# Local search

Luisa María Álvarez García

# Introduction

The exercise focuses on detailing the algorithm for the Parallel Machines Scheduling Problem with Tardiness Minimization.

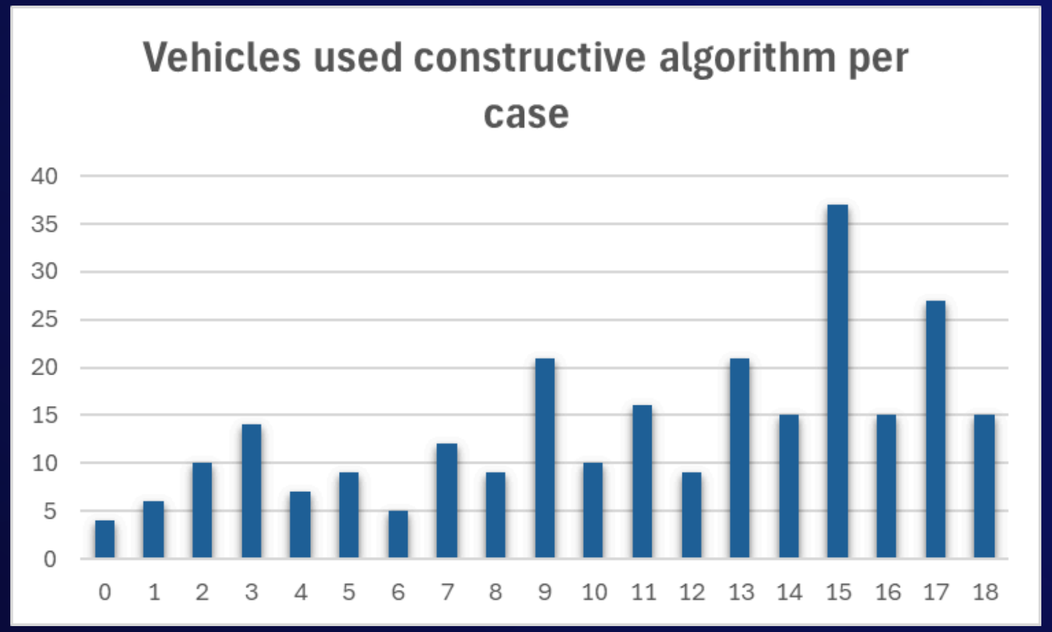
In this problem, we have a set of jobs J={1,2,...,n}J = \{1, 2, \ldots, n\}J={1,2,...,n}, each with a processing time pip_ipi, release time rir_iri, and due date did_idi. These jobs are processed on mmm identical machines K={1,2,...,m}K = \{1, 2, \ldots, m\}K={1,2,...,m}, where each job must be assigned to one machine and cannot overlap with others.

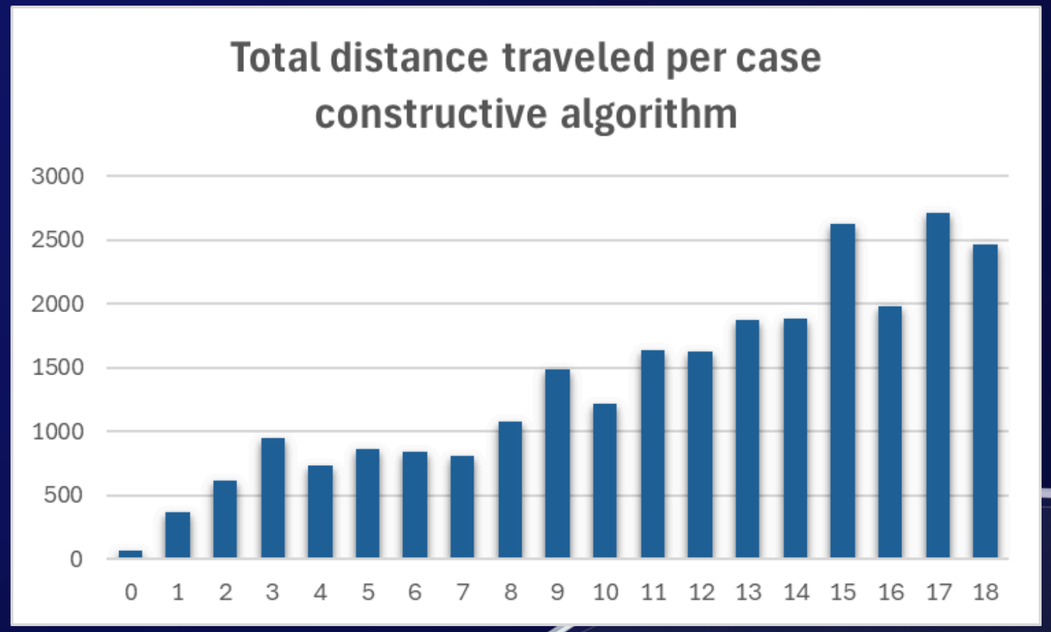
The conditions are: jobs must start after their release times, and tardiness is incurred if a job finishes after its due date. The objective is to minimize the total tardiness across all jobs. The task includes providing a detailed description of the algorithm and corresponding pseudocode.

# Constructive solution

- Each vehicle starts with a defined capacity, and jobs are added based on their feasibility (not exceeding capacity or time windows).
- By choosing the closest feasible job, the algorithm aims to reduce overall travel distance, including the return to the depot.
- If a vehicle can’t take any more jobs, the algorithm moves to the next vehicle, ensuring efficient use of resources.

# CONSTRUTIVE SOLUTION ANALYSIS

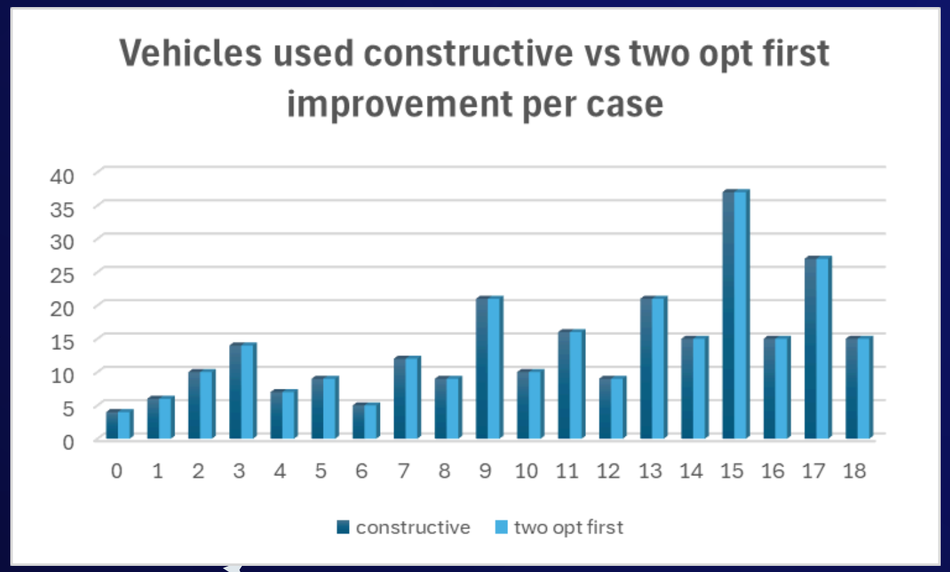


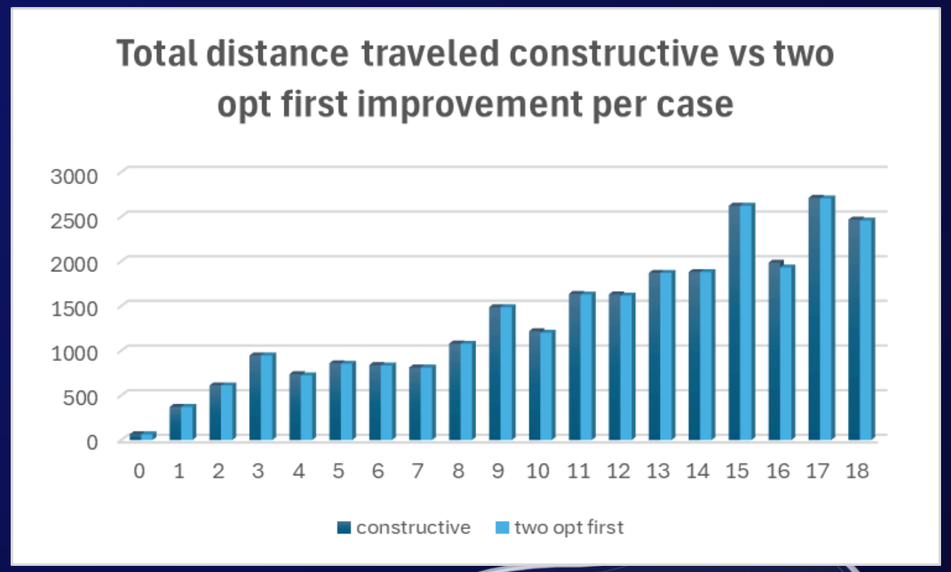



# TWO-OPT FIRST IMPROVEMENT

- The algorithm first assigns jobs to vehicles based on feasibility (capacity and time windows). Each vehicle is initialized with a route starting and ending at a depot, and jobs are added iteratively by selecting the nearest feasible job.
- After constructing initial routes, the algorithm applies a two-opt first improvement method to enhance the routes. This involves checking pairs of points in each vehicle's route and reversing the order of nodes between them if it results in a shorter route, thus reducing total travel distance.
- Throughout the optimization, the algorithm ensures that any new route remains feasible—respecting both the vehicle's capacity and the time windows for each job.
- The total travel distance is calculated for all routes, and the results, including vehicle usage and route details, are saved to an Excel file for analysis.

# Two opt first improvement solution analysis

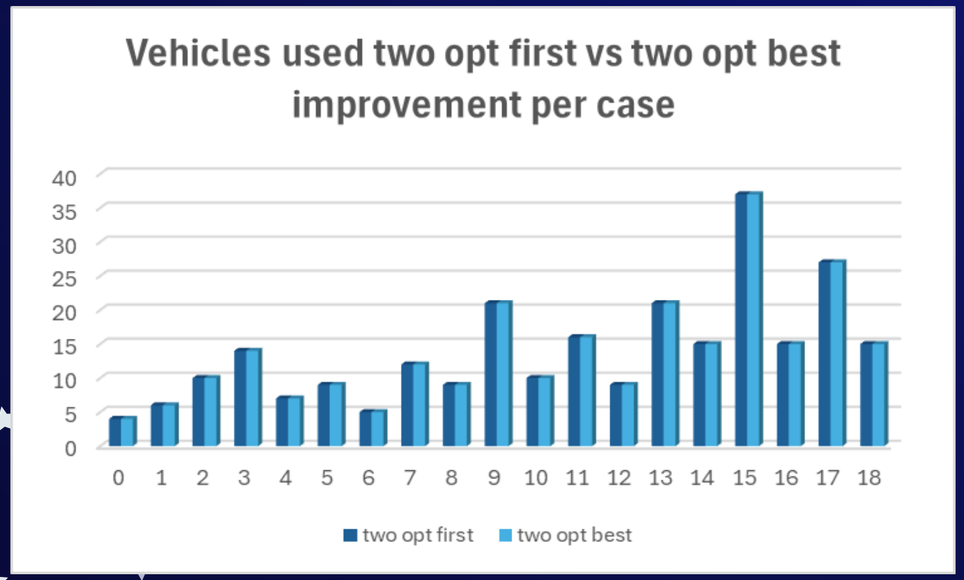


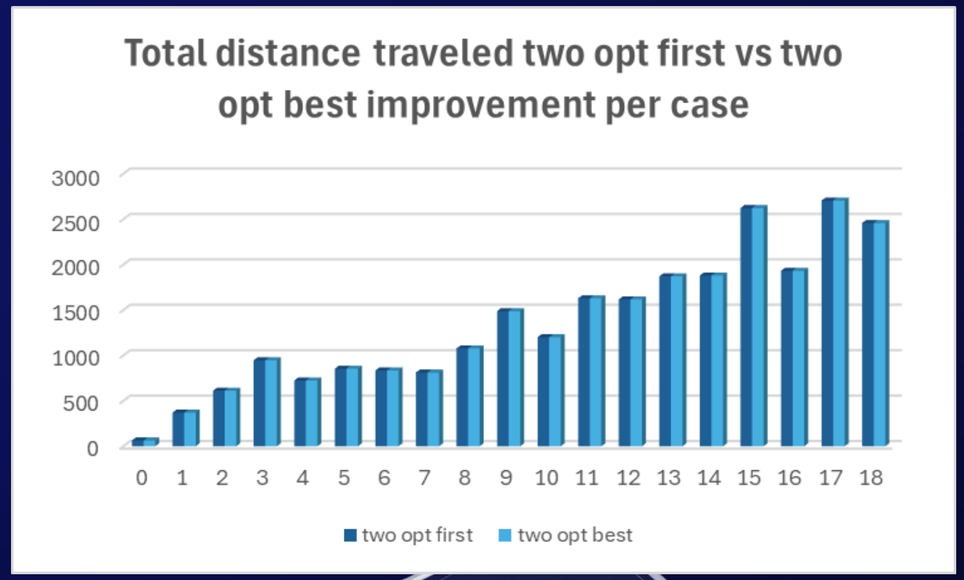



# TWO OPT BEST IMPROVEMENT

- The algorithm starts by creating initial routes for vehicles. It assigns jobs based on feasibility—ensuring that each vehicle does not exceed its capacity and respects the time windows for deliveries. This is done by selecting the nearest feasible node to add to the vehicle's route.
- After constructing the initial routes, the algorithm applies the two-opt best improvement method. It evaluates potential swaps in the route to minimize travel distance. The best improvement is chosen iteratively, making adjustments only if they yield a shorter route. This ensures that the solution progressively converges towards a more optimal configuration.
- Throughout the process, the algorithm maintains strict checks to ensure that any changes to routes remain feasible regarding both the vehicle capacity and the time constraints for each job.
- The total travel distance for all routes is computed, and the results—including the number of vehicles used and details of each route—are saved to an Excel file for further analysis.

# Two opt best improvement solution analysis

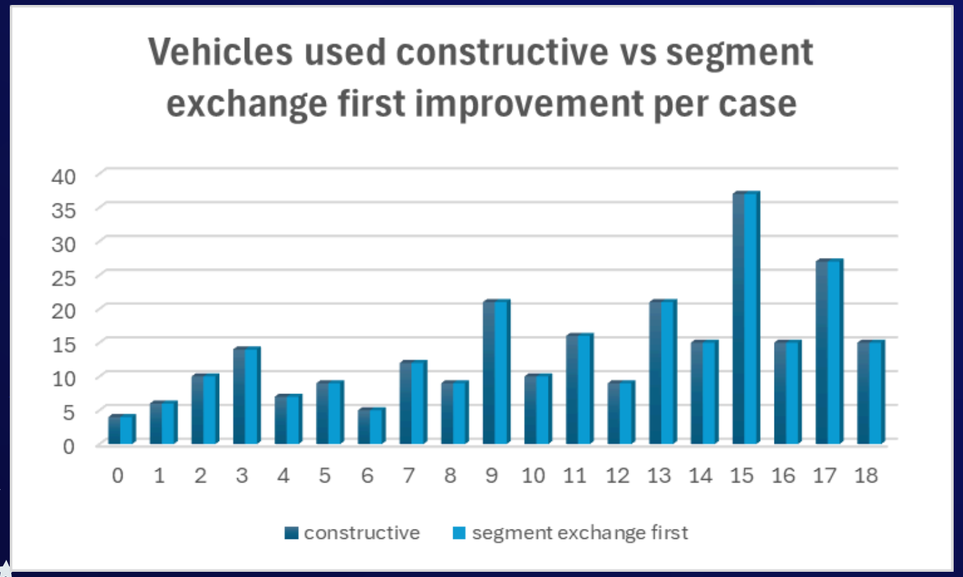


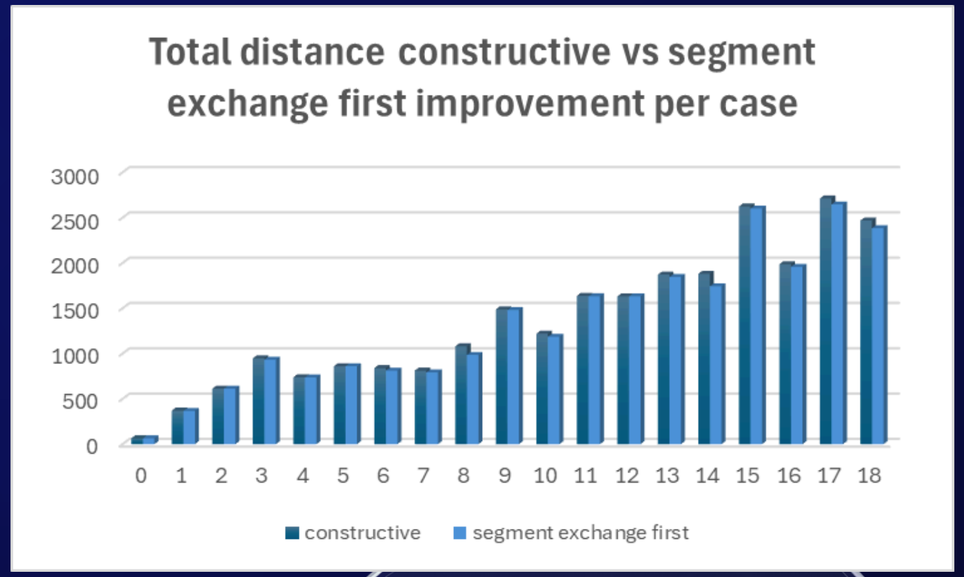



# SEGMENT EXCHANGE FIRST IMPROVEMENT

- The algorithm starts by creating initial routes for vehicles, assigning jobs based on feasibility to ensure capacity limits and time windows are met. It selects the nearest feasible node for each vehicle's route.
- Next, it uses a segment exchange first improvement method to evaluate potential swaps between vehicle routes, minimizing overall travel distance. This iterative process continues until no further improvements are found.
- Finally, the total travel distance is calculated, and results—including the number of vehicles and route details—are saved to an Excel file for analysis. This method effectively optimizes delivery routes while adhering to constraints in the Vehicle Routing Problem with Time Windows (VRPTW).

# SEGMENT EXCHANGE FIRST ANALYSIS

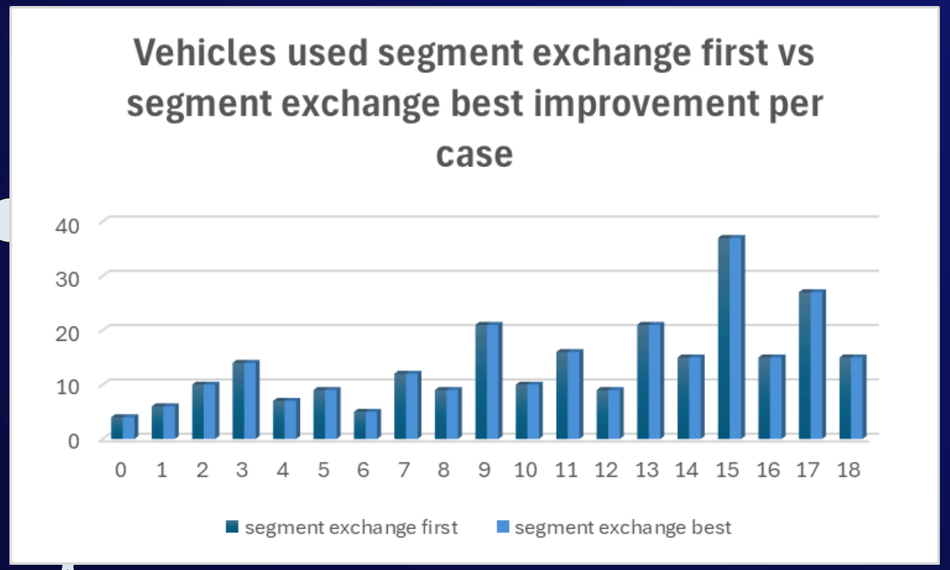


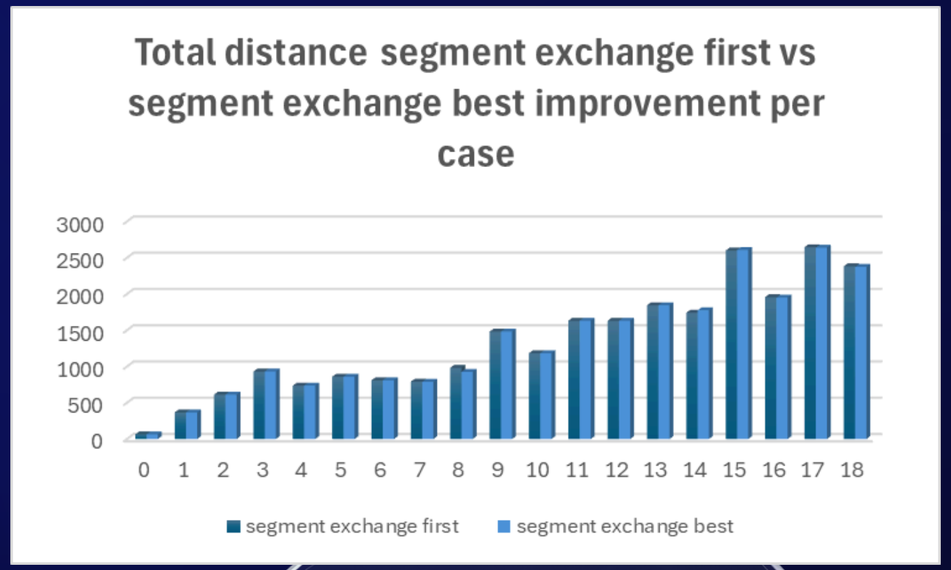



# SEGMENT EXCHANGE BEST IMPROVEMENT

- The algorithm begins by constructing initial routes for vehicles, ensuring that each route adheres to capacity limits and time windows. It selects the nearest feasible node for each vehicle's route.
- Subsequently, it employs a segment exchange best improvement method. This evaluates potential swaps between vehicle routes to identify the most beneficial exchanges, iteratively optimizing the routes until no further improvements can be made.
- Finally, the total travel distance is calculated, and results—including the number of vehicles used and route details—are saved to an Excel file for analysis. This approach effectively addresses the Vehicle Routing Problem with Time Windows (VRPTW) by enhancing delivery efficiency while respecting constraints.

# SEGMENT EXCHANGE BEST ANALYSIS

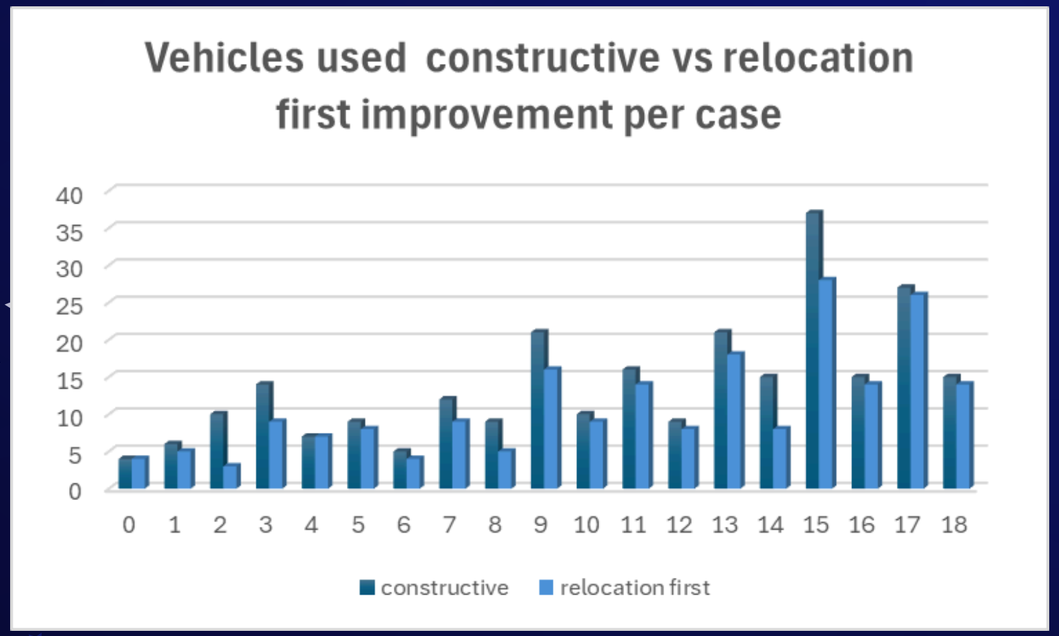


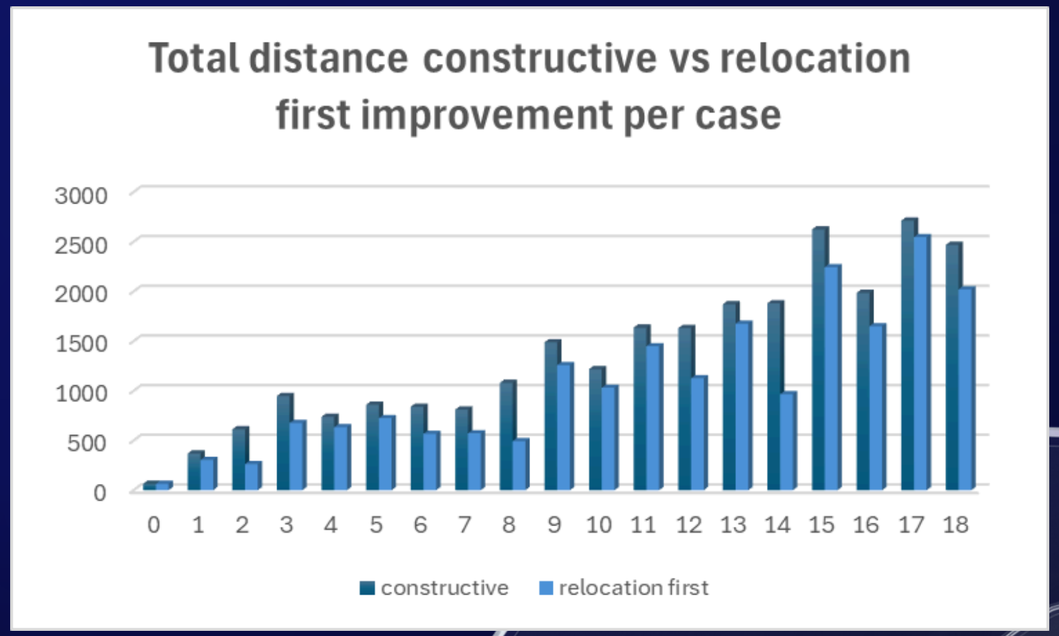



# Relocation first improvement

The code implements a solution for the Vehicle Routing Problem with Time Windows (VRPTW), which focuses on optimizing the delivery routes for a fleet of vehicles while adhering to constraints such as vehicle capacity and customer time windows. It defines classes for nodes (representing delivery locations) and vehicles, and uses a greedy approach to construct initial routes based on distance and feasibility. The algorithm then refines these routes using a "relocation first improvement" strategy, which evaluates potential relocations of nodes within and between vehicle routes to minimize total travel distance. Results, including route details and performance metrics, are saved to an Excel file for further analysis. This tool aims to enhance logistical efficiency in transportation scenarios.

# RELOCATION FIRST ANALYSIS

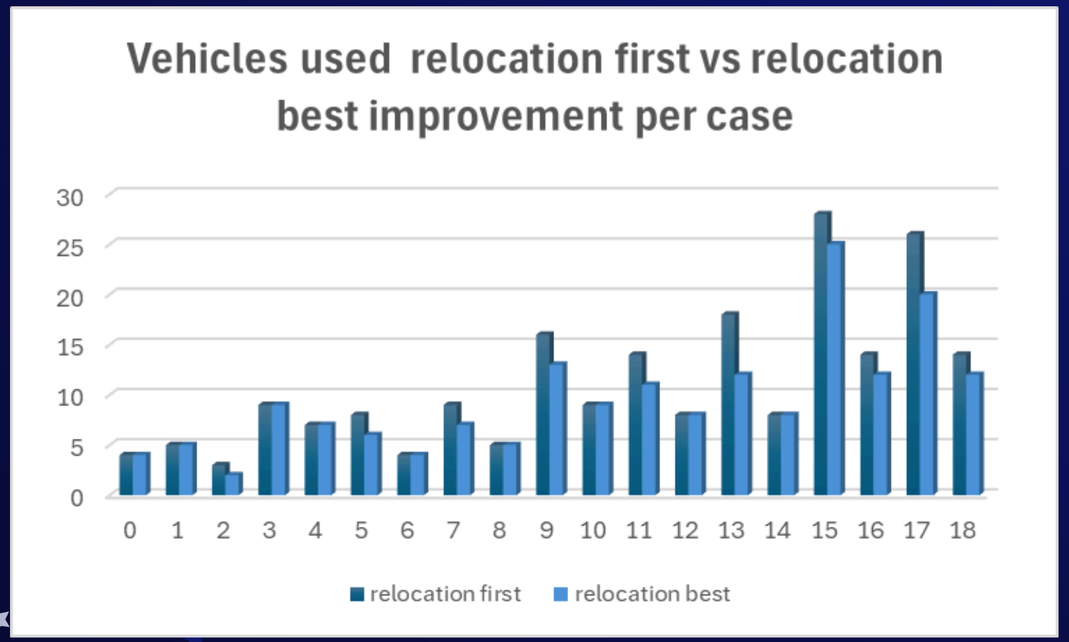


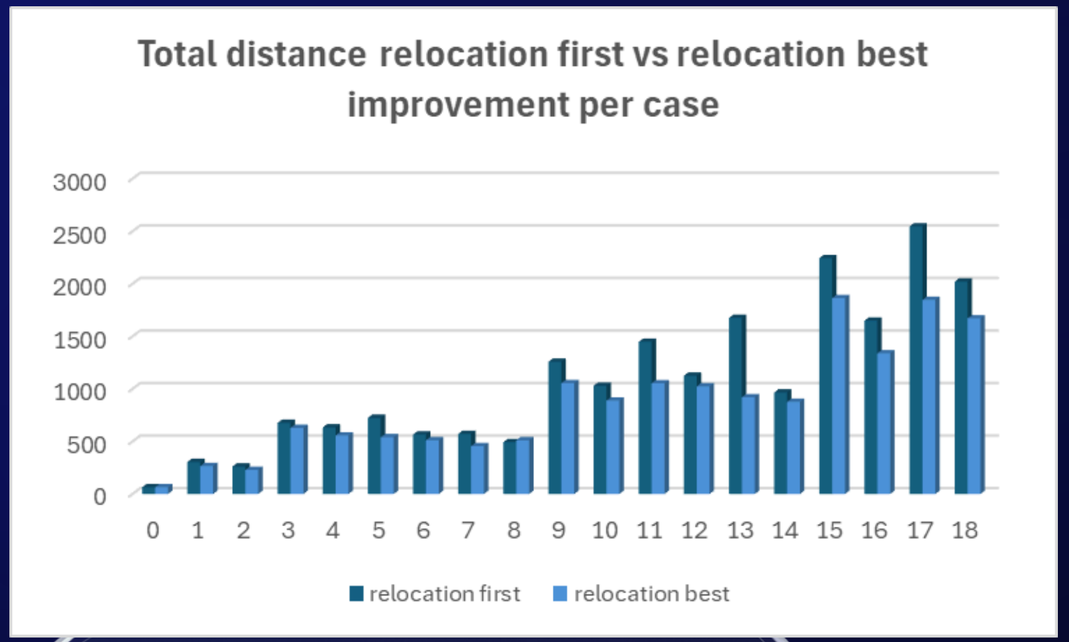



# RELOCATION BEST ANALYSIS

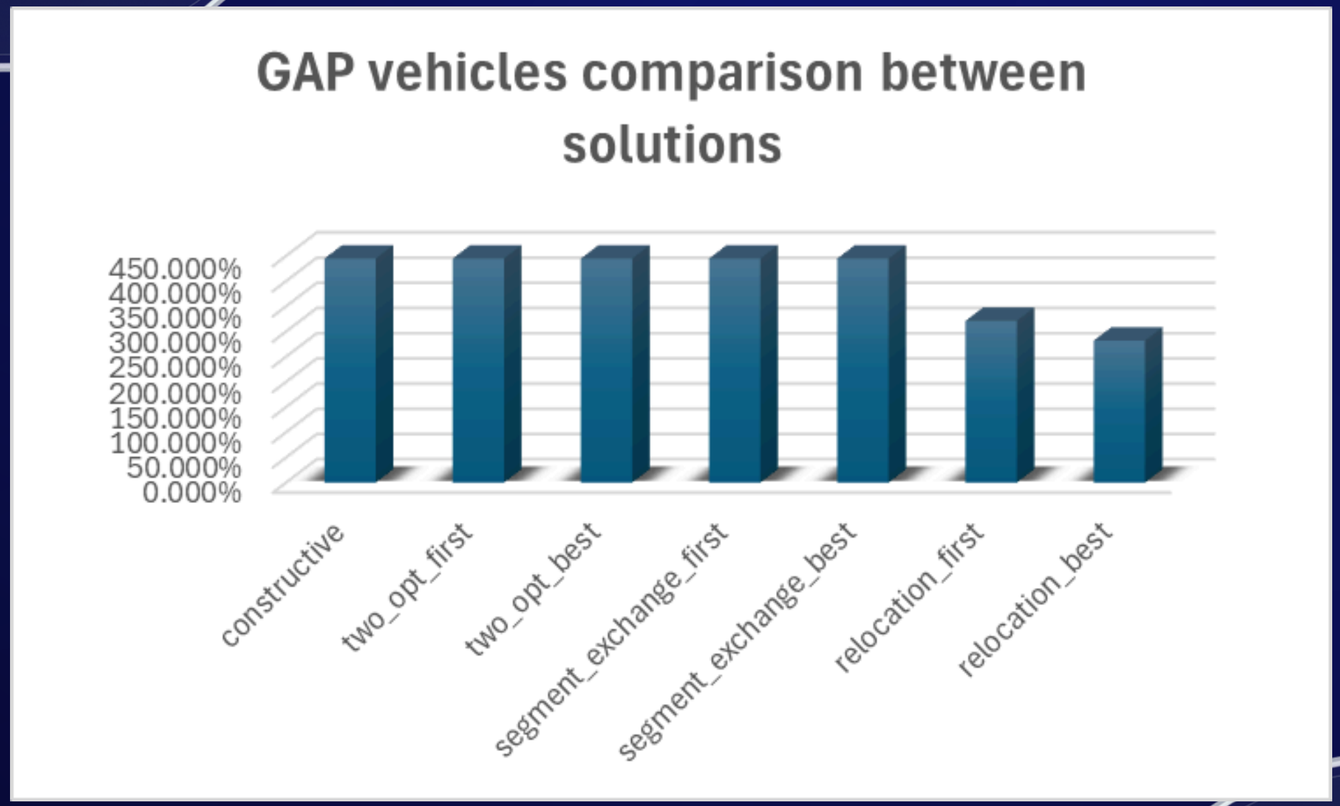
This code is an enhanced implementation of the Vehicle Routing Problem with Time Windows (VRPTW) that utilizes a "relocation best improvement" strategy for optimizing vehicle routes. It defines classes for nodes (representing delivery points) and vehicles, and calculates distances between nodes using Euclidean distance. The main components include constructing initial routes for a fleet of vehicles based on feasibility and demand, followed by refining these routes by evaluating potential node relocations both within and between vehicle routes. The algorithm seeks the best possible improvement in total travel distance for each relocation and updates the routes accordingly. Results, including the optimized routes and associated metrics, are saved to an Excel file for analysis, providing an efficient solution to complex logistical challenges.

# RELOCATION BEST ANALYSIS



# GAP VEHICLES



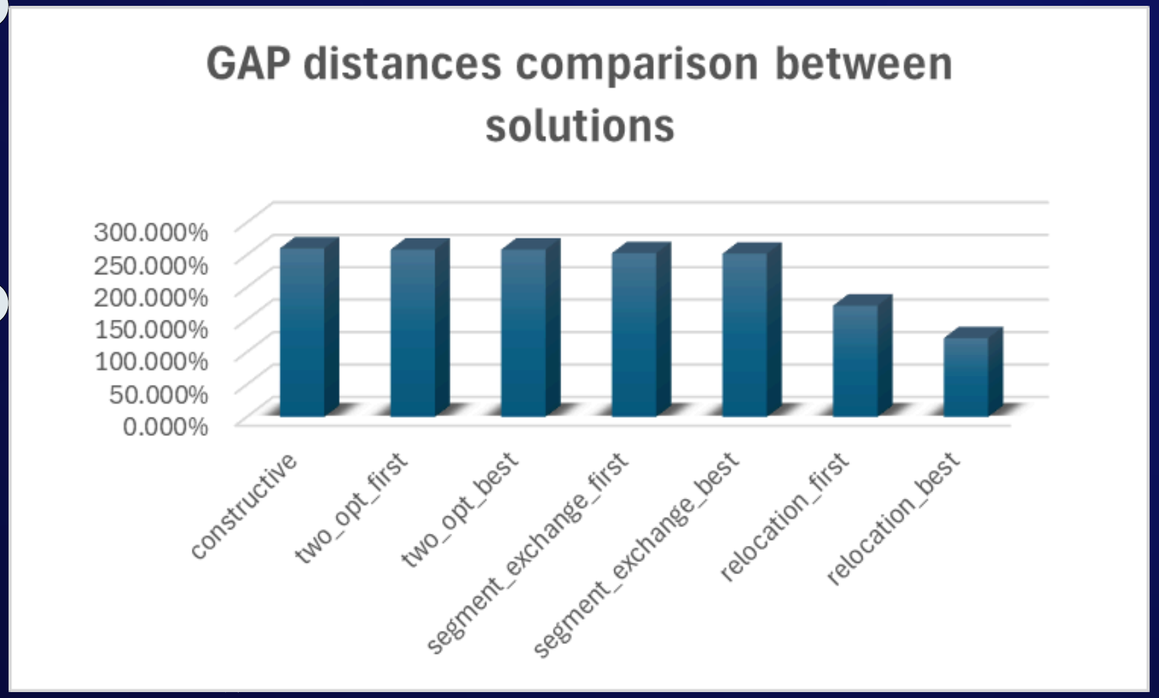

| Algorithm | GAP vehicles |
|---|---|
| constructive | 444.861% |
| two_opt_first | 444.861% |
| two_opt_best | 444.861% |
| segment_exchange_ | 444.861% |
| segment_exchange_ | 444.861% |
| relocation_first | 321.049% |
| relocation_best | 281.836% |

# GAP VEHICLES ANALYSIS

1. Constructive, Two-Opt, and Segment Exchange algorithms (both first and best) all show the worst performance, with a high GAP of 444.861%, indicating inefficient vehicle utilization.
2. Relocation algorithms perform much better:
3. Relocation First reduces the GAP to 321.049%.
4. Relocation Best achieves the lowest GAP of 281.836%, making it the most efficient in minimizing vehicle use.

- The Relocation heuristic outperforms the other algorithms, with Relocation Best being the most effective at reducing vehicle usage.

# GAP distances




| Algorithm | GAP distances |
|---|---|
| constructive | 260.081% |
| two_opt_first | 258.266% |
| two_opt_best | 258.266% |
| segment_exchange_ | 252.699% |
| segment_exchange_ | 251.859% |
| relocation_first | 171.821% |
| relocation_best | 121.623% |

# GAP DISTANCE ANALYSIS

1. Constructive, Two-Opt, and Segment Exchange algorithms (both first and best) have high GAP values, ranging from 251.859% to 260.081%, indicating inefficient and longer routes.
2. Relocation algorithms perform significantly better:
3. Relocation First reduces the GAP to 171.821%.
4. Relocation Best achieves the lowest GAP of 121.623%, indicating the most optimized and shortest routes.

- The Relocation heuristic excels at minimizing route distances, with Relocation Best being the most efficient in reducing the distance GAP. This makes it the most effective algorithm for optimizing travel distances.
-

# Conclusions

- Constructive, Two-Opt, and Segment Exchange algorithms consistently underperformed, with a GAP of 444.861% for vehicles and GAPs ranging from 251.859% to 260.081% for distances, indicating inefficiency in both metrics.
- The Relocation algorithms significantly outperformed the others, with Relocation Best achieving the lowest GAP of 281.836% for vehicles and 121.623% for distances, making it the most effective in optimizing vehicle usage and minimizing route distances.
- First also showed good results, with GAPs of 321.049% in vehicles and 171.821% in distances, indicating that both Relocation variants are reliable choices for optimizing vehicle and route efficiency.
- The Relocation heuristic is the most effective algorithm for minimizing both vehicle count and travel distance, making it the preferred choice for solving the routing problem.

# Thank You