HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Fundamentals of Optimization

MINIPROJECT REPORT

Problem: Minimizing Maximal Route Length in Multi-Taxi Mixed-Transport Systems

Name: Tô Lê Quang
Student ID: 202416741
Mail: Quang.TL2416741@sis.hust.edu.vn

A Hybrid Metaheuristic Framework Centered on Adaptive Large Neighborhood Search for the Min–Max Vehicle Routing Problem

Tô Lê Quang
Hanoi University of Science and Technology
quang.tl2416741@sis.hust.edu.vn

Abstract

The Vehicle Routing Problem (VRP) with a Min–Max objective (Min–Max VRP) is a crucial variant of the classical VRP that focuses on minimizing the maximum cost among all vehicle routes, thereby promoting a balanced workload distribution across the fleet. This problem arises in numerous real-world optimization scenarios, including ride-sharing, last-mile delivery, and fleet coordination, and is particularly relevant to complex transportation and logistics systems faced by enterprises. Recent advancements in general-purpose optimization, particularly metaheuristic algorithms, have demonstrated superior performance over both traditional exact methods and specialized toolkits such as backtracking and Google OR-Tools. Among these, the adaptive large neighborhood search (ALNS) has emerged as one of the most effective global search strategies to solve VRP variants due to its flexibility and adaptive nature, especially when dealing with complex constraints and large-scale number of vehicles. In this mini-project, I propose a customized framework based on the ALNS paradigm to efficiently address the Min–Max VRP. The experimental results demonstrate the practical effectiveness of the proposed method in solving this class of problems and further assess its performance through key evaluation metrics.

I. Introduction

The routing problem has its origins in the classical Traveling Salesman Problem (TSP), first formulated in the early 20th century, which seeks the shortest route visiting a set of locations exactly once. As transportation systems became increasingly complex and subject to more real-world constraints, the single-route TSP evolved into multi-route variants, most notably the Vehicle Routing Problem (VRP), introduced by Dantzig and Ramser in 1959. Early solution approaches relied heavily on linear programming and exact algorithms, which were computationally feasible only for small-scale instances. As the number of nodes increases, the number of decision variables and possible combinations grows exponentially, leading to the well-known combinatorial explosion.

Since the 1990s, the rise of metaheuristic algorithms has significantly advanced the ability to solve large-scale, highly constrained VRPs. Techniques such as Simulated Annealing, Genetic Algorithms, Tabu Search, and Ant Colony Optimization have been widely applied to various VRP variants, including those that explicitly address load balancing objectives, such as minimizing the maximum route length. A prevailing trend in recent years involves the integration of metaheuristics with modern optimization libraries, such as Google OR-Tools to develop hybrid methods that leverage the strengths of each component. In this context, the design of an effective initial solution plays a crucial role in improving the performance and quality of the solution of metaheuristic frameworks.

Despite substantial progress, several real-world challenges remain when applying VRP algorithms, particularly in scenarios with complex constraints such as time windows, vehicle capacity limits, and requirements for balanced workload distribution across routes. In modern logistics systems, where fairness and efficient utilization of fleet resources are increasingly emphasized, minimizing imbalance among routes is considered as important as reducing total operational costs.

A prominent VRP variant addressing these concerns is the Min-Max Vehicle Routing Problem (Min-Max VRP), where the objective is to minimize the length of the longest route among all vehicles. This formulation is especially relevant in practical contexts that require equitable workload distribution among drivers, reduction of service-time discrepancies, and optimized fleet deployment under resource constraints.

However, due to its extremal objective and tight coupling between vehicle routes, the Min–Max VRP exhibits a rugged and deceptive search space, where naive optimization tends to favor improvements on already short routes rather than reducing the length of the longest one. Traditional exact solvers are computationally infeasible for large instances, and general-purpose metaheuristics often fail to maintain global balance without custom adaptations. In this context, Adaptive Large Neighborhood Search (ALNS) has emerged as a highly flexible and effective metaheuristic framework, capable of handling complex constraints and exploring the solution space via adaptive combinations of problem-specific removal and insertion heuristics.

In this report, I propose and evaluate an ALNS-based framework specifically designed to address the Min–Max Vehicle Routing Problem with a large number of passenger and parcel requests. The key contributions of this work are as follows:

- Integration of Google's OR-Tools as a benchmark solver to provide a comparative baseline for performance evaluation.
- Development of a simple yet effective greedy initialization strategy to construct a feasible starting solution.
- Implementation of a clustering mechanism to divide the initial solution into smaller, manageable subproblems.

- Application of Adaptive Large Neighborhood Search (ALNS) for both local refinement within clusters and global solution improvement across routes.

II. Background and Related Works

A. Foundations of Adaptive Large Neighborhood Search

The Adaptive Large Neighborhood Search (ALNS) algorithm was first introduced by Røpke and Pisinger in 2006 [1] as a flexible metaheuristic framework designed to solve complex variants of the Vehicle Routing Problem with Time Windows (VRPTW). The algorithm begins by constructing an initial feasible solution using a simple heuristic, typically a greedy method.

In each iteration, a portion of the current solution is selectively removed using one of several destroy operators and is subsequently reinserted into the solution using a repair operator. This destroy–repair mechanism allows the algorithm to explore large and diverse neighborhoods that are often inaccessible through traditional local search, thereby enabling it to escape local optima.

At the core of ALNS lies its adaptive operator selection strategy, which determines the pair of destroy and repair operators to be applied in each iteration. Over time, the weights assigned to each operator are updated based on their recent performance. This is achieved through a reinforcement-based scoring system that rewards operators contributing to solution improvement or diversification, thus guiding the search toward more promising regions of the solution space.

Finally, ALNS incorporates a solution acceptance mechanism based on Simulated Annealing (SA), which occasionally accepts worse solutions with a probability that decreases over time. This acceptance criterion balances exploration and exploitation, improving the robustness of the search process.

B. Further Improvements and Integrated ALNS Frameworks

Having demonstrated its effectiveness and flexibility, ALNS has been further developed in terms of both its internal algorithmic structure and its integration into hybrid metaheuristic frameworks.

On the one hand, the study by Chentli *et al.* [2]—Selective ALNS for Profitable Tour with Simultaneous Pickup and Delivery—enhanced the original ALNS by introducing two additional insertion operators (Random Insertion and Scoring Insertion), combined with an optimized Simulated Annealing temperature adjustment strategy. These modifications were designed to improve the diversity of the search space and were shown to be particularly effective for problem instances involving 50 to 199 destinations.

On the other hand, the work of Li *et al.* [3]—Adaptive Large Neighborhood Search for the Crowd-Shipping Problem with Time Windows—proposed a hybrid metaheuristic framework centered on ALNS, integrating Mixed Integer Programming (MIP), heuristic initialization, and ALNS as the core improvement mechanism. This framework reflects a growing trend in the literature to combine exact optimization techniques with heuristic methods in order to tackle increasingly complex routing problems.

III. Problem Formulation

This report considers a shared transportation problem that integrates both passenger transportation and parcel delivery. The system involves K taxis, all of which depart from and return to a central depot (node 0), serving N passengers and M parcels. The objective is to design feasible routes that minimize the maximum route length among all taxis, thereby achieving load balancing and service fairness.

A. Graph Representation and Overview

Consider a transportation system comprising K taxis, all of which depart from a common depot (node 0), tasked with fulfilling the service demands of N passengers and M parcels. *(The problem can be described in Figure 1.)*

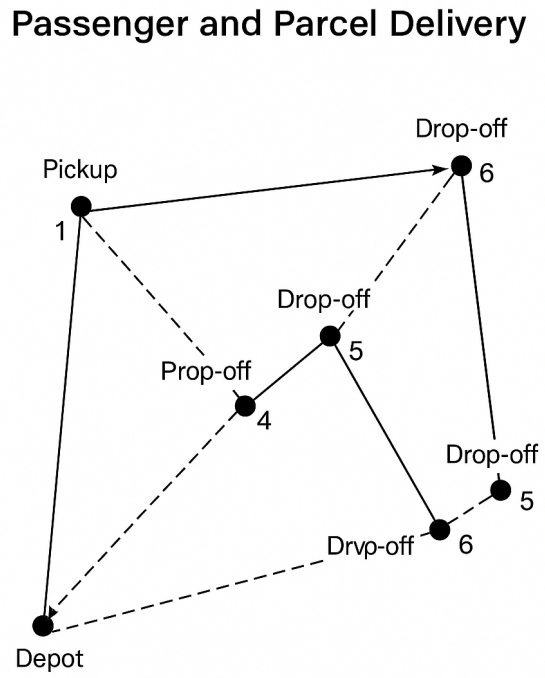

Fig. 1. Visualization of Pickup and Drop-off Points for Passengers and Parcels in a Shared Taxi Routing Scenario.

Let the problem be defined over a complete directed graph $G = (V, E)$, where:

- $V = \{0, 1, 2, ..., 2N + 2M\}$ is the set of all nodes including the depot (node 0), pickup and drop-off points for passengers and parcels.
- E is the set of edges between nodes, with cost $d(i, j)$ representing the travel distance from node i to node j.
- K: number of taxis.
- N: number of passenger requests.
- M: number of parcel requests.
- $Q[k]$: capacity of taxi k for handling parcels.
- $q[j]$: quantity/demand of parcel request j.
- $x \in \{0, 1\}$: binary variable, equals 1 if taxi k travels directly from node i to node j; 0 otherwise.
- $d[i][j]$: the cost to arrive from node i to j.
- L_k: total travel distance of taxi k.
- $L_{\max}$: maximum route length among all taxis.

B. *Mathamatical Modelling*

Input:

- Number of passenger N, parcel requests M and taxis K.
- Each passenger request i $(1 \leq i \leq N)$ is associated with a pickup point i and a drop-off point $i + N + M$.
- Each parcel request j $(1 \leq j \leq M)$ has a pickup location $j + N$ and a drop-off point $j + 2N + M$.
- All passenger trips must be served as non-stop direct routes, meaning no intermediate stops are allowed between their pickup and drop-off locations.
- Each taxi k has a parcel handling capacity denoted by $Q[k]$, and each parcel j has a specific demand quantity $q[j]$.
- The travel cost or distance between any pair of nodes is given by the matrix $d[i][j]$.

Constraints:

- All passenger and parcel requests must be served exactly once.
- Each passenger must be served by a direct, uninterrupted route.
- The total parcel quantity on any taxi route must not exceed the taxi's capacity.

Objective:
Minimize the maximum route length (or cost) among all taxi routes, thereby achieving a balanced and efficient allocation of transportation resources.
Output:

- The first line: The number of taxis K.
- For line k: The number of destination arrived by taxi k.
- For line $k + 1$: The route of taxi k.

IV. METHODOLOGY

A. *Solution benchmark: Google's OR-Tools*

1) Problem Modeling: This report considers a shared transportation problem that integrates both passenger transportation and parcel delivery. The system involves K taxis, all of which depart from and return to a central depot (node 0), serving N passengers and M parcels. The objective is to design feasible routes that minimize the maximum route length among all taxis, thereby achieving load balancing and service fairness.

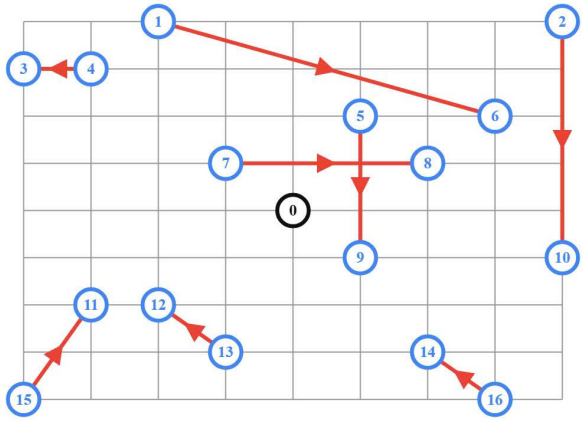

Fig. 2. Illustration of the problem structure as modeled in OR-Tools

The problem is represented on a directed graph *(describe in Figure 2)* $G = (V, E)$, where:

- V denotes the set of all nodes, including:
  - the depot (node 0),
  - pickup and drop-off nodes for passengers, and
  - pickup and drop-off nodes for parcels.
- Each route begins and ends at the depot.
- For each passenger request i $(1 \leq i \leq N)$, the pickup node is i and the corresponding drop-off node is $i + N + M$.
- For each parcel request j $(1 \leq j \leq M)$, the pickup node is $j + N$, and the drop-off node is $j + 2N + M$.

2) Solver Initialization and Cost Function Registration: To solve the problem, I proceed as follows:

- Create Solver Components:
  - *manager*: An instance of `RoutingIndexManager`, responsible for mapping logical node indices.
  - *routing*: An instance of `RoutingModel`, which encapsulates the routing logic, including cost functions, constraints, and search strategies.
- Register Distance Function:
  - Define and register a transit callback using `RegisterTransitCallback` to represent travel distances $d(i, j)$.
  - Use `SetArcCostEvaluatorOfAllVehicles` to assign the distance function to all vehicles uniformly.

3) Constraints Encoding: To accurately model the problem requirements, several key constraints are encoded within the OR-Tools routing framework as follows:

- The `AddDimension` method is utilized to define a cumulative dimension (e.g., distance), which enables tracking of accumulated metrics such as travel distance or time across each route.

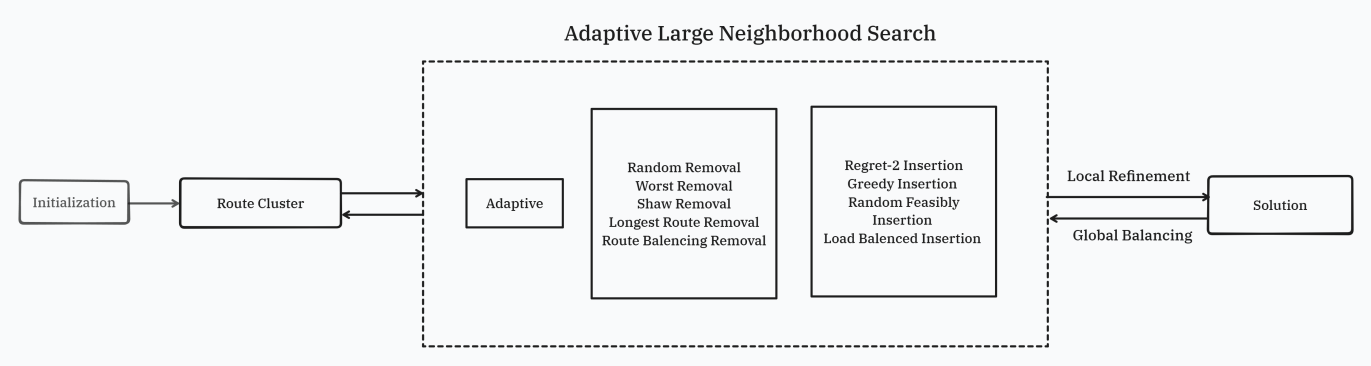

Fig. 3. Illustration the overall structure of the proposed hybrid metaheuristic framework, which is centered on an Adaptive Large Neighborhood Search.

- The `NextVar` variable is used to capture the routing sequence, explicitly specifying the successor node for each visit and enforcing route continuity.
- The combination of `VehicleVar` and `CumulVar` allows the model to impose precedence constraints between pickup and delivery nodes, ensuring that for each request, the drop-off occurs after the corresponding pickup on the same vehicle.

4) Objective Definition: All benchmark instances were run with a time limit of 60 seconds per instance to ensure fairness in comparison with the ALNS approach. To promote balanced route planning, the objective function is designed to minimize the length of the longest individual route among all vehicles. This is achieved using the method `SetGlobalSpanCost-Coefficient(coefficient)`, which instructs OR-Tools to incorporate the global span, defined as the difference between the maximum and minimum cumulative distances across all routes in the cost function. By assigning a sufficiently large coefficient, the solver is guided to favor solutions that reduce route imbalance, thereby improving overall fairness and efficiency in the distribution of vehicle workload.

*B. **Proposed Structure: Hybrid Metaheuristic Centered on ANLS***

1) Overview: The proposed hybrid meta-heuristic structure follows these steps: (*Figure 3*)

The proposed hybrid metaheuristic framework is based on an adaptive large neighborhood search (ALNS) algorithm to minimize the maximum route length across all vehicles. To improve the performance of ALNS in medium instances (50-199 nodes), the framework uses an initial clustering procedure that partitions requests into subsets whose total route length lies within a desirable range.

During the optimization process, once an improved (local refine) is identified, a global refinement phase is performed to further reduce the imbalance among the routes. At the end of each iteration, the new solution is evaluated using a Simulated Annealing (SA) acceptance criterion, which probabilistically allows worse solutions to escape local optima and encourages search diversification.

Compared to existing approaches, the proposed framework introduces two notable enhancements.

Firstly, key parameter configurations are selected and adjusted based on insights from prior studies, ensuring the framework maintains competitive performance while remaining generalizable. These parameters include operator scoring weights, acceptance cooling schedules, and removal ratios, all of which are informed by successful configurations in similar ALNS applications.

Secondly, while most existing ALNS-based methods are designed and evaluated on small to medium-sized instances (typically 50 to 200 nodes), the proposed framework is specifically optimized for large-scale scenarios involving more than 500 pickup and delivery points. In such settings, existing ALNS frameworks often suffer from scalability issues, including computational bottlenecks, operator stagnation, and imbalance persistence. To address these challenges, the proposed method incorporates a scalable initialization via clustering and a global refinement step that dynamically redistributes load among routes, thereby enhancing its robustness on real-world, high-volume transportation networks.

2) Stage 1: **Initialization**

According to previous studies on ALNS, the quality of the final solution is not highly dependent on the initial one, due to the algorithm's ability to extensively explore the solution space. Therefore, the proposed framework adopts a simple greedy initialization procedure. At each step, the request pair (pickup and drop-off) that introduces the smallest incremental cost is inserted into a feasible route.

This greedy strategy results in an initial solution that is both feasible and relatively balanced in terms of route cost. The cost associated with inserting a new request is evaluated as follows:

$$\text{Cost} = d[0][\text{pickup}] + d[\text{pickup}][\text{dropoff}]$$

As a result, the initial solution achieves a relatively balanced distribution across all routes, which is particularly suitable for the min–max vehicle routing objective. This objective favors configurations with low variability in route lengths, often characterized by a small standard deviation among routes.

3) Stage 2: **Clustering**

The Adaptive Large Neighborhood Search (ALNS) algorithm has been shown to perform effectively on problem instances with fewer than 200 destinations. However, in this mini-project, the problem scale exceeds 2,000 nodes, making the direct application of ALNS computationally inefficient.

To address this, a clustering procedure is introduced to decompose the initial solution into smaller subproblems of manageable size. In each iteration, the current solution is partitioned into multiple clusters, each containing no more than 120 nodes. This ensures that subsequent ALNS operations remain computationally tractable while preserving the quality of the search within each local neighborhood. For sample size under 100 destinations, the algorithm will execute directly after the initialization stage without any division.

4) Stage 3: **Executing Adaptive Large Neighborhood Search**

a) Destroy Operator:

- *Random Removal:* In each iteration, this operator randomly removes a proportion of requests (both pickup and drop-off nodes) from the current solution.
- *Worst Removal:* This operator evaluates the potential cost savings of removing each request (i.e., a pickup–drop-off pair) from the solution. It then removes a fixed number of requests that contribute the most to the total route cost.
- *Shaw Removal:* This operator removes a group of related requests, based on a similarity measure such as geographical proximity or time window closeness. Starting from a randomly selected base request, it iteratively selects and removes similar requests to exploit structural patterns.
- *Longest Route Removal:* At each iteration, the operator identifies the routes with the highest costs (usually the top two) and removes a selected number of requests from them.
- *Load-Balancing Removal:* This operator targets routes whose costs significantly exceed the average. From these unbalanced routes, it removes requests that most contribute to the cost imbalance, thereby enabling better load redistribution across the fleet.

Each operator represents a distinct trade-off between intensification and diversification. The adaptive mechanism selects among them based on their recent performance. Below is a description of each insertion strategy:

b) Insertion Operator:

- *Regret-2 Insertion:* For each unassigned request r, this operator identifies the two best feasible insertion positions across all routes and calculates a regret value:

$$\text{Regret}_2(r) = C_2(r) - C_1(r)$$

  where $C_1(r)$ and $C_2(r)$ denote the insertion costs at the best and second-best positions, respectively. The request with the highest regret is inserted first, under the rationale that delaying its assignment may incur significant future cost. This operator is known for its balance between exploration and exploitation.

- *Greedy Insertion:* At each step, this heuristic selects the request–route–position combination that yields the minimum cost increase:

$$\Delta C(r,k) = \min_{(i,j)} \left[\text{Cost}(R_k^{+r(i,j)}) - \text{Cost}(R_k) \right]$$

  where R_k is the current route and $R_k^{+r(i,j)}$ represents the route after inserting request r between positions i and j. This strategy is computationally efficient and tends to improve local optimality.
- *Random Feasibility Insertion:* This operator selects an unassigned request at random and inserts it into a randomly selected feasible position within any route. While it does not exploit cost information, it enhances diversification and is particularly useful in the early iterations of the search to avoid premature convergence.
- *Load-Balanced Insertion:* This operator biases insertion toward underloaded routes by penalizing routes with higher current cost. The adjusted cost function is defined as:

$$\Delta C'(r,k) = \Delta C(r,k) \cdot \left(1 + \lambda \cdot \frac{\text{Cost}(R_k)}{\overline{\text{Cost}}}\right)$$

  where $\overline{\text{Cost}}$ is the average route cost across all vehicles and $\lambda > 0$ is a control parameter. This encourages workload balancing and indirectly minimizes the maximum route cost. The λ range is optimal on the interval $[0.5, 2]$.

c) Adaptive: Heart of ALNS

To balance exploration and exploitation during the search process, an adaptive mechanism is employed to select pairs of removal and insertion operators at each iteration. This mechanism assigns a dynamic weight to each operator, reflecting its historical contribution to solution improvement. At the start of each iteration, a removal-insertion pair is selected through a weighted random selection, where the probability of selecting an operator is proportional to its current weight. A minimum selection probability is enforced to ensure sufficient exploration of less frequently chosen operators. After the operators are applied and a new solution is generated, the quality of the solution is evaluated. If the solution meets predefined criteria (e.g., improvement over the current or best-known solution), the weights of the responsible operators are updated accordingly, typically through a reward-based scheme. This adaptive process enables the algorithm to gradually favor more effective operators, enhancing overall performance and robustness across different problem instances.

Several parameters play a critical role in controlling the operator selection behavior in the adaptive mechanism of ALNS. These parameters govern the balance between exploration and exploitation during the search process:

- *Decay rate* γ: This parameter controls the exponential decay of the exploration rate over time. At the early stages of the search, exploration is prioritized to allow broader coverage of the solution space. As the search progresses,

the algorithm gradually shifts toward exploitation by reducing ε, thereby favoring operators with better historical performance.

- *Minimum probability* $\varepsilon_{\min}$: This lower bound ensures that the selection mechanism retains a non-zero probability of choosing any operator, even in the final iterations. This helps maintain a minimum level of exploration and avoids premature convergence to suboptimal strategies.
  Therefore, the probability of randomly choosing an operator parameter is:

$$P = \max(\varepsilon_{min}, \varepsilon - \gamma * t)$$

After each iteration, the operator weights are updated based on the difference between the previous and the current solution. Specifically, an operator receives a reward when it contributes to a reduction in cost, or a penalty otherwise.
Let ω_i denote the weight of operator i. The weights are updated using a decay-based forgetting mechanism:

$$\omega_i \leftarrow \omega_i \cdot \lambda + r_i,$$

where $\lambda \in (0, 1)$ is the decay factor (e.g., 0.98), and r_i is the reward assigned to operator i based on its performance in the current iteration:

$$r_i = \begin{cases} 10 & \text{if the solution improves,} \\ -0.5 & \text{otherwise (penalty).} \end{cases}$$

This adaptive weighting strategy not only guides the selection of operators based on recent performance but also prevents stagnation caused by outdated reward accumulation. By discounting past rewards through the decay factor λ, the algorithm maintains a dynamic balance between exploration and exploitation.

5) Iterative Solution Refinement via ALNS and SA: **Global Optimization and Adaptive Acceptance Strategy**

After independently optimizing each cluster, the solutions often converge to local optima due to the limited neighborhood search space within each subproblem. To enhance global optimality,the Adaptive Large Neighborhood Search (ALNS) strategy is applied at the overall solution level. Specifically, I iteratively selects and re-optimizes the pair of routes with the highest cost disparity, which effectively targets the most unbalanced parts of the solution.
Let S be the current solution and S' be a new candidate solution generated via remove-insert operators (neighborhood move), where:

$$S' = \text{ALNS}(S)$$

The pair of routes (r_i, r_j) selected for refinement satisfies:

$$(r_i, r_j) = \arg \max_{(r_p, r_q) \in \mathcal{R}} |\text{Cost}(r_p) - \text{Cost}(r_q)|$$

where $\mathcal{R}$ is the set of all pairs of routes in the current solution.

To avoid premature convergence, I implements a Simulated Annealing (SA)-based acceptance criterion. A new solution S' is accepted with probability:

$$P_{\text{accept}} = \begin{cases} 1, & \text{if } f(S') < f(S) \\ \exp\left(-\dfrac{f(S') - f(S)}{T}\right), & \text{otherwise} \end{cases}$$

where:
- $f(\cdot)$ is the objective function (e.g., total cost or total time).
- T is the current temperature.
The temperature T decreases over iterations using an exponential cooling schedule:

$$T_{k+1} = \alpha T_k, \quad \alpha \in (0, 1)$$

To escape stagnation, if no improvement is observed for consecutive iterations, the framework performs a reheating step.

$$T \leftarrow T_0 + \Delta t$$

where T_0 is the initial temperature.
This combination of adaptive neighborhood selection and probabilistic acceptance improves the diversity of solutions and helps escape local optima.

C. Evaluation Metrics

To comprehensively assess the performance of the proposed methods, I employ the following three evaluation metrics:

- Maximum Route Length (MRL): This metric reflects the length of the longest route among all taxis. Since the objective of our problem is to minimize the maximum workload, MRL serves as the primary criterion for evaluating solution quality. It is defined as:

$$\text{MRL} = \max_{k=1,\ldots,K} L_k$$

  where L_k denotes the total distance traveled by vehicle k, and K is the total number of vehicles.
- Average Route Length (ARL): Defined as the mean length of all taxi routes, this metric provides an overall measure of system efficiency. It helps to evaluate the global performance beyond only the worst-case route. The formula is:

$$\text{ARL} = \frac{1}{K} \sum_{k=1}^{K} L_k$$

- Standard Deviation of Route Lengths (σ): This metric quantifies the variability in route lengths across all taxis. A lower standard deviation indicates a more balanced distribution of workloads, which is desirable in practical deployment scenarios. It is computed as:

$$\sigma = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (L_k - \text{ARL})^2}$$

- Execution Time (sec): The total wall-clock time measured from the beginning of the algorithm to its termination. This metric includes all computational overheads, such

TABLE I
DATASET CONFIGURATIONS USED IN THE EXPERIMENTS

Instance	Passengers (N)	Parcels (M)	Taxis (K)	Total Nodes	Max Capacity	Benchmark
1	100	100	10	401	134	1491
2	450	450	40	1801	137	1681
3	10	10	2	41	121	929
4	400	400	40	1601	137	1426
5	3	3	2	13	21	67
6	5	5	2	21	101	440
7	100	100	10	401	134	1491
8	200	200	10	801	137	2676
9	500	500	50	2001	137	1493
10	400	400	40	1601	137	1479
11	300	300	30	1201	137	1493

as route evaluations, insertion/removal operations, and local refinements. It reflects the practical efficiency and scalability of the method.

- Memory Usage (KB): The peak memory consumption during algorithm execution, measured using [tool/library].

Furthermore, to analyze the optimization behavior over time, I plots the convergence curves of each algorithm. These plots illustrate how the objective value (i.e., the maximum route length) evolves with increasing iterations, providing insights into both the speed of convergence and the quality of the final solutions.

V. EXPERIMENTS

A. Experimental Setup

1) Environment: All experiments were conducted on the Google Colab platform, which offers a cloud-based Python environment with support for GPU acceleration and configurable computational resources. The implementation was executed using Python 3.x with relevant scientific computing libraries (e.g., NumPy, Matplotlib, and psutil).

2) Dataset Description: The data set consists of 11 synthetic instances, varying in the number of passenger requests (N), parcel requests (M), and taxis (K). Each instance specifies the maximum capacity for parcel delivery per taxi and the corresponding total number of nodes in the graph. The benchmark column indicates the reference cost or the best-known solution for comparison.

(The detailed specifications of the input instances used in the experiments are presented in Table 1.)

B. Baseline Methods

To establish a point of reference for evaluating the proposed approach, the Google OR-Tools library is utilized to generate baseline solutions for selected problem instances. OR-Tools provides a powerful and widely adopted optimization framework that includes heuristics and exact methods for solving routing problems.

Table II summarizes the performance of the baseline method on three representative instances where a feasible solution could be obtained. Key evaluation metrics include the maximum route length (MRL), average route length (ARL), standard deviation (σ), execution time (in milliseconds), and memory usage (in kilobytes).

TABLE II
PERFORMANCE OF BASELINE METHOD ON TEST CASES

Instance	MRL	ARL	σ	Time	Memory
3	672	658.0	19.8	290	15,851
5	54	53.5	0.71	290	14,427
6	348	345.5	3.54	290	14,768

C. Results and Discussion

1) Performance Comparison: To evaluate the effectiveness of the proposed method, a summary of the experimental results is presented in Table III. The table reports key performance indicators, including maximum route length, average route length, standard deviation, execution time, and memory usage across selected test instances.

2) Performance Comparison: Table II and Table III report the experimental results obtained from the baseline method (using Google's OR-Tools) and the proposed ANLS algorithm, respectively. The comparison highlights key performance indicators, including maximum route length (MRL), average route length (ARL), standard deviation (σ), and execution time.

Due to scalability limitations, Google's OR-Tools encounters difficulties in handling large-scale instances—particularly when the number of destinations exceeds approximately 40. Consequently, only instances 3, 5, and 6 were successfully solved by the baseline framework, while the proposed method remained applicable across all tested instances.

For the feasible cases, the proposed method demonstrates superior performance. In instance 3, although the MRL of ANLS (811) is slightly higher than the baseline (672), the standard deviation is significantly lower (3.54 vs. 19.8), suggesting a more balanced route distribution. In instance 5, both methods yield the same MRL (54), but ANLS achieves perfect balance

TABLE III
PERFORMANCE OF THE PROPOSED METHOD ON TEST INSTANCES

Instance	N (Passengers)	M (Parcels)	K (Taxis)	MRL	ARL	σ	Time (s)
1	100	100	10	1278	1247.8	28.62	119.57
2	450	450	40	1411	1336.35	38.02	156.85
3	10	10	2	811	808.5	3.54	0.13
4	400	400	40	1197	1146.05	32.61	261.08
5	3	3	2	54	54	0	0.03
6	5	5	2	414	398.5	21.92	0.04
7	100	100	10	1278	1247.8	28.62	119.57
8	200	200	10	2558	2532	41.42	115.87
9	500	500	50	1282	1216.15	41.8	178.59
10	400	400	40	1250	1192.9	41.91	200.83
11	300	300	30	1229	1190.2	28.4	177.92

($\sigma = 0$) and runs considerably faster (0.03s vs. 0.29s). In instance 6, the proposed method reports a slightly higher MRL (414 vs. 348), but again exhibits better balance and faster computation time (0.04s vs. 0.29s).

Beyond these small-scale cases, the proposed ANLS algorithm scales effectively to larger problem instances (e.g., instances 1–11), with reasonable execution times even for the largest configurations (e.g., 500 passengers and 500 parcels in instance 9). In contrast, the baseline method fails to generate feasible solutions for these larger instances, underscoring the practicality and robustness of the proposed approach in real-world scenarios.

3) Performance Efficiency Analysis: To evaluate the convergence behavior and runtime characteristics of the proposed method, I recorded several performance metrics over the course of iterations. Figure 4 illustrates the progression of the maximum route length (MRL), average route length (ARL), standard deviation (σ), and memory usage throughout the optimization process.

Instance 5 serves as a representative case for evaluating the algorithm on small-scale problems, specifically with fewer than 100 destinations.

As illustrated in Figure 4, the maximum route cost exhibits considerable oscillations throughout the optimization process. Although the algorithm occasionally identifies high-quality solutions, it struggles to maintain them consistently, reflecting limited convergence stability. The standard deviation of route costs also fluctuates significantly, ranging from near-zero to over 40, which indicates frequent imbalance across routes and inconsistent workload distribution among vehicles.

During the initial iterations, the average route cost shows a downward trend, suggesting effective exploration of favorable neighborhoods. However, this trend stagnates in later stages, which may be attributed to the algorithm's insufficient diversification or premature convergence to local optima. In contrast, memory usage remains largely stable throughout the entire process, demonstrating the algorithm's efficiency in managing computational resources.

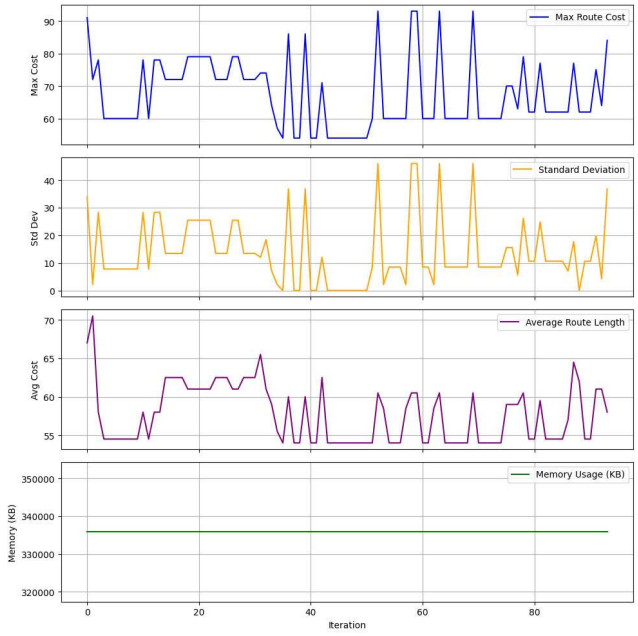

Fig. 4. Convergence trends of performance metrics over iterations for Instance 5.

These observations highlight both the strengths and limitations of the proposed ALNS approach in small-scale instances, particularly its potential in reducing overall cost and the need for enhanced strategies to improve route balance and convergence robustness.

Instance 10 represents a large-scale scenario involving significantly more destinations and request pairs compared to smaller samples. The performance metrics over 16 iterations are visualized in Figure 5, capturing the convergence of the solution with respect to cost, route balance, and computational resources.

The **maximum route cost**, which serves as the primary objective, shows a clear downward trend, decreasing rapidly during the first few iterations and continuing to improve

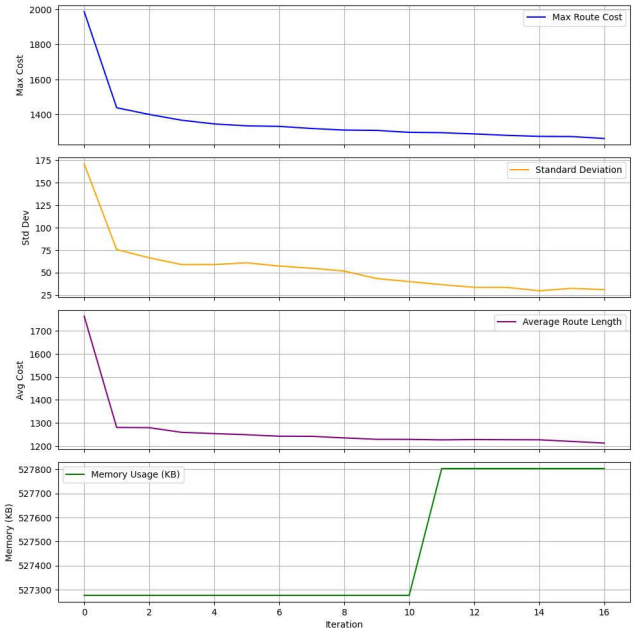

Fig. 5. Convergence behavior of ALNS over iterations for Instance 10 (large-scale instance).

more gradually thereafter. The initial cost exceeds 1950 units and is reduced to approximately 1290 by the final iteration, reflecting an overall improvement of more than 33%. This steep early descent suggests that the ALNS framework is highly effective in exploiting suboptimal initial solutions via aggressive removal-insertion strategies.

The **standard deviation of route costs** also drops substantially, from over 160 to around 30 units, indicating a progressive reduction in the disparity between route lengths. This behavior suggests that the algorithm not only minimizes the maximum route cost but also contributes to more balanced workload distribution across vehicles. The stabilization of this metric around iteration 10 implies that the routes have become relatively uniform in cost, potentially limiting further improvements without disrupting balance.

The **average route cost** follows a similar convergence pattern. After a sharp drop in the first two iterations, it gradually decreases, reaching a plateau around iteration 13. The early-phase improvement corresponds to global restructuring of the solution space, whereas the later flattening hints at diminishing returns, likely due to fewer high-impact request redistributions being available.

Memory usage remains highly stable up to iteration 10, after which a slight but consistent increase is observed. This is attributed to the caching of route evaluations and neighborhood exploration as the search intensifies. Despite this, the overall memory footprint remains under 528,000 KB, demonstrating the algorithm's scalability and controlled resource consumption even for larger instances.

In summary, the results from Instance 10 highlight the robustness of the proposed ALNS framework in large-scale settings. It not only achieves significant reductions in routing costs but also promotes route balance while maintaining computational efficiency. However, the convergence trajectory indicates that beyond a certain point, further improvements become marginal, underscoring the importance of adaptive operator selection and potential hybridization with local search to escape shallow local minima in the later stages of search.

4) Discussion: The convergence results in terms of total cost, standard deviation, and average route length provide strong empirical evidence that the proposed ALNS-based framework is well-suited for addressing the Min-Max Vehicle Routing Problem (Min-Max VRP). In the early iterations, the algorithm achieves rapid improvements due to the high aggressiveness of the removal operators, which facilitate exploration of diverse and promising neighborhoods. This phase is particularly effective when coupled with a simulated annealing acceptance criterion, allowing occasional acceptance of worse solutions to escape local optima.

However, as the algorithm transitions to refinement phases, route clustering into smaller subproblems may lead to premature convergence. In particular, the subproblem decomposition strategy, while computationally efficient, can restrict neighborhood exploration by isolating interdependent route segments. This often results in the solution becoming trapped in a local minimum. To mitigate this, the algorithm incorporates targeted optimization of the longest and shortest routes—thereby reintroducing global diversity into the search process.

For larger instances, especially those with over 200 destinations per vehicle, the framework faces increased difficulty. In such cases, the insertion operations become more computationally expensive due to the combinatorial explosion of feasible insertion positions in long routes. Moreover, maintaining feasibility (e.g., precedence and capacity constraints) in such large solutions poses significant challenges for both greedy and regret-based insertion heuristics.

Finally, the observed increase in memory usage during later iterations can be attributed to the accumulation of route evaluation cache and auxiliary data structures used to track subproblem partitions. While this behavior is expected in a memory-time tradeoff setting, it may become a limiting factor in extremely large-scale scenarios. Strategies such as dynamic cache pruning or memory-aware operator selection could be explored in future work to address this concern.

VI. Conclusion

This mini-project presents an Adaptive Large Neighborhood Search (ALNS) framework for solving the Min-Max Vehicle Routing Problem with mixed requests, including both passenger and parcel transportation. The objective is to minimize the longest individual route among all vehicles, ensuring workload balance while satisfying precedence and capacity constraints. The proposed method integrates a diverse set of removal and insertion operators, guided by an adaptive selection mechanism and a simulated annealing-based acceptance criterion, combining with an optimization process for large-scale instances.

Experimental results demonstrate that the algorithm is capable of significantly reducing the maximum route cost while improving route balance. In particular, the ALNS framework effectively exploits neighborhood structures in the early search phase and applies local refinements via subproblem clustering in later iterations.

The convergence trends across instances reveal that the framework achieves rapid improvements in both objective value and route uniformity. However, performance tends to plateau when the number of destinations per vehicle becomes large, primarily due to the complexity of insertion heuristics over long routes. Additionally, memory usage grows moderately during later iterations due to increased caching overhead, though it remains within acceptable bounds.

Overall, the results confirm that the ALNS framework is a viable and efficient approach for the Min-Max VRP with mixed transport tasks. Future work may explore enhanced operator learning mechanisms, hybridization with local search methods, and adaptation for real-time or dynamic routing environments.

REFERENCES

[1] S. Røpke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006.

[2] L. Chentli, M. Mellouli, and M. Bouzidi, "Selective ALNS for profitable tour problems with simultaneous pickup and delivery," in *Proc. Int. Conf. on Advanced Systems and Electric Technologies (IC_ASET)*, Hammamet, Tunisia, Mar. 2018, pp. 356–361.

[3] C. Li, Y. Mi, and A. Lim, "Adaptive large neighborhood search for the crowd-shipping problem with time windows," *Computers & Operations Research*, vol. 105, pp. 30–42, 2019.