Adaptive Large Neighborhood Search for Order Dispatching and Vacant Vehicle Rebalancing in First-Mile Ride-Sharing Services

Ye, Jinwen; Pantuso, Giovanni; Pisinger, David

Publication date:
2023

Document version
Publisher's PDF, also known as Version of record

Document license:
Unspecified

Citation for published version (APA):
Adaptive Large Neighborhood Search for Order Dispatching and Vacant Vehicle Rebalancing in First-Mile Ride-Sharing Services

Jinwen Ye\textsuperscript{a,}\textsuperscript{*}, Giovanni Pantuso\textsuperscript{a}, David Pisinger\textsuperscript{b}

\textsuperscript{a}University of Copenhagen
\textsuperscript{b}Technical University of Denmark

July 7, 2023

Abstract

This article addresses the first-mile ride-sharing problem, which entails efficiently transporting passengers from a set of origins to a shared destination. Typical destinations are stations, central business districts, or hospitals. Successful optimization of this problem has the potential to alleviate congestion, reduce pollution, and enhance the overall efficiency of transportation systems. However, the inherent complexity of simultaneous order dispatching and vacant vehicle rebalancing often leads to time-consuming computations. In this study, we present an extension of the Adaptive Large Neighborhood Search (ALNS) meta-heuristic, specifically designed to tackle this problem. Through computational experiments on a diverse set of instances, we demonstrate that the proposed ALNS approach delivers high quality solutions within a short timeframe, outperforming off-the-shelf MILP solvers. Furthermore, we conduct a comprehensive case study using simulation, where we show that significant service rate improvements can be achieved by means of rebalancing activities.

Keywords: Adaptive Large Neighborhood Search, Ride-sharing, First-mile, Rebalancing, Meta-heuristic

1 Introduction

Ride-sharing services have emerged as a potential solution to the increase in road congestion and air pollution generated by growing urban areas and population (Taniguchi et al., 2014). Ride-sharing services are transportation solutions that arrange on-demand transport of passengers via shared trips. Of particular interest in this study are first-mile ride-sharing (FMRS) services that connect passengers to a common destination such as a station. Hence the prefix first-mile stresses the first portion of a trip. The full potential of ride-sharing services remains predominantly unexplored. As an example, according to the NYC taxicab data (Commission and Limousine, 2020), during January 2020 only 6% of the taxi trips to the Pennsylvania Station, a fairly busy transit station in New York City were shared by multiple passengers. Significant margins for further market penetration exist.

A successful implementation of FMRS services necessitates the ability to effectively respond to transportation demand. This entails primarily matching transportation requests to vehicles and deciding vehicle routes to the common destination. Such routes need to be feasible with operating specifications such as vehicle capacities and passengers latest arrival times. The ultimate goal is to maximize some measure of performance such as profit. This task is complicated by fluctuating demand. Potential geo-temporal mismatches between supply and demand must be prevented. This is typically done via rebalancing activities, i.e., vehicle movements that aim to anticipate demand changes. Throughout this document we refer to this decision problem as the fist-mile ride-sharing problem (FMRSP).

The routing decisions considered in the FMRSP share similarities with classical routing problems, such as the Vehicle Routing Problem (VRP) (Dantzig and Ramser, 1959) and its variants (Bräysy and Gendreau, 2005; Bertsimas et al., 2019; Kumar and Panneerselvam, 2012; Pillac et al., 2013; Lin et al., 2014; Ritzinger et al., 2016; Brackers et al., 2016). A prominent distinction between the FMRSP and
VRP variants lies in the nature of tour design. The VRP entails the creation of tours that culminate in returning to the depot, whereas the FMRSP focuses on constructing open paths from vehicle origins to a shared destination. Moreover, the VRP conventionally assumes vehicles operating from a single depot, although certain variants do consider multiple depots.

Particularly, the FMRSP exhibits notable similarities with the Dial-a-ride Problem (DARP) and the Pick-up-and-delivery Problem (PDP), see Cordeau and Laporte (2003); Ropke and Cordeau (2009); Berbeglia et al. (2010); Ho et al. (2018). The primary objective of DARP is to minimize the cost or time required to transport a set of passengers utilizing a fixed fleet of vehicles. Requests have different pickup and delivery locations, and DARP allows for the possibility of multiple customers being served by a single vehicle. Various DARP variants exist, including scenarios where the aim is to minimize the detour experienced by customers aboard the vehicles (Pfeiffer and Schulz, 2022). The DARP can be considered as a variant of the PDP, where the former primarily focuses on passenger transportation while the latter typically deals with goods transportation (Parragh et al., 2008). Consequently, distinction between DARP and PDP is often to be found in additional constraints or objectives that explicitly account for user (in)convenience, such as time window restrictions and vehicle capacity limitations. The FMRSP can be regarded as a specific instance of the Dial-a-Ride Problem (DARP), wherein passengers are transported to a shared depot or station, while also accommodating service-specific constraints. Particularly, in the FMRSP we address in this paper, we focus on both routing and rebalancing decisions. This entails effectively managing the transportation of customers who have been promised transportations in earlier decision epochs (thus necessitating their transportation), as well as attending to new customers who may be collected in a price-collecting fashion (Balas, 1989).

On-demand ride-sharing problems have attracted considerable attention in light of the rapid advancements in GPS technology and the widespread adoption of smartphones. Notably, commercial enterprises such as Uber and Didi have successfully implemented versions of this service, see, e.g., Xu et al. (2018); Lin et al. (2018). Considerable attention from the research community has been captured as well. Scholars have responded to this trend by devising optimization techniques (Stiglic et al., 2015; Masoud et al., 2017; Masoud and Jayakrishnan, 2017; Alonso-mora et al., 2018; Stiglic et al., 2018; Huang et al., 2014; Wang et al., 2018; Mourad et al., 2019) as well as reinforcement learning methodologies (Xu et al., 2018; Lin et al., 2018; Li et al., 2019; Tang et al., 2019; Qin et al., 2019) to address the associated challenges. While the ride-sharing problem broadly seeks to optimize the sharing of rides among multiple passengers, the FMRSP concentrates on the initial leg of passengers’ transportation journeys.

The FMRSP represents a comparatively nascent problem and, as a consequence, the corresponding literature is relatively sparse. Yu Shen (2018) investigate the integration of a first-mile autonomous vehicle (AV) ride-sharing service into public transportation, aiming to preserve high-demand bus routes while offering shared AVs as an alternative for low-demand routes. The authors evaluate the system’s performance through simulation experiments, examining various fleet sizes, ride-sharing preferences, and dispatching algorithms. The findings demonstrate the potential of the integrated system to enhance service quality and improve the efficient utilization of bus services. However, it should be noted that the article does not present the mathematical model that underlies the planning decisions. Bian and Liu (2019) designed a mechanism for FMRSP and introduces a novel Solution Pooling Approach (SPA) for effectively addressing the challenges posed by large-scale FMRSP. In contrast to SPA, our approach encompasses not only the order dispatching process but also incorporates the rebalancing process, thereby maximizing the utilization efficiency of vehicles. Furthermore, our algorithm exhibits exceptional efficiency, enabling the efficient computation of larger instances within significantly reduced time frames.

This study represents an extension of our prior research (Ye et al., 2022), where we presented a mathematical formulation for simultaneous decision-making concerning order dispatching and vacant vehicle rebalancing. In this study, we present a novel solution algorithm for the problem which is based on and extends the Adaptive Large Neighborhood Search (ALNS) introduced by Pisinger and Røpke (2010). To the best of our knowledge, this is the first paper using ALNS for the first-mile ride-sharing problem that considers order dispatching and vacant vehicle rebalancing optimization simultaneously. After comparing the algorithm with a state-of-the-art MILP solver, we present a comprehensive case study where we employ our algorithm to simulate dispatch and rebalancing decisions in an 8-hour commuting scenario within a 10 × 10 km² area.

The remainder of this paper is organized as follows. Section 2 describes the problem in detail...
We consider the operator of a fleet of vehicles $K := \{1, \ldots, K\}$ concerned with dispatch and relocation decisions in order to ensure a first-mile ride-sharing service. The fleet is homogeneous with capacity $Q$. We assume the operator makes dispatch and relocations decisions periodically, e.g., every 5 or 10 minutes, as a result of the arrival of new transportation requests. We refer to these decision times as “(re)-optimization phases” and we focus on the decision problem arising at the individual re-optimization phase.

At each re-optimization phase, the available customers can be partitioned into sets $N_P := \{1, \ldots, N_P\}$ contains the customers whose transportation request had already been accepted during a previous re-optimization phase, but not yet fulfilled. Thus, we assume acceptance of a transport request is binding. The set $N_C := \{1, \ldots, N_C\}$ contains newly arrived customers whose request may or may not be accepted. For convenience we set $N := N_C \cup N_P$.

All customers travel to a common destination $d$ located at position $o(d)$ (e.g., a transit station) and for each customer $i \in N$, the operator knows the requested arrival time $T_{id}$ and the origin $o(i)$. At the beginning of the re-optimization phase, denoted time $T$, each vehicle $k$ is located at $o(k)$ as a result of ongoing activities or relocation decisions. The vehicle is either idle in its location, or traveling between customers or to the station. In addition, vehicles might initially have customers on board. We denote $V_k$ the number of customers on board of vehicle $k$ at the beginning of the re-optimization phase and $T_k$ the earliest arrival time of all the passengers already on board vehicle $k$.

The operator bears a cost $C$ for each unit of time a vehicle is in movement. We let $T_{ij}$ be the travel time between locations $o(i)$ and $o(j)$ with $i \in K \cup N, j \in N \cup R \cup \{d\}$. The operator collects a revenue $P_i$ when picking up customer $i$, for $i \in N_C$. We assume the revenue for the customers in $N_P$ has already been collected.

For each re-balancing point $r$ we let $E_r$ denote the expected revenue collected for each vehicle relocated to rebalancing center $i \in R$ and $D_j$ an upper bound on the number of vehicles that can be relocated to the rebalancing center. Expected revenues from rebalancing activities are discounted at a rate $\beta$.

Finally, a set $R := \{1, \ldots, R\}$ contains potential rebalancing points in the operating area. Vehicles with no customers on board may be sent to a rebalancing point or stay at their origin location $o(k)$. For each rebalancing point $r$ we let $E_r$ denote the expected revenue collected for each vehicle relocated to rebalancing center $i \in R$ and $D_j$ an upper bound on the number of vehicles that can be relocated to the rebalancing center. Expected revenues from rebalancing activities are discounted at a rate $\beta$.

The decisions made by the operator can be formalized as follows. We let $x_{ij}^k$ take value 1 if vehicle $k$ moves directly between $o(i)$ and $o(j)$, 0 otherwise, for all $i \in \{k\} \cup N, j \in N \cup R \cup \{d\}, k \in K$.

Furthermore, we let $t_{ij}^k$ denote the actual arrival time of vehicle $k$ to the station, for $k \in K$ and $t_i^p$ denote the actual pick-up time of customer $i$, for $i \in N$.

### 2.2 Mathematical Model

Using the notation above we formulate the FMRSP as follows:

$$
\max \sum_{k \in K} \sum_{i \in N_C} \sum_{j \in N \cup \{d\}} P_i x_{ij}^k - \sum_{i \in (k) \cup N} \sum_{j \in N \cup R \cup \{d\}} \sum_{k \in K} CT_{ij} x_{ij}^k + \beta \sum_{i \in R} \sum_{k \in K} x_{ki}^k E_i
$$

\[1a\]
s.t. \[
\sum_{j \in \mathcal{N} \cup \{d\}} \sum_{k \in \mathcal{K}} x_{ij}^k \leq 1 \quad \forall i \in \mathcal{N}_C \quad (1b)
\]

\[
\sum_{j \in \mathcal{N} \cup \{d\}} \sum_{k \in \mathcal{K}} x_{ij}^k = 1 \quad \forall i \in \mathcal{N}_P \quad (1c)
\]

\[
\sum_{i \in \mathcal{N} \cup \mathcal{K}} x_{id}^k \leq 1 \quad \forall k \in \mathcal{K} \quad (1d)
\]

\[
\sum_{i \in \mathcal{N} \cup \mathcal{K}} x_{ij}^k = \sum_{i \in \mathcal{N} \cup \{d\}} x_{ji}^k \quad \forall j \in \mathcal{N}, k \in \mathcal{K} \quad (1e)
\]

\[
\sum_{j \in \mathcal{R} \cup \mathcal{N} \cup \{d\}} x_{kj}^k = \sum_{j \in \mathcal{N} \cup \{k\}} \sum_{i \in \mathcal{R} \cup \{d\}} x_{ji}^k \quad \forall k \in \mathcal{K} \quad (1f)
\]

\[
\sum_{i \in \{k\} \cup \mathcal{N} \cup \mathcal{R}} \sum_{j \in \mathcal{N}} x_{ij}^k + V_k \leq Q \quad \forall k \in \mathcal{K} \quad (1g)
\]

\[
\sum_{k \in \mathcal{K}} x_{kj}^k \leq D_j \quad \forall j \in \mathcal{R} \quad (1h)
\]

\[
V_k \leq Q(1 - \sum_{j \in \mathcal{R}} x_{kj}^k) \quad \forall k \in \mathcal{K} \quad (1i)
\]

\[
V_k \leq Q \sum_{j \in \mathcal{R} \cup \mathcal{N}} x_{kj}^k \quad \forall k \in \mathcal{K} \quad (1j)
\]

\[
t_i^P + T_{ij} \leq t_j^P + T^L(1 - \sum_{k \in \mathcal{K}} x_{ij}^k) \quad \forall i, j \in \mathcal{N} \quad (1k)
\]

\[
T + T_{kj} \leq t_j^P + T^L(1 - x_{kj}^k) \quad \forall j \in \mathcal{N}, k \in \mathcal{K} \quad (1l)
\]

\[
t_i^A \leq T_i^A + T^L(1 - \sum_{j \in \mathcal{N} \cup \{k\}} x_{ji}^k) \quad \forall i \in \mathcal{N}, k \in \mathcal{K} \quad (1m)
\]

\[
t_k^A \leq T_k \quad \forall k \in \mathcal{K} \quad (1n)
\]

\[
t_i^P + T_{jd} x_{jd}^k \leq t_j^A + T^L(1 - x_{jd}^k) \quad \forall j \in \mathcal{N}, k \in \mathcal{K} \quad (1o)
\]

\[
x_{ij}^k \in \{0, 1\} \quad \forall i \in \{k\} \cup \mathcal{N}, j \in \mathcal{N} \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K} \quad (1p)
\]

\[
t_i^A \in \mathbb{R}^+ \quad \forall k \in \mathcal{K} \quad (1q)
\]

\[
t_i^P \in \mathbb{R}^+ \quad \forall i \in \mathcal{N} \quad (1r)
\]

The objective function (1a) represents the profit for the operator. The first term represents the revenue generated by picking up customers. The second term sums the cost born due to vehicle movements. The third term is the discounted expected profit generated by rebalancing movements.

Constraints (1b) and (1c) state that new customer may be picked up at most once and customers already accepted must be picked up exactly once, respectively. Observe, in (1b) and (1c), that after visiting a customer \(i \in \mathcal{N}_C\) or \(i \in \mathcal{N}_P\), the vehicle can only move to another customer \(i \in \mathcal{N}_P \cup \mathcal{N}_C\) or to the station \(d\), but not to a rebalancing center. Constraints (1d) ensure that vehicles travel to the station at most once. Constraints (1e) state that whenever a vehicle arrives at a customer location, it must then move to another customer or to the station. Notice that a vehicle can visit a customer either immediately after another customer or immediately after the vehicle’s origin \(o(k)\). Constraints (1f) state that, if a vehicle departs from its original location \(o(k)\) it must terminate its journey either at the station or at a rebalancing point. Constraints (1g) ensure that the capacity of the vehicles is not exceeded, while constraints (1h) ensure that the total number of vehicles dispatched to a rebalancing center will not exceed the upper bound on the vehicles dispatchable at the rebalancing center.

Constraint (1i) state that only empty vehicles may be dispatched to rebalancing centers. For instance, if vehicle \(k\) is dispatched to one of the rebalancing center, the right-hand-side becomes 0, and the constraints can only be satisfied when \(V_k\) is equal to 0. If vehicle \(k\) is not dispatched to any rebalancing center, the right-hand-side reduces to the capacity of the vehicle, and the constraint holds with any value of \(V_k\). Notice that the movements between customer points \(\mathcal{N}\) and rebalancing points are automatically forbidden by the absence of the corresponding \(x_{ij}^k\) variables. Constraints (1j) state that the vehicles that already have customers on board at the beginning of the period must be
dispatched (i.e., cannot stay idle). If \( V_k \) is strictly positive, the constraint forces the right-hand-side to be strictly positive as well, and thus to dispatch the vehicle.

Constraints (1k) state that if customer \( j \) is picked up by vehicle \( k \) immediately after picking up customer \( i \), then the actual pick up time of customer \( i \) plus the travel time between customer \( i \) and \( j \) must be less or equal to customer \( j \)'s actual pick up time. Here \( T^j_k := \text{max}\{T^j_i\} \) for \( i \in \mathcal{N} \) is an upper bound on the requested arrival time. Similarly, constraints (1l) denote the pick-up time for the first customers in the route. Constraints (1k)-(1l) prevent subtours. Constraints (1m) ensure that the actual arrival time of vehicle \( k \) is earlier than the requested arrival time of any of the customers on board of it. For instance, if customer \( i \) is picked up by vehicle \( k \), the right-hand-side becomes \( T^i_k \) enforcing that the actual arrival time of vehicle \( k \) is before \( T^i_k \). Constraints (1n) ensure that the actual arrival time of vehicle \( k \) is earlier than the earliest requested arrival time \( T_k \) of the passengers on board at the beginning of the re-optimization phase. Constraints (1o) state the relationship between pick-up time and arrival time. For example, if \( j \) is the last customer picked up by vehicle \( k \) before the station \( x^d_{k,d} \) takes value 1, the left-hand-side becomes \( t^j_k + T_{jd} \), and the right-hand-side becomes \( t^k_A \), enforcing that the actual pick-up time of customer \( j \) plus the travel time between customer \( j \) and station be less than or equal to the actual arrival time of vehicle \( k \). If \( j \) is not the last customer picked up by the vehicle \( k \) before arrive at the station, then the left-hand-side becomes \( t^j_k \), the right-hand-side becomes \( t^k_A + T^L \), which always holds. Finally, constraints (1p)-(1r) define the domain of the decision variables.

The size of the formulation is thus \( O(|\mathcal{N}_c|\cdot|\mathcal{K}|\cdot|R|) \). Observe that the FMRSP is NP-hard, as it contains the prize-collecting TSP (Balas, 1989) as a special case.

### 3 Adaptive Large Neighborhood Search

In this section, an Adaptive Large Neighborhood Search (ALNS) is devised to find solutions to the FMRSP. The procedure of the proposed algorithm is depicted in Algorithm 1. The individual elements of the algorithm will be explained in details in the remainder of this section. Throughout the algorithm the value of a solution generated in the ALNS algorithm is calculated using objective value (1a).

The algorithm receives as input integers \text{maxIter} and \text{maxNoImprov}. They indicate the maximum number of iterations in total and without improvements, the algorithm is allowed to run. Further, it receives sets \( \mathcal{O}^D \) and \( \mathcal{O}^R \) of destroy and repair operators, and constants \( r \in [0,1] \), \( \alpha_1, \ldots, \alpha_4 \) used to adjust their chances of being applied.

It starts by initializing counters and parameters. The weights \( w_k \) of the operators are initially homogeneous. Parameters \( \pi_k \) and \( \theta_k \) keep track of the score accumulated by the operators and the number of times they are called. Finally, an initial feasible solution \( x_i \) is generated and is set as the current solution \( x_c \) and current best solution \( x_b \).

While the algorithm is allowed to run, a new solution \( x' \) is derived from \( x_c \) through the application of destroy and repair operators. These operators, respectively, remove and insert either customers or rebalancing centers from/into the existing vehicle routes. The operators to apply are selected using a roulette-wheel with probabilities proportional to the weights of the operators. Once an operator \( k \) is selected, its frequency is updated.

If the objective value of the new solution is better than that of the current solution, the new solution replaces the current solution. Operators \( k' \) and \( k'' \) are thus rewarded with a score increase by \( \alpha_2 \). In this case the algorithm also checks whether the best solution can be updated. In this case \( k' \) and \( k'' \) are rewarded with a score increase by \( \alpha_3 > \alpha_2 \).

If no improvement is obtained, the counter \text{NoImprov} is incremented. In this case, the new solution is accepted based on a simulated annealing acceptance criterion. Operators \( k' \) and \( k'' \) are rewarded with a score increase by \( \alpha_3 \) if the solution is accepted, otherwise by \( \alpha_4 < \alpha_3 \).

Finally, we partition the iterations into contiguous segments of 100 consecutive iterations. At the beginning of each segment, the scores are reset to zero. At the end of each segment, the weights \( w_k \) are updated according to the scores \( \pi_k \) and frequency \( \theta_k \) during the last 100 iterations using the equation

\[
 w_k := w_k(1-r) + r\frac{\pi_k}{\theta_k}
\]

where \( r \in [0,1] \) controls the extent to which the performance of a segment influences the weight. Specifically, values close to zero imply that weights are only marginally affected by scores. Conversely, with values close to one the weights are highly affected by performance.
Algorithm 1: ALNS algorithm

**Input:**
- \(\text{maxIter, maxNoImprov}\) 
  \(\text{// Maximum \# iterations, total and with no improvements}\)
- \(\mathcal{O}^D, \mathcal{O}^R\) 
  \(\text{// Destroy and repair operators}\)
- \(r, \alpha_1, \ldots, \alpha_4\) 
  \(\text{// Constants used in the adjustment of the weights}\)

**Initialize:**
- \(i \leftarrow 0\) 
  \(\text{// Iteration counter}\)
- \(\text{NoImprov} \leftarrow 0\) 
  \(\text{// Counter of the iterations without improvement}\)
- \(w_k \leftarrow 1/|\mathcal{O}^D|\) for all \(k \in \mathcal{O}^D\) 
  \(\text{// Initial weight of the destroy operators}\)
- \(w_k \leftarrow 1/|\mathcal{O}^R|\) for all \(k \in \mathcal{O}^R\) 
  \(\text{// Initial weight of the repair operators}\)
- \(\pi_k \leftarrow 0\) for all \(k \in \mathcal{O}^D \cup \mathcal{O}^R\) 
  \(\text{// Initial score of the operators}\)
- \(\theta_k \leftarrow 0\) for all \(k \in \mathcal{O}^D \cup \mathcal{O}^R\) 
  \(\text{// Counters of the operators}\)
- \(x_i \leftarrow \text{generateInitialSolution}()\) 
  \(\text{// Initial solution}\)
- \(x_c, x_b \leftarrow x_i\) 
  \(\text{// Current and current best solution}\)

**while** \(\text{NoImprov} \leq \text{maxNoImprov}\) and \(i \leq \text{maxIter}\) **do**

- \(i \leftarrow i + 1\)
- \(x' \leftarrow x_c\)
- Select a destroy operator \(k'\) randomly with probabilities \(p_{k'} = \frac{w_{k'}}{\sum_{i=1}^{|\mathcal{O}^D|} w_i}\) and apply it to \(x'\)
- \(\theta_{k'} \leftarrow \theta_{k'} + 1\)
- Select a repair operator \(k''\) randomly with probabilities \(p_{k''} = \frac{w_{k''}}{\sum_{i=1}^{|\mathcal{O}^R|} w_i}\) and apply it to \(x'\)
- \(\theta_{k''} \leftarrow \theta_{k''} + 1\)
- Compute the objective value \(f(x')\) of \(x'\)
- if \(f(x') > f(x_c)\) then
  - \(x_c \leftarrow x'\)
  - \(\pi_{k'} \leftarrow \pi_{k'} + \alpha_2\)
  - \(\pi_{k''} \leftarrow \pi_{k''} + \alpha_2\)
  - \(\text{NoImprov} \leftarrow 0\)
- if \(f(x_c) > f(x_b)\) then
  - \(x_b \leftarrow x_c\)
  - \(\pi_{k'} \leftarrow \pi_{k'} + \alpha_1\)
  - \(\pi_{k''} \leftarrow \pi_{k''} + \alpha_1\)
- else
  - \(\text{NoImprov} \leftarrow \text{NoImprov} + 1\)
  - if \(x'\) is accepted by simulated annealing criteria then
    - \(x_c \leftarrow x'\)
    - \(\pi_{k'} \leftarrow \pi_{k'} + \alpha_3\)
    - \(\pi_{k''} \leftarrow \pi_{k''} + \alpha_3\)
  - else
    - \(\pi_{k'} \leftarrow \pi_{k'} + \alpha_4\)
    - \(\pi_{k''} \leftarrow \pi_{k''} + \alpha_4\)
- end
- end
- if \(i\) modulo 100 = 0 then
  - for \(k \in \mathcal{O}^D \cup \mathcal{O}^R\) **do**
    - if \(\theta_k > 0\) then
      - \(w_k \leftarrow w_k (1 - r) + r \frac{\pi_k}{\theta_k}\)
      - \(\pi_k, \theta_k \leftarrow 0\)
    - end
  - end
- end
3.1 Initial Solution

The construction of the initial solution follows a three-step procedure. The procedure ensures the construction of a feasible solution.

In the first step, mandatory customers are assigned to routes. The assignment is performed as described in Algorithm 2. For each previous (mandatory) customer, the set of vehicles with remaining capacity is ordered randomly. Vehicles are then scanned in the order of the list. Given a vehicle, the procedure looks for the cheapest feasible insertion of the customer (recall that mandatory customers do not contribute to revenue). A feasible insertion is one that satisfies the latest arrival times of all passengers and of the vehicle.

In the second step, new customers are randomly inserted into the routes. The insertion procedure (Algorithm 3) is very similar to that of previous customers, with the only exception that it looks for the first feasible insertion of the customer rather than the best feasible insertion.

In the third and last step, empty vehicles are randomly dispatched to rebalancing centers, ensuring that the capacity of the rebalancing centers is respected.

### Algorithm 2 Allocation of previous customer

**Input:**
\( N_P, K \)

**for** \( c \in N_P \) **do**

- Let \( K^O \) be a randomly ordered list of the vehicles with residual capacity

  **for** \( v \in K^O \) **do**

    - **for** \( i = 1, \ldots, Q_v \) **do**
      - Let \( T \) be the tour obtained by inserting customer \( c \) into position \( i \) of the existing tour of vehicle \( v \) if \( T \) is feasible with respect to arrival times then
      - Let \( C^v_i \) be the cost of \( T \)
      - else
        - \( C^v_i = \infty \)
      - end
    - end
    - if \( \min_{i=1,\ldots,Q_v} \{C^v_i\} < \infty \) then
      - Assign customer \( c \) to position \( i = \arg \min_{i=1,\ldots,Q_v} \{C^v_i\} \) of the tour of vehicle \( v \)
      - Move on to next customer
    - end
  - end
- end

### Algorithm 3 Allocation of new customers

**Input:**
\( N_C, K \)

**for** \( c \in N_C \) **do**

- Let \( K^O \) be a randomly ordered list of the vehicles with residual capacity

  **for** \( v \in K^O \) **do**

    **for** \( i = 1, \ldots, Q_v \) **do**
    - Let \( T \) be the tour obtained by inserting customer \( c \) into position \( i \) of the existing tour of vehicle \( v \) if \( T \) is feasible with respect to arrival times then
      - Assign customer \( c \) to position \( i = \arg \min_{i=1,\ldots,Q_v} \{C^v_i\} \) of the tour of vehicle \( v \)
    - Move on to next customer
  - end
- end

Figure 1 illustrates the construction heuristic in a small fictitious example. We have \( N_P := \{P1\} \),
$N_C := \{C1, C2, C3\}$, $R := \{R1\}$, $K := \{D1, D2, D3\}$. The station is indicated by $S$. The procedure starts by assigning customer $P1$ to $D2$. Following customers $C1$ and $C2$, are assigned to $D1$. Instead, customer $C3$ is not assigned to any vehicle as it is not possible to find a feasible insertion (e.g., the latest arrival time is too tight). Finally, the construction heuristic assigns empty vehicles to rebalancing centers. As a result, vehicle $D3$ is dispatched to rebalancing center $R1$.

![Figure 1: Example construction of a feasible solution.](image)

### 3.2 Solution Acceptance

A simulated annealing criterion is employed to determine whether non-improving solutions are accepted. The probability of accepting a non-improving solution is

$$p = e^{\frac{f(x_c) - f(x')}{T_i}}$$

where $x_c$ is the current solution, $x'$ is the newly generated solution, and $T_i$ is the temperature at iteration $i$. The initial temperature is carefully selected such that a solution $M\%$ worse than the initial solution has a 50% probability of being accepted. At each iteration, the temperature is reduced according to $T := T(1 - i/\text{maxIter})$. During parameter configuration, we conduct numerical experiments to identify the optimal value of $M$.

### 3.3 Removal Operators

Five removal operators are designed to destroy a complete solution into a partial solution. The operators are described as follows:

- **Nearby Location Removal** (NLR). A random customer $c$ is selected from the customers assigned to a vehicle. All new (optional) customers $c'$ situated within 2Km from $c$ are then removed from the respective vehicles.

- **Same Arrival Time Removal** (SATR). A random customer $c$ is selected from the customers assigned to a vehicle. All new (optional) customers with the same requested arrival time as $c$ are removed from the respective routes.

- **Free Part of Routes** (FPR). One-fourth of the vehicles are randomly selected. All the new (optional) customers and rebalancing centers are removed from their routes.

- **Remove Optional** (RO). All new (optional) customers are removed from their respective routes.

- **Destroy Worst** (DW). Vehicles are ranked according to their objective value contribution. All new (optional) customers and rebalancing centers assigned to the worst one-third vehicles are then removed.
3.4 Insertion Operators

Four insertion operators are introduced to reconstruct partial solutions.

- **Random Insertion Customer First** (RICF). The new customers not assigned to a vehicle are randomly inserted into partial routes with residual capacity and that not terminate at rebalancing centers. The insertion is always checked for feasibility. Infeasible insertion are discarded and a new random insertion is evaluated. Remaining empty vehicles are randomly assigned to rebalancing centers.

- **Random Insertion Rebalancing First** (RIRF). Same procedure as the RICF operator, with the exception that rebalancing centers are randomly assigned before customers.

- **Greedy Insertion** (GI). For each vehicle with residual capacity, it goes through all unassigned new customers until it finds new customers to insert in the route in a feasible manner. Finally, any remaining empty vehicle is randomly assigned to a rebalancing center with residual capacity.

- **Best Customer Insertion** (BCI). For each route, it checks the score of inserting each new customer that has not yet been assigned to a vehicle. The customer is inserted into each position of the route and, if the insertion is feasible, the score is recorded. The customer with the highest score is then inserted in the route at the best position.

4 Computational Experiments

In this section we report on a number of experiments that test the performance of the ALNS method developed. Particularly, we compare the performance of the method against that of the MILP solver Gurobi (version 9.5.0). The ALNS algorithm is implemented in Python. The experiments are performed on a set of instances of various sizes. In addition, to assess the practical relevance of the model and solution method developed, we simulate its usage on a 10 \times 10 \text{km}^2 for an eight-hour operating period. All instances are accessible at https://github.com/JWYOpt/InstancesALNS_FMRSP. All experiments are performed on compute nodes with 40 cores and 171GB memory.

4.1 Test instances

The experiments are conducted on four distinct instance classes, each encompassing three individual randomly generated instances. Particularly we construct instance classes denoted as $V|K|N_C|N_P|R$, where $K$ represents the number of vehicles, $N_C$ represents the number of new customers, $N_P$ represents the number of previous customers, and $R$ represents the number of rebalancing centers. As an illustration, the instance class V50-C150-P45-R3 indicates instances with 50 vehicles, 150 new customers, 45 previous customers, and 3 rebalancing centers. For every instance class, we randomly generate three distinct instances.

Instances are generated assuming a 10 \times 10 \text{km}^2 area, see Figure 2. The station is located at point (0,0). The locations of the vehicles are randomly drawn in the entire area. The locations of new customers are randomly drawn following the distribution illustrated in Figure 2. Particularly, 10% of the new customers are randomly drawn in the square delimited by the points (0,0), (5,0), (0,5) and (5,5), additional 10% in the square delimited by the points (5,0), (5,5), (10,0) and (10,5), additional 30% in the square delimited by the points (0,5), (5,5), (0,10) and (5,10), and the remaining 50% in the remaining area. Previous customers are randomly generated across the entire area.

We assume a fleet of homogeneous vehicles of capacity $Q = 4$ and that each vehicle is initially empty $V_k = 0$, for all $k \in K$. The requested arrival time for new customers is randomly selected from the set (30, 40, 50), while the requested arrival time for previous customers is randomly chosen from the set (20, 30, 40). The distance between locations is calculated using the Euclidean distance. The traveling speed is set to at 36 Km/h. The unit transportation cost $C$ is set to $11.25$/h. Furthermore, trip revenue $P_i$ of a new customer $i$ is computed using a fare of $2.59$ per Km (using the Euclidean distance between the customer and the station) plus $0.74$ per minute (again using the Euclidean distance) from the picking up point of customer $i$ to the station (destination), ensuring a minimum fare of $8$. The discount factor for the rebalancing benefit is set to $\beta = 0.1$. 


4.2 ALNS parameters

An implementation of the ALNS method requires setting a number of parameters. In this section we describe how such parameters are chosen.

The scores used to reward the performance of the operators are set as \((\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (33, 9, 13, 0)\) following Ropke et al. (2006). The initial temperature \(T_0\) is chosen in order to give 50% probability of accepting solutions that are \(M\%\) worse than the initial solution. In order to tune the value of \(T_0\) we tested the algorithm under different values of \(M\). Particularly, we tested \(M \in \{20, 50, 80\}\) on instance of classes V50-C150-P45-R3, V100-C300-P50-R3, V150-C450-P75-R3, and V200-C600-P100-R3. For each class, three random instances were generated. The individual instances are indicated by appending their number at the end of the class name. As an example V50-C150-P45-R3-1 indicates instance number 1 of class V50-C150-P45-R3.

The results of the experiments are presented in Table 1. The results show that the average objective value achieved with \(M = 50\) is higher than with the other values of \(M\). This is valid for all partial averages, thus across all instance sizes tested. Hence, we adopt the value of \(M = 50\) in the remainder of the experiments presented.

4.3 Performance of the algorithm

In this section, we report on the performance of the algorithm. Particularly, we test the algorithm using six instance classes, each containing three random instances.

Figure 3 illustrates the progression of the best objective value and current objective value during the ALNS solution process. This is done on three instances, each of a different class. The horizontal axis reported the time. It can be noted that during the initial phases, the algorithm tends to accept considerably worse solutions. However, the acceptance rate decreases significantly as the temperature decreases and the quality of the solutions found increases. It can be further be noticed that the ALNS is able to significantly improve the initial solution. This often happens in significant jumps after the initial phases where many worse solutions were evaluated. It appears that the initial exploration phase is followed by an exploitation phase in more promising regions of the solution space.

Following, we compare the optimality gap and solution time of the ALNS to those obtained by the solver Gurobi. The experiments are meant to replicate a realistic scenario where solutions are required within rather short times. For this reason, both the ALNS and Gurobi are allowed to run for at most 300 seconds. The optimality gap of the solutions delivered by ALNS is calculated using the upper bound delivered by Gurobi (when available). Table 2 reports the results for the six instance classes tested.

It can be noted that, for the smallest instances (namely V20-C40-P10-R3 and V30-C60-P15-R3), both ALNS and Gurobi deliver rather small optimality gaps. The optimality gaps of the ALNS solutions are slightly larger than those of the solutions delivered by Gurobi on the V20-C40-P10-R3
instances and comparable on the V30-C60-P15-R3 instances. However, ALNS delivers such solutions within a few seconds, while Gurobi uses the 300-second period entirely.

For larger instances, such as V40-C80-P30-R3 and V50-C100-P45-R3, the performance of Gurobi falls dramatically. The solutions delivered by the solver within 300 seconds exhibit an extremely large optimality gap. In some instances the solver is not to deliver feasible solutions. In contrast, the ALNS algorithm delivers feasible solutions with an average optimality gap of 10% within 20 seconds.

Finally, for the largest instances Gurobi fails to deliver upper and lower bounds. Consequently, the optimality gaps of the solutions delivered by ALNS could not be computed. Nonetheless, ALNS reliably delivers feasible solutions for all instances, in many cases much earlier than the allowed 300 seconds.

### 5 Case study

We present now a case study where we simulate the process of order dispatching and rebalancing in a typical operational day. The purpose is to assess the solution obtained in two operating scenarios, namely with rebalancing (WR) and without rebalancing (WOR). Intuitively, in the WR scenario we allow empty vehicles to move to rebalancing centers, while in the WOR scenario we focus solely on order dispatching.

The simulation framework replicates an 8-hour business day, e.g., 9:00-17:00. Re-optimizations are performed at regular intervals of 5 minutes throughout the day. The ALNS algorithm is used to solve re-optimization problems. At the beginning, no previous customer exist. The first re-optimization phase concerns only new customers and, in the WR scenario, rebalancing centers. However, as new customers are allocated to vehicles during the re-optimizations, they might materialize as previous customers in later re-optimizations. Particularly, this happens any time a new customer is allocated to a vehicle but not picked-up within the 5-minute interval between re-optimizations. If they are picked-up within the 5-minute interval, but not delivered, they appear in the occupied seats $V_k$.

The demand distribution, vehicle capacity, and number of rebalancing centers for each re-optimization period are as described in Section 4.1. In the simulation, a fleet of 40 vehicles is employed. We consider three distinct sizes for the number of new customers gathered during each re-optimization period,
namely 70, 80, and 90. The requested arrival time for the new customers is randomly selected from the set \{20, 30, 40\}. Additionally, if these arrivals become previous customers for subsequent re-optimizations, their requested arrival time is adjusted by deducting the 5 minutes elapsed between re-optimization. In summary, we conduct experiments on three different instance sizes denoted as $V^{40} - C |N|$, where $|N|$ signifies the number of new customers generated during each re-optimization period. In the WR scenario we use three rebalancing centers. For each instance size, we generate three random instances.

Table 3 reports the total number of customers picked up and the service rate during the 8-hour period. The service rate is calculated as the ratio between the number of customers picked up and the total number of customers appeared in the simulation. It can be observe that, when 70 new customers are generated in each re-optimization, the service rate of WR is only marginally better than that of WOR. This signals that, in this case, the fleet is large enough to serve the majority of the customers, even without anticipating demand. Conversely, for larger instances, performing rebalancing yields significant service rates improvements. On average, the service service rate of the WR scenario is higher than that of the WOR scenario by approximately 10 percentage points.

6 Conclusions

The article introduced an ALNS to find high quality solution to the problem of dispatching and rebalancing vehicles in first-mile ride-sharing services. Our computational experiments demonstrate the superiority of the proposed algorithm to commercial solvers, particularly for medium and large problem instances. The method consistently and quickly delivers good feasible solutions within short computation times even for instances where the solver fails to find even an upper bound. In addition, a simulation experiment reveals that the service rate obtained by means of rebalancing activities is consistently higher than that obtained without rebalancing.

Acknowledgement

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 801199.

Declaration of generative AI and AI-assisted technologies in the writing process
During the preparation of this work the author(s) used ChatGPT in order to improve the language and readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.
Table 2: Comparison of ALNS and Gurobi. Optimality gaps are computed as \((\text{bestUpperBound-objective})/(\text{bestUpperBound}+\epsilon)\), where \text{bestUpperBound} is the best upper bound delivered by Gurobi (when available) and \(\epsilon\) is a small constant. A dash – indicates that no feasible solution was found. \(\text{NC}\) (Non-computable) indicates that a feasible solution was found, but the gap could not be calculated as Gurobi failed to provide an upper bound.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Gap [%]</th>
<th>ALNS Runtime [s]</th>
<th>ALNS Gap [%]</th>
<th>Gurobi Runtime [s]</th>
<th>Gurobi Gap [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>V20-C40-P10-R3-1</td>
<td>3.11%</td>
<td>3.88</td>
<td>0.93%</td>
<td>300.21</td>
<td></td>
</tr>
<tr>
<td>V20-C40-P10-R3-2</td>
<td>3.40%</td>
<td>3.30</td>
<td>1.00%</td>
<td>300.21</td>
<td></td>
</tr>
<tr>
<td>V20-C40-P10-R3-3</td>
<td>1.94%</td>
<td>6.09</td>
<td>1.06%</td>
<td>300.58</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>2.81%</td>
<td>4.43</td>
<td>0.99%</td>
<td>300.33</td>
<td></td>
</tr>
<tr>
<td>V30-C60-P15-R3-1</td>
<td>7.44%</td>
<td>6.77</td>
<td>6.49%</td>
<td>304.60</td>
<td></td>
</tr>
<tr>
<td>V30-C60-P15-R3-2</td>
<td>8.08%</td>
<td>8.04</td>
<td>7.31%</td>
<td>304.55</td>
<td></td>
</tr>
<tr>
<td>V30-C60-P15-R3-3</td>
<td>6.83%</td>
<td>12.79</td>
<td>6.12%</td>
<td>307.79</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>7.45%</td>
<td>9.20</td>
<td>6.64%</td>
<td>305.65</td>
<td></td>
</tr>
<tr>
<td>V40-C80-P30-R3-1</td>
<td>9.76%</td>
<td>8.47</td>
<td>147.34%</td>
<td>300.11</td>
<td></td>
</tr>
<tr>
<td>V40-C80-P30-R3-2</td>
<td>10.31%</td>
<td>7.02</td>
<td>352.47%</td>
<td>300.52</td>
<td></td>
</tr>
<tr>
<td>V40-C80-P30-R3-3</td>
<td>9.34%</td>
<td>11.30</td>
<td>3349.01%</td>
<td>304.16</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>9.80%</td>
<td>8.93</td>
<td>1282.94%</td>
<td>301.60</td>
<td></td>
</tr>
<tr>
<td>V50-C100-P45-R3-1</td>
<td>9.89%</td>
<td>17.50</td>
<td>2189.29%</td>
<td>300.21</td>
<td></td>
</tr>
<tr>
<td>V50-C100-P45-R3-2</td>
<td>9.72%</td>
<td>22.73</td>
<td>-</td>
<td>300.03</td>
<td></td>
</tr>
<tr>
<td>V50-C100-P45-R3-3</td>
<td>10.60%</td>
<td>18.14</td>
<td>-</td>
<td>300.62</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>10.07%</td>
<td>19.46</td>
<td>2189.29%</td>
<td>300.29</td>
<td></td>
</tr>
<tr>
<td>V100-C200-P50-R3-1</td>
<td>NC</td>
<td>92.12</td>
<td>-</td>
<td>361.65</td>
<td></td>
</tr>
<tr>
<td>V100-C200-P50-R3-2</td>
<td>NC</td>
<td>109.79</td>
<td>-</td>
<td>353.21</td>
<td></td>
</tr>
<tr>
<td>V100-C200-P50-R3-3</td>
<td>NC</td>
<td>86.72</td>
<td>-</td>
<td>363.87</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>NC</td>
<td>96.21</td>
<td>-</td>
<td>359.57</td>
<td></td>
</tr>
<tr>
<td>V150-C300-P75-R3-1</td>
<td>NC</td>
<td>251.04</td>
<td>-</td>
<td>300.96</td>
<td></td>
</tr>
<tr>
<td>V150-C300-P75-R3-2</td>
<td>NC</td>
<td>216.79</td>
<td>-</td>
<td>301.04</td>
<td></td>
</tr>
<tr>
<td>V150-C300-P75-R3-3</td>
<td>NC</td>
<td>221.92</td>
<td>-</td>
<td>300.99</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>NC</td>
<td>229.92</td>
<td>-</td>
<td>300.99</td>
<td></td>
</tr>
</tbody>
</table>

References


Table 3: Results of the simulation of an 8-hour period with re-optimizations every 5 minutes. The service rate is computed as the number of customers picked up over the total number of customers appeared in the entire simulation.

<table>
<thead>
<tr>
<th></th>
<th>WR Picked Up</th>
<th>Service Rate</th>
<th>WOR Picked Up</th>
<th>Service Rate</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V40-C70-1</td>
<td>5151</td>
<td>89.43%</td>
<td>5141</td>
<td>89.25%</td>
<td>5760</td>
</tr>
<tr>
<td>V40-C70-2</td>
<td>5328</td>
<td>92.50%</td>
<td>5154</td>
<td>89.48%</td>
<td>5760</td>
</tr>
<tr>
<td>V40-C70-3</td>
<td>5304</td>
<td>92.08%</td>
<td>5245</td>
<td>91.06%</td>
<td>5760</td>
</tr>
<tr>
<td>Average</td>
<td>5261</td>
<td>91.34%</td>
<td>5180</td>
<td>89.93%</td>
<td>5760</td>
</tr>
<tr>
<td>V40-C80-1</td>
<td>5759</td>
<td>74.99%</td>
<td>4396</td>
<td>57.24%</td>
<td>7680</td>
</tr>
<tr>
<td>V40-C80-2</td>
<td>5832</td>
<td>75.94%</td>
<td>4803</td>
<td>62.54%</td>
<td>7680</td>
</tr>
<tr>
<td>V40-C80-3</td>
<td>5887</td>
<td>76.65%</td>
<td>4192</td>
<td>54.58%</td>
<td>7680</td>
</tr>
<tr>
<td>Average</td>
<td>5826</td>
<td>75.86%</td>
<td>4463.667</td>
<td>58.12%</td>
<td>7680</td>
</tr>
<tr>
<td>V40-C90-1</td>
<td>5632</td>
<td>65.19%</td>
<td>5468</td>
<td>63.29%</td>
<td>8640</td>
</tr>
<tr>
<td>V40-C90-2</td>
<td>5875</td>
<td>68.00%</td>
<td>5140</td>
<td>59.49%</td>
<td>8640</td>
</tr>
<tr>
<td>V40-C90-3</td>
<td>5972</td>
<td>69.12%</td>
<td>4030</td>
<td>46.64%</td>
<td>8640</td>
</tr>
<tr>
<td>Average</td>
<td>5826.333</td>
<td>67.43%</td>
<td>4879.333</td>
<td>56.47%</td>
<td>8640</td>
</tr>
<tr>
<td>Total Average</td>
<td>5637.778</td>
<td>78.21%</td>
<td>4841</td>
<td>68.17%</td>
<td>7360</td>
</tr>
</tbody>
</table>


