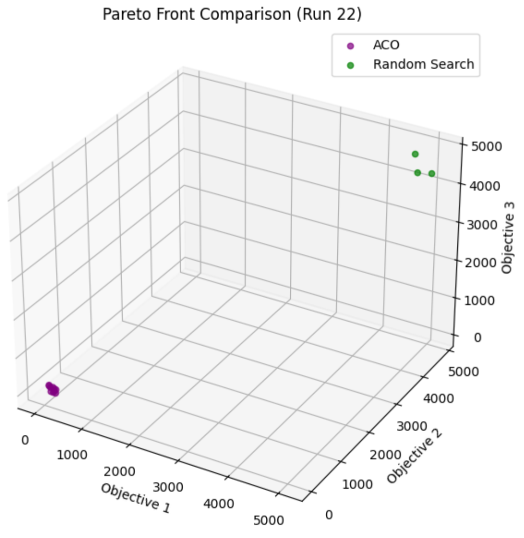

Fig. 23. Run 22

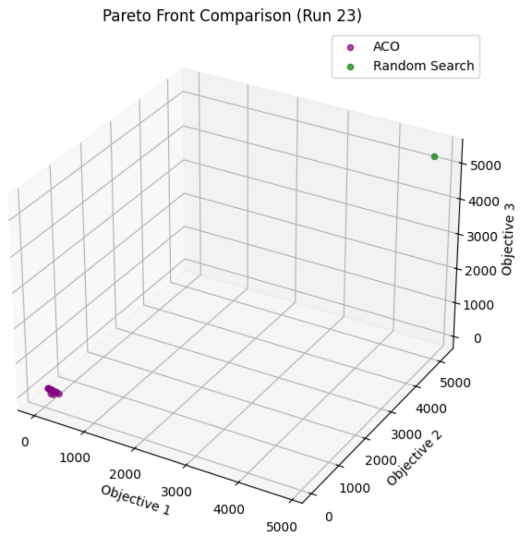

Fig. 24. Run 23

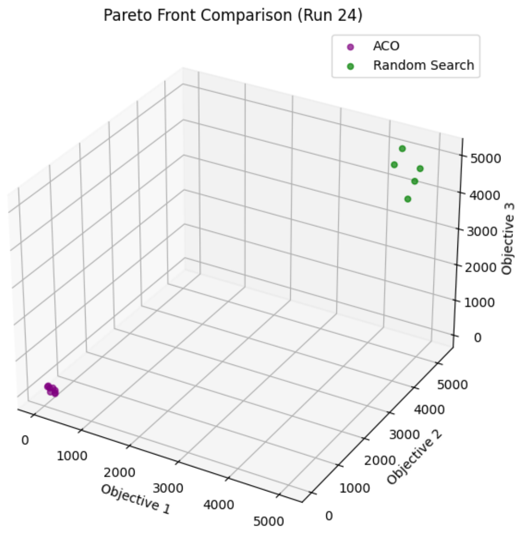

Fig. 25. Run 24

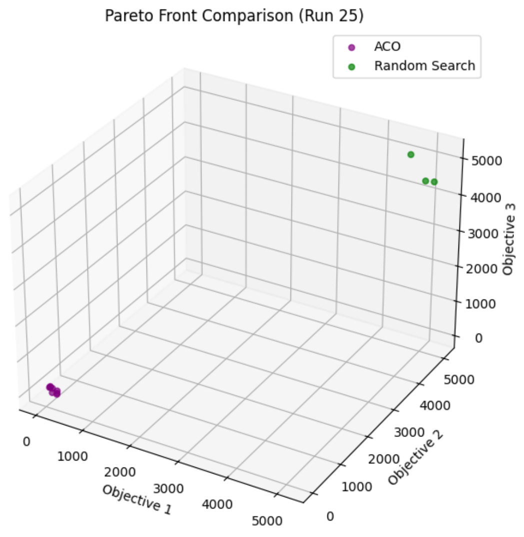

Fig. 26. Run 25

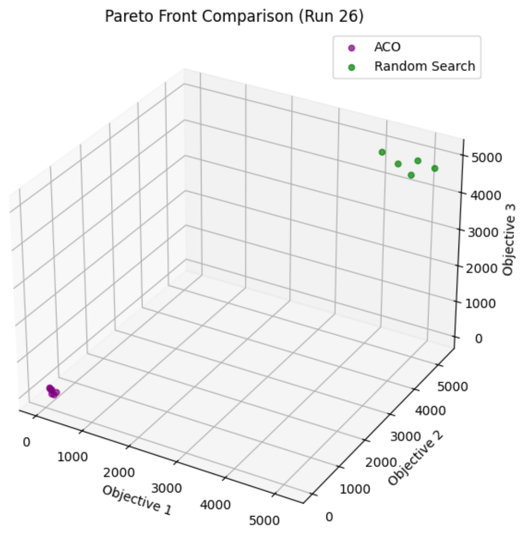

Fig. 27. Run 26

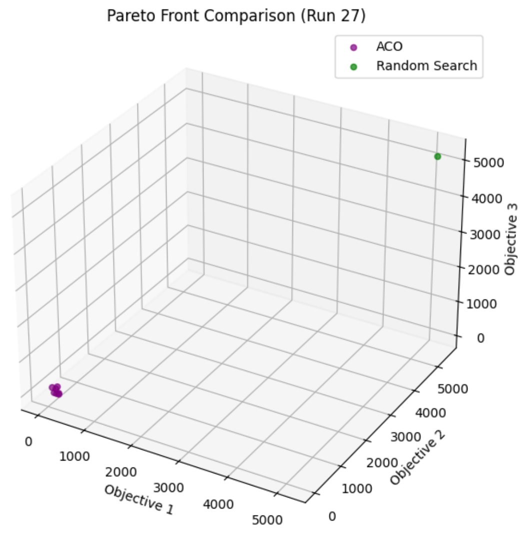

Fig. 28. Run 27

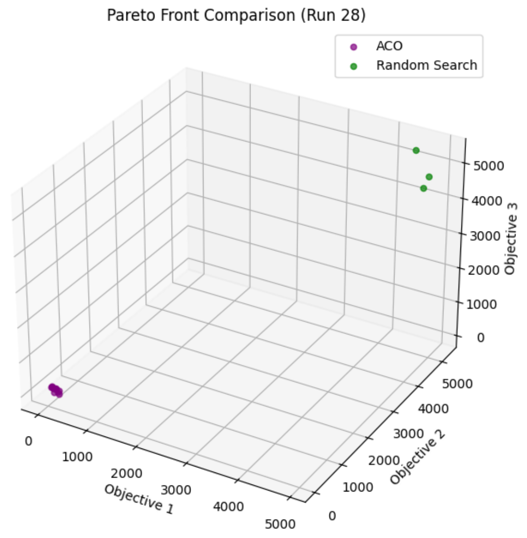

Fig. 29. Run 28

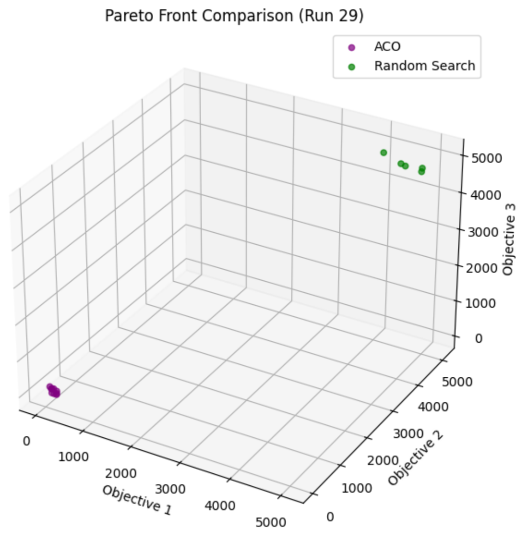

Fig. 30. Run 29

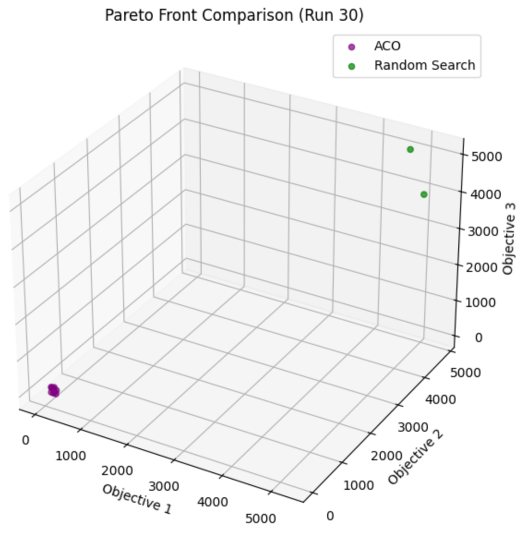

Fig. 31. Run 30