TÉCNICO LISBOA

Optimization and Decision

Msc. in Mechanical Engineering

Capacitated Vehicle Routing Problem

Group 14

GITHUB

Authors:

Madalena Sampaio - 93294
Maria Cardoso - 93661
João Rosado - 96409

2023/2024 – 2nd Semester, P1

Contents

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a classical optimization problem in logistics and operations research. It involves determining the most efficient routes for a fleet of vehicles to deliver goods to a set of destinations, while considering capacity constraints for each vehicle. Each route visits a subset of nodes and both starts and terminates at the depot. The primary objective is to minimize the total distance or cost of all routes, ensuring that the demands of all destinations are met without exceeding the truck capacities, without specifying the number of trucks to be used.

The main goal of this project is to explore and analyze the effectiveness of different heuristic and metaheuristic methods in solving the CVRP, with a particular focus on two heuristics and two metaheuristics approaches: the Sweep Algorithm, Nearest Neighbor, Artificial Bee Colony, and Ant Colony Optimization. Achieving these goals involves several steps:

- **Understand and Model CVRP:** Define the CVRP by incorporating its constraints and objectives into a solvable model, preparing it for algorithmic solutions.
- **Implement Heuristic Methods:** Select and implement heuristic methods that are expected to provide good quality solutions within reasonable computational times, focusing on strategies that are simpler and more intuitive.
- **Implement Metaheuristic Methods:** Select and implement more sophisticated algorithms capable of escaping local optima and potentially providing solutions closer to the optimum.
- **Compare and Analyze Results:** Conduct a comparison of the outcomes generated by the heuristic and metaheuristic methods. This analysis will focus on solution quality, computational efficiency, and the methods' scalability. Based on the comparative analysis, insights into the strengths and weaknesses of each method will be provided.

Individual Contributions

Regarding the implementation phase of this project, Maria Cardoso implemented the Sweep Algorithm, Madalena Sampaio focused on the Nearest Neighbor Algorithm and the Artificial Bee Colony, and João Rosado was responsible for the Ant Colony Optimization. Beyond these individual roles, all team members worked together to analyze the results, discuss the findings, and write the report. This collective effort was key in solving and analyzing the CVRP at hand.

2 Problem Formulation

Let x_{kij} be a binary decision variable defined to indicate whether the vehicle k, $k \in 1, 2, ..., p$, traverses an arc (i, j) in an optimal solution (1 if it is, 0 if it is not), c_{ij} representing the travelling distance from node i to node j and $q_j \leq Q$ depicting the demand at destination node j. Each vehicle has a maximum capacity Q in terms of the total demand it can serve. The minimum number of vehicles needed to serve all customers is given by Equation 1.

$$\left\lceil \frac{\sum_{i=1}^{n} q_i}{Q} \right\rceil \tag{1}$$

Although this number does not necessarily represent the optimal number of vehicles required to satisfy the goal of this study.

2.1 Objective Function

The objective function is to minimize the total distance of each route while covering the demands of every node/costumer, mathematically represented in Equation 2.

$$\sum_{k=1}^{p} \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} c_{ij}\, x_{kij} \tag{2}$$

2.2 Problem Constraints

- **Capacity Constraint**

  The sum of the demands of the visited nodes in a route is less than or equal to the capacity of the vehicle performing the service:

$$\sum_{i=1}^{n} \sum_{j=2, i \neq j}^{n} q_j\, x_{kij} \leq Q \quad , \quad \forall k \in \{1, \ldots, p\} \tag{3}$$

- **Routing Constraint**

  Each node is visited by exactly one vehicle:

$$\sum_{k=1}^{p} \sum_{i=1, i \neq j}^{n} x_{kij} = 1 \quad , \quad \forall j \in \{1, \ldots, n\} \tag{4}$$

- **Depot Constraints**

  1. Each vehicle must start from the depot node:

$$\sum_{j=2}^{n} x_{k1j} = 1 \quad , \quad \forall k \in \{1, \ldots, p\} \tag{5}$$

  2. Each vehicle must terminate at the depot node:

$$\sum_{i=2}^{n} x_{ki1} = 1 \quad , \quad \forall k \in \{1, \ldots, p\} \tag{6}$$

- **Flow Constraint**

  The number of the vehicles arriving at every customer and entering the depot is equal to the number of the vehicles leaving:

$$\sum_{i=1, i\neq j}^{n} x_{kij} = \sum_{i=1}^{n} x_{kji} \quad , \quad \forall j \in \{1, \ldots, n\},\ k \in \{1, \ldots, p\} \tag{7}$$

- **Variable Domain**

  Specify the definition domains of the variable x:

$$x_{kij} \in \{0, 1\} \quad , \quad \forall k \in \{1, \ldots, p\},\ i, j \in \{1, \ldots, n\} \tag{8}$$

3 Implementation

3.1 Benchmarks

Prior to the implementation of algorithms for solving the CVRP, it was important to select appropriate benchmarks for evaluation. In the second part of the project, the benchmarks used were diferent from those in the initial part. This choice was motivated by the need to know the optimal solutions, which aids in the critical assessment of the algorithms' performance.

For this purpose, the CVRPLib library [1] was used. The benchmarks chosen for this project include:

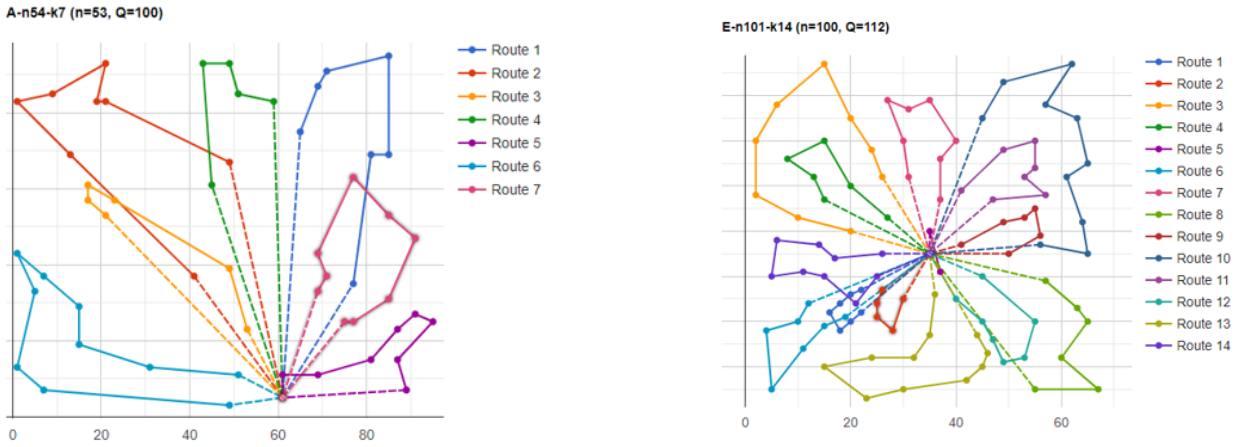
- **A-n54-k7:** This dataset represents a problem instance with 54 nodes, including the depot. The 'A' series in the CVRPLib is known for its moderate problem size and complexity, making it an ideal starting point for testing our algorithms. This benchmark is particularly useful for evaluating the efficiency of our initial solution construction and optimization processes.
- **E-n101-k14:** This benchmark consists of 101 nodes, including the depot. The 'E' series datasets are characterized by their larger size and increased complexity, presenting a more challenging scenario for our algorithms. The selection of E-n101-k14 allows us to test the scalability and robustness of our solution methods under more demanding conditions.

Each dataset provides detailed information including the coordinates and demands of each node, the capacity of each truck, and the minimum number of trucks required to meet the demands (though this figure is not utilized in solving the problem with our algorithms). Additionally, the optimal solution is included for assessment and comparison. In both benchmarks, the first node listed in each dataset represents the depot, with its demand set to zero.

Table 1 summarizes the most important specifications of each benchmark and Figure 1 portrays the respective optimal routes.

Table 1: Benchmark specifications for CVRP instances

Benchmark	No. of Location Nodes	Capacity	Min. No. of trucks	Optimal Values
A-n54-k7	53	100	7	1167
E-n101-k14	100	112	14	1071



(a) **(b)**

Figure 1: Benchmark optimal routes. **(a)** A-n54-k7. **(b)** E-n101-k14.

All methods' implementations begin by defining the necessary functions to import the datasets and defining the variables required for solving the problem. Once these are set, the distance matrix (equivalent to the cost matrix) is computed. This matrix contains the Euclidean distance between each pair of nodes and is crucial for determining the solution routes and evaluating the total cost of the obtained solution, regardless of the method or algorithm used.

3.2 Heuristics

3.2.1 Sweep Algorithm

The Sweep algorithm is a constructive heuristic frequently used to solve the CVRP. This algorithm operates by rotating a conceptual line around the depot, either clockwise or counterclockwise, beginning at a 0-degree angle and progressing through the nodes. The first node encountered is allocated to the first vehicle, subject to capacity constraints. A node is marked as visited if it falls within the vehicle's capacity limits. Subsequently, the line sweeps again to identify the next node, continuing in this manner until all nodes have been visited, resulting in an initial solution for the CVRP.

3.2.2 Nearest Neighbor

The Nearest Neighbor algorithm is a straightforward heuristic used for solving optimization problems, such as the CVRP. The algorithm initializes by marking all nodes as unvisited. Then,

starting from the depot (node 0), it iteratively selects the nearest unvisited node that the vehicle can visit without exceeding its capacity. When a vehicle can no longer visit an unvisited node without exceeding capacity, it returns to the depot, and a new vehicle starts a new route using the same procedure. This process repeats until all nodes have been visited and assigned to a route, resulting in an initial solution for the CVRP.

3.2.3 Two-Opt Improvement

The Two-Opt algorithm is used as a local search heuristic to optimize routes derived from initial solutions. In this context, the initial solutions are the ones previously obtained using the Sweep Algorithm and Nearest Neighbors. By iteratively selecting and removing two edges from a given route and then reconnecting these edges in a new configuration, the algorithm aims to minimize the total cost associated with the route. If the reconfiguration results in a lower total cost compared to the original route, the new, improved route is retained. This iterative process of adjustment continues until no further reductions in total cost can be achieved. A total of 100 000 iterations were used to achieved the final improved results.

3.3 Metaheuristics

3.3.1 Artificial Bee Colony

The Artificial Bee Colony (ABC) algorithm is a swarm-based metaheuristic optimization algorithm inspired by the foraging behavior of honey bees. Its implementation in this project was based on pre-existing code discovered online [3], which was then adapted to meet our specific needs. This algorithm operates with an initial population of solutions (food sources), which are generated randomly. This algorithm comprises three groups of bees with different tasks:

- **Employed Bees**: Each employed bee is tasked with exploring the vicinity of its food source (solution) to find a new, potentially better, food source. The number of employed bees is equal to the number of food sources. In the implementation, it corresponds to the variable *n_initials*.
- **Onlooker Bees**: Onlooker bees select among the food sources based on a probability proportional to the food sources' fitness. In this case, the fitness is equal to the solution cost. They then try to improve these food sources further. The number of onlooker bees can vary but is often set equal to the number of employed bees for simplicity and balance between exploration and exploitation. In the implementation, it corresponds to the variable *n_onlookers*.
- **Scout Bees**: Scout bees are employed bees that abandon their food sources if no improvement is made after a certain number of trials (defined by a limit). They then scout for new food sources randomly. In the implementation, it corresponds to the variable *search_limit*.

The parameters were chosen through a long process of trial and error. The number of epochs was defined by evaluating the fitness value evolution. When there is a plateau at the end of

the plot, more iterations are not required as they will not significantly improve the results. The search limit was set to high values due to the complexity of the benchmarks used (high number of nodes) and the number of employer and onlooker bees was set to a higher value for the benchmark with the most nodes. The values used are summarized in Table 2.

Table 2: ABC Implementation parameters

	A-n54-k7	E-n101-k14
n_epochs	500	800
n_initials	30	50
n_onlookers	30	50
search_limit	100	150

3.3.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a bio-inspired algorithm rooted in the foraging behavior of ants. Mimicking the ants' pheromone-based communication, ACO iteratively constructs solutions to optimization problems. Initially, artificial ants explore solution spaces by probabilistically selecting components while being guided by both heuristic information and pheromone trails. The solutions are evaluated, and pheromone trails are updated based on their quality, with better solutions receiving stronger pheromone deposits. Through iteration, the algorithm converges towards optimal solutions, with pheromone trails guiding ants to explore promising regions of the solution space. Optional local search methods may refine solutions, enhancing convergence speed and quality.

At each iteration, each edge is given a probabilistic value, p_{ij}, given by Equation 9, in which τ_{ij} and μ_{ij} represent the pheromone value and local heuristic of that edge, respectively. The parameters α and β account for the relative importance of pheromone versus heuristic. The value of τ_{ij} for the initial iteration was set to 1.

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \times \mu_{ij}^{\beta}}{\sum_{j \in N} \tau_{ij}^{\alpha} \times \mu_{ij}^{\beta}} \tag{9}$$

While μ_{ij} takes the form of $1/c_{ij}$, in other words, the inverse of the cost (or euclidean distance) of the edge, τ_{ij} is an iterative function (Equation 10), meaning that on each iteration it takes a different value. The evaporation coefficient ρ regulates the rate at which pheromone trails diminish over time.

$$\tau_{n+1} = \tau_n \cdot (1 - \rho) + \Delta\tau_{ij} \tag{10}$$

Furthermore, $\Delta\tau_{ij}$ is obtained as:

$$\Delta\tau_{ij} = \begin{cases} Q/f & \text{, if } (i,j) \in S \\ 0 & \text{, otherwise} \end{cases} \tag{11}$$

where f stands for the average optimal distance covered by the artificial ants at each iteration.

The parameters and coefficients used for the implementation of the ACO algorithm are shown in Table 3. These were set after a tedious trial and error process which finalized upon the lowest cost reached.

Table 3: ACO Implementation configuration

	A-n54-k7	E-n101-k14
α	2	2
β	4	4
Q	5	5
ρ	0.1	0.1
No. of Ants	50	100
No. of Iterations	500	1000

4 Results

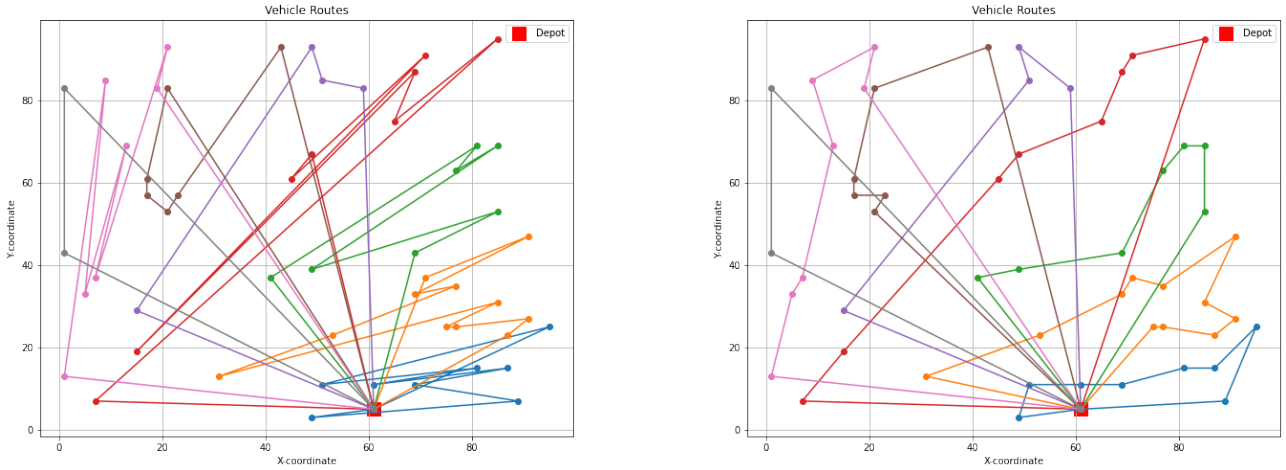
4.1 Heuristics

4.1.1 Sweep Algorithm

The results obtained from the application of the Sweep algorithm to the CVRP are summarized in Table 4 and the solution routes are plotted in Figures 2 and 3.

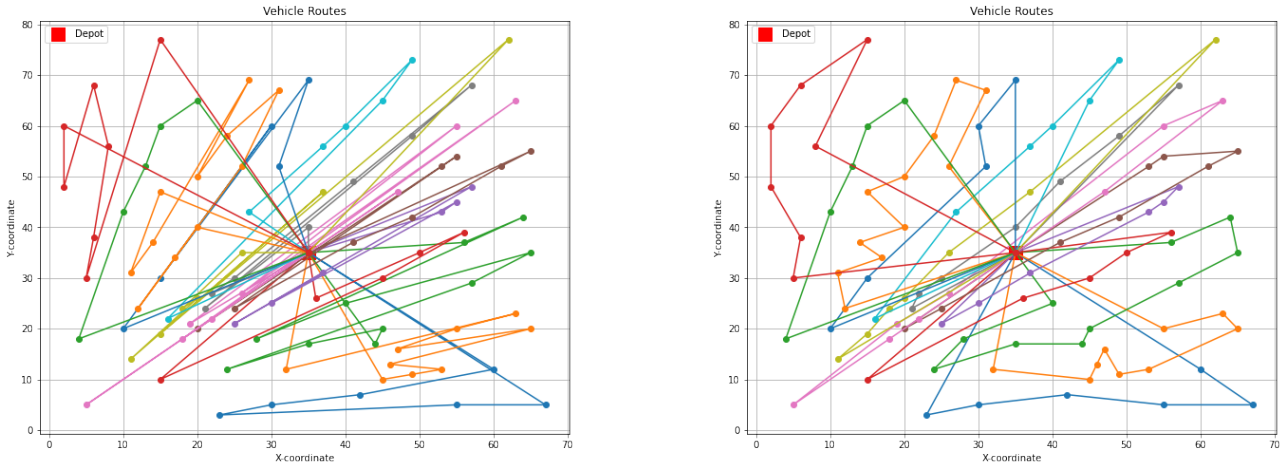
Table 4: Results for Sweep Algorithm

Benchmark	Optimal Cost	Sweep Algorithm Cost	Error	Sweep Algorithm + 2 opt Cost	Error
A-n54-k7	1167	2361.56	102%	1614.85	38%
E-n101-k14	1071	2604.50	143%	1722.82	61%



(a) **(b)**

Figure 2: Sweep algorithm solutions for A-n54-k7. **(a)** without 2-opt optimization. **(b)** with 2-opt optimization.



(a) **(b)**

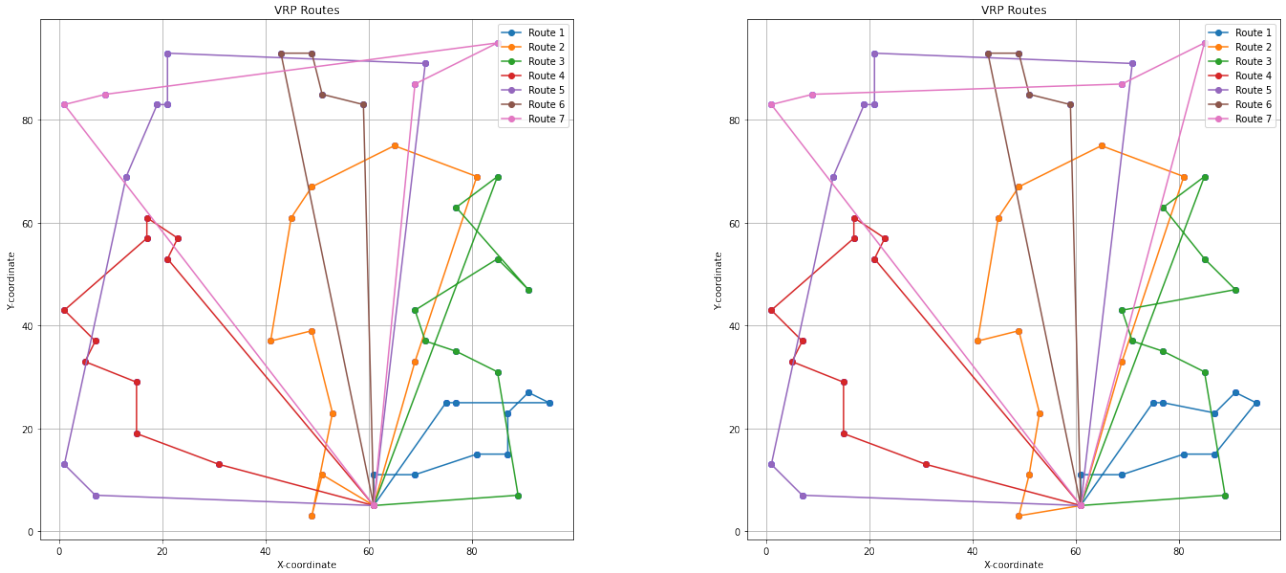
Figure 3: Sweep algorithm solutions for E-n101-k14. **(a)** without 2-opt optimization. **(b)** with 2-opt optimization.

4.1.2 Nearest Neighbor

The results obtained from the application of the Nearest Neighbor algorithm to the CVRP are summarized in Table 5 and the solution routes are plotted in Figures 4 and 5.

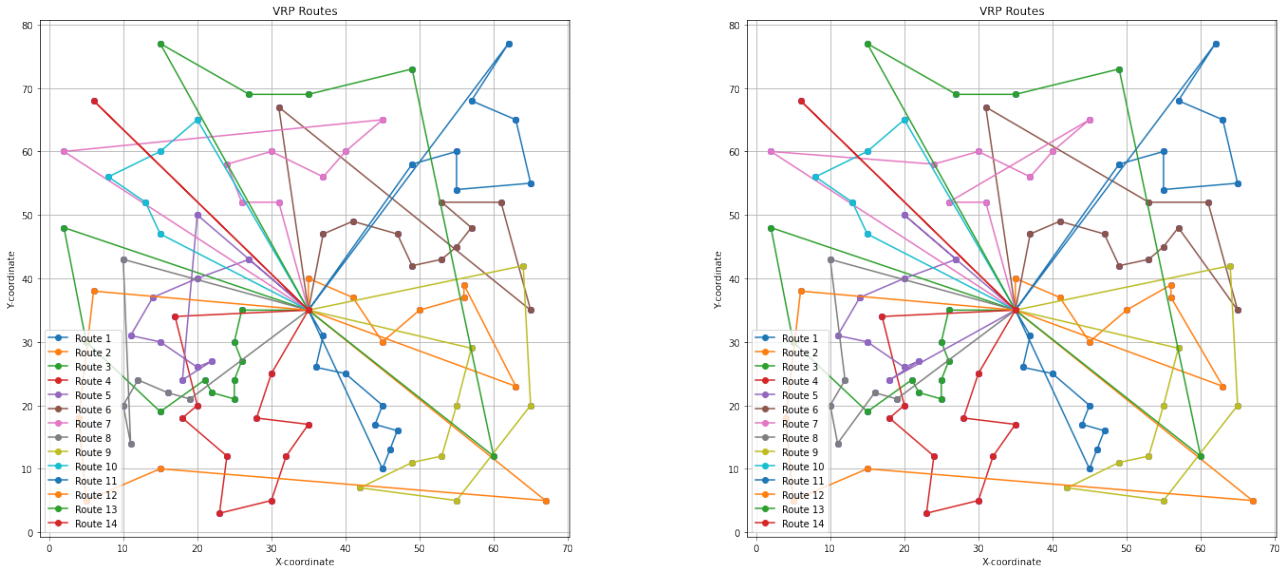
Table 5: Results for Nearest Neighbor Algorithm

Benchmark	Optimal Cost	Nearest Neighbour Cost	Error	Nearest Neighbour + 2 opt Cost	Error
A-n54-k7	1167	1433.64	22.8%	1412.09	21%
E-n101-k14	1071	1610.91	50.4%	1582.83	47.8%



(a) **(b)**

Figure 4: Nearest Neighbours solutions for A-n54-k7. **(a)** without 2-opt optimization. **(b)** with 2-opt optimization.



(a) **(b)**

Figure 5: Nearest Neighbours solutions for E-n101-k14. **(a)** without 2-opt optimization. **(b)** with 2-opt optimization.

4.2 Metaheuristics

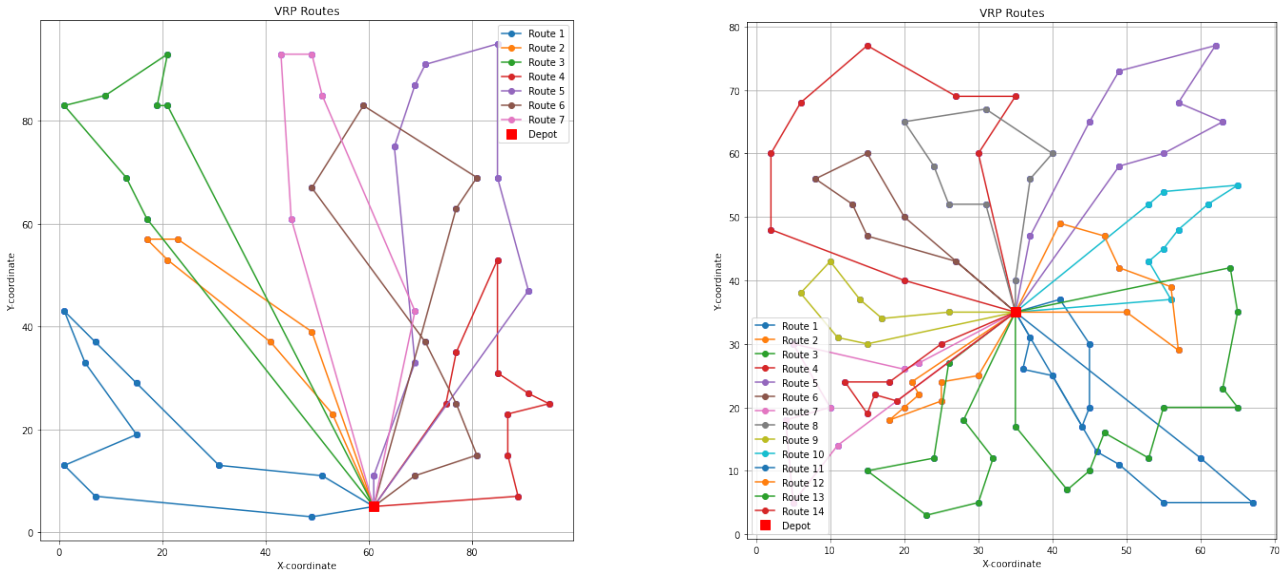
4.2.1 Artificial Bee Colony

The results obtained from the application of the Nearest Neighbor algorithm to the CVRP are summarized in Table 6 and the solution routes are plotted in Figure 6. The evolution of

the ABC cost over the epochs is represented in Figure 7.

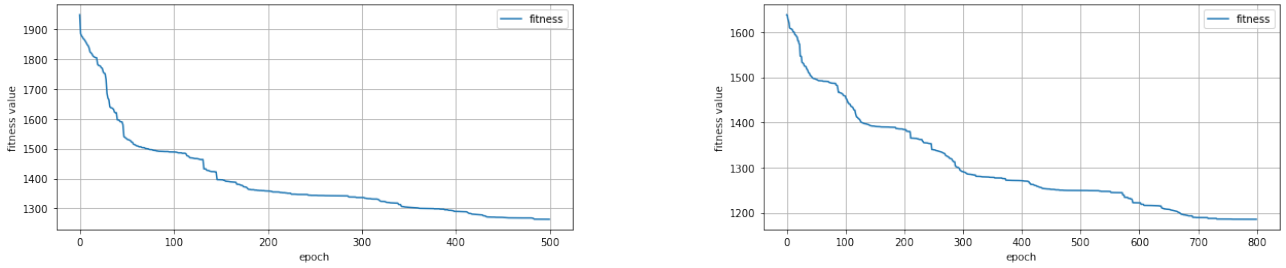
Table 6: Results for ABC Algorithm

Benchmark	Optimal Cost	ACO Cost	Error
A-n54-k7	1167	1274.13	9.17%
E-n101-k14	1071	1191.38	11.2%



(a) **(b)**

Figure 6: Artificial Bee Colony solutions. **(a)** A-n54-k7. **(b)** E-n101-k14.



(a) **(b)**

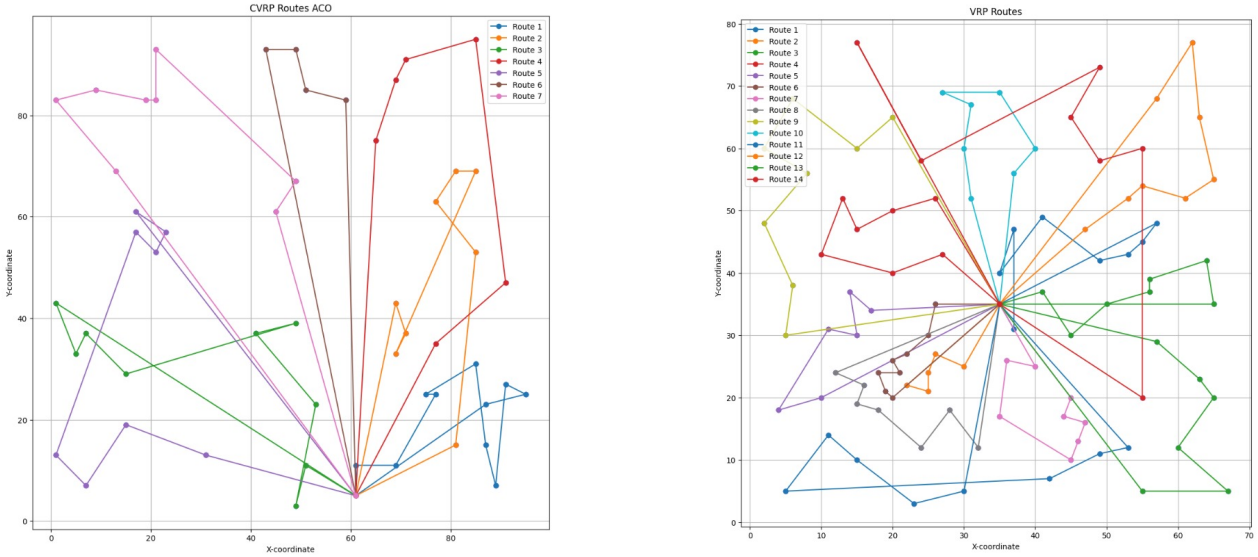
Figure 7: ABC fitness evolution through iterations. **(a)** A-n54-k7. **(b)** E-n101-k14.

4.2.2 Ant Colony Optimization

The results obtained from the application of the Nearest Neighbor algorithm to the CVRP are summarized in Table 7 and the solution routes are plotted in Figure 8. The evolution of the ACO cost over the epochs is represented in Figure 9.

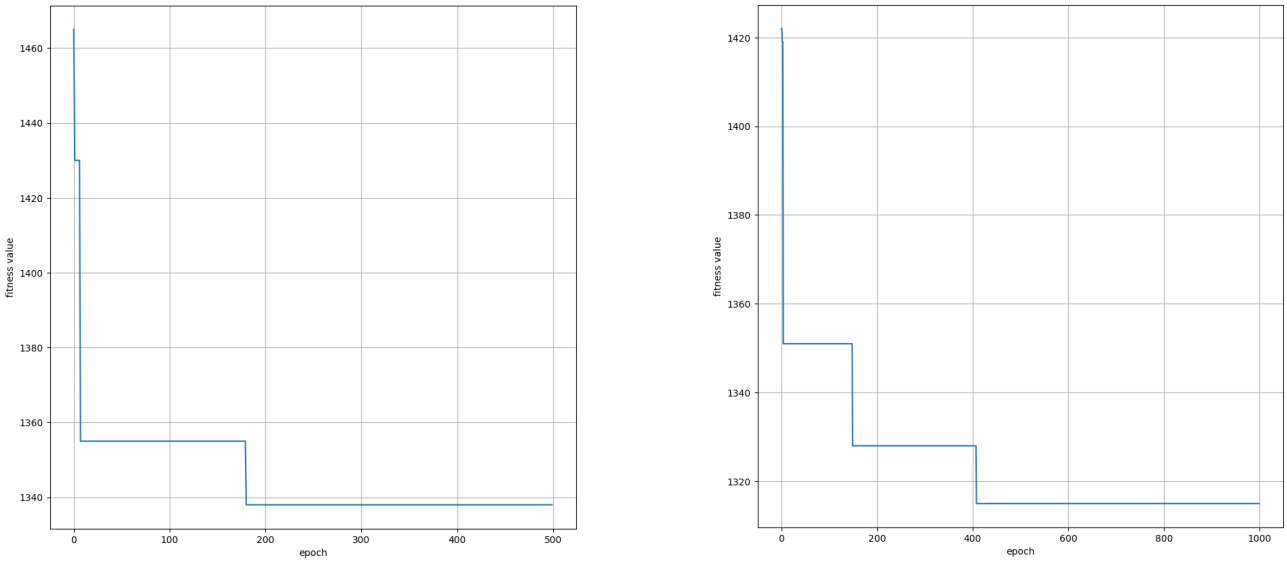
Table 7: Results for ACO Algorithm

Benchmark	Optimal Cost	ACO Cost	Error
A-n54-k7	1167	1338.13	14.7%
E-n101-k14	1071	1315.85	22.9%



(a) **(b)**

Figure 8: Ant Colony Optimization solutions. **(a)** A-n54-k7. **(b)** E-n101-k14.



(a) **(b)**

Figure 9: ACO fitness evolution through iterations. **(a)** A-n54-k7. **(b)** E-n101-k14.

5 Conclusion

The fact that all methods provided solutions with the correct number of routes without specifying the number of trucks suggests that the methods used were effective in organizing routes in an optimized manner without the need to pre-determine the number of vehicles. This highlights the flexibility and adaptability of the approaches used to automatically adjust to the problem's demand.

Regarding the heuristic methods employed, it is evident that the Two-Opt improvement observed with the Sweep Algorithm is significantly greater than that obtained with the Nearest Neighbor method. It is also clear that the routes resulting from the Sweep Algorithm exhibit substantially more crossings than those produced by the Nearest Neighbor method. This observations confirm that the Two-Opt method is indeed suitable for eliminating route crossings, thereby reducing the total distance traveled.

Both heuristic methods employed are greedy, meaning they solve the problem by focusing solely on the next node, without considering the broader context. In contrast, metaheuristics consider the entire solution when solving and making adjustments. For this reason, it could be expected that metaheuristics would yield better results than greedy approaches, a hypothesis that was indeed confirmed.

A common challenge encountered with metaheuristics is the necessity for fine-tuning parameters to achieve optimal results. Computational time constraints can prevent a full exploration of the solution space, suggesting the importance of efficient parameter tuning techniques and code optimization to enhance the feasibility and effectiveness of these methods in practical applications. Indeed, it is noticeable how metaheuristics require significantly more time to solve the problem compared to heuristics, which take only a few seconds.

References

[1] Cvrplib. `http://vrp.galgos.inf.puc-rio.br/index.php/en/`. Accessed on: March 20, 2024.

[2] Github - cvrp˙aco. `https://github.com/pkonowrocki/CVRP_ACO`.

[3] Github - vehicle routing problem. `https://github.com/CodeSopranos/VehicleRoutingProblem/`.

[4] Lectures' slides.

[5] Writing for Ara. Solving vehicle routing problems with python: Heuristics algorithm. `https://medium.com/@writingforara/solving-vehicle-routing-problems-with-python-heuristics-algorithm-2cc57fe7079c`, Mar 2024.

[6] M. A. Mohammed, M. K. Ghani, R. I. Hamed, S. A. Mostafa, D. A. Ibrahim, H. K. Jameel, and A. H. Alallah. Solving vehicle routing problem by using improved k-nearest neighbor algorithm for best solution. *Journal of Computational Science*, 21:232–240, 2017.