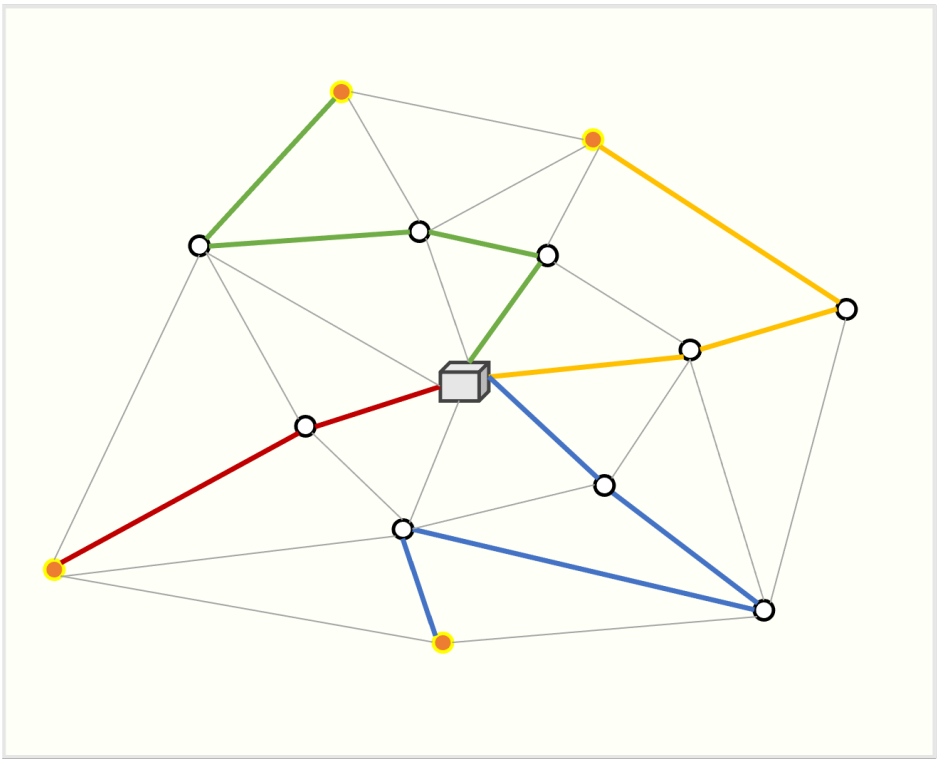
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



PhD thesis

Optimization in First-mile Ride-sharing Problems

Jinwen Ye

Advisor: Giovanni Pantuso

Submitted: August 31, 2023

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen

PhD Thesis by:

Jinwen Ye
Department of Mathematical Sciences
University of Copenhagen
DK-2100 Copenhagen Ø
j.y@math.ku.dk

Assessment committee:

Trine Krogh Boomsma (chair)
Associate Professor, University of Copenhagen

Paolo Toth
Emeritus Professor, University of Bologna

Jose Fernando Oliveira
Professor, University of Porto

Supervisor:

Giovanni Pantuso
Associate Professor, University of Copenhagen

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen on August 31st, 2023. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 801199.

ISBN: 978-87-7125-222-4.

Abstract

This thesis presents mathematical models and methods for supporting the decision making process in first-mile ride-sharing services and is structured into four chapters. The first chapter is an introduction to the background of First-mile Ride-sharing Problems and the optimization methodologies. It is followed by three self-contained chapters.

The second chapter ***Fleet Size Control in First-mile Ride-sharing Problem*** considers the fleet size optimization in *first-mile ride-sharing problem*. The problem is formulated as a MIP model and numerical experiments are presented on a small-scale system. Results show that, by employing and controlling the drivers, the operator can achieve higher profits and service rates compared to hiring independent drivers.

The third chapter ***Order Dispatching and Vacant Vehicles Rebalancing for the First-mile Ride-sharing Problem*** formulates a mathematical programming model to optimize the vehicle rebalancing and order matching process simultaneously. Particularly, we propose and compare two methods to identify rebalancing centers based on historical data. Numerical experiments are conducted in a rolling horizon framework on small-scale instances, and the result demonstrate the consistent effectiveness of rebalancing.

The fourth chapter ***Adaptive Large Neighborhood Search for Order Dispatching and Vacant Vehicle Rebalancing in First-Mile Ride-Sharing Services*** further discusses the rebalancing in first-mile ride-sharing problems for large scale cases. To solve large scale instances, we specifically design and implement an extension of the *Adaptive Large Neighborhood Search* (ALNS) meta-heuristic. Tailor-made destroy and repair operators are designed and tested through computational experiments. Results on a diverse set of instances show that our proposed ALNS outperform commercial solvers for small-scale instances as well as large-scale instances, as ALNS delivers high quality solutions in short timeframe in all scenarios. Furthermore, our case study demonstrates the effectiveness of rebalancing in increasing service rates.

Resumé

Denne afhandling præsenterer matematiske modeller og metoder med formål at understøtte beslutningsprocessen i first-mile ride-sharing services og er struktureret i fire kapitler. Det første kapitel er en introduktion til baggrunden for First-mile Ride-sharing-problemer og optimeringsmetoderne. Den efterfølges af tre selvstændige kapitler.

Det andet kapitel ***Fleet Size Control in First-mile Ride-sharing Problem*** omhandler flådestør-relsesoptimering i *first-mile ride-sharing problems*. Problemstillingen er formuleret som en MIP-model og numeriske eksperimenter præsenteres via små-skala eksempler. Resultatet viser, at ved at ansætte og styre chafførerne, kan operatøren opnå højere profitter og servicarater end uafhængige chaffører.

Det tredje kapitel ***Order Dispatching and Vacant Vehicles Rebalancing for the First-mile Ride-sharing Problem*** formulerer en matematisk programmeringsmodel til at samtidigt optimere køretøjsombalanceringen og ordrematchningsprocessen. Specifikt foreslår og sammenligner vi to metoder til at lokalisere ombalanceringscentre baseret på historisk data. Numeriske eksperimenter bliver udført i en rullende horisontramme på små testsæt, og resultatet viser den konsistente effektivitet af rebalancering.

Det fjerde kapitel ***Adaptive Large Neighborhood Search for Order Dispatching and Vacant Vehicle Rebalancering in First-Mile Ride-Sharing Services*** diskuterer yderli-gere rebalanceringen i first-mile ride-sharing-problemer i store testsæt. Til at løse eksperimenter i stor skala har vi designet og implementeret en udvidelse af *Adaptive Large Neighborhood Search* (ALNS) metaheuristikken. Specialdesignede ødelægge- og reparationsoperatører testes igennem beregningseksperimenter. Resultaterne fra en række af forskellige testsæt viser, at vores foreslåede ALNS overgår kommercielle løsere i små testsæt såvel som store testsæt, da ALNS leverer løsninger af høj kvalitet på kort tid i alle scenarier. Ydermere, viser vores casestudie effektiviteten af rebalancering i forøgelsen af servicerater.

Preface

This thesis has been prepared in fulfillment of the requirements for the PhD degree at the Department of Mathematical Sciences, Faculty of Science, University of Copenhagen. The work has been written between September 2020 and August 2023, under the supervision of my supervisor Associate Professor Giovanni Pantuso (University of Copenhagen) and co-supervisor Professor David Pisinger (Technical University of Denmark).

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.

Acknowledgments

I want to thank my supervisor Giovanni Pantuso for the guidance and support through my entire PhD project. I enjoyed every discussion we had during these three years. And whenever I had any struggles with the project, I always felt inspired and encouraged after our meeting. And many thanks to my co-supervisor David Pisinger for always willing to support me and give me inspirations and guidance for my project.

Furthermore, I would like to thank Nikolas Geroliminis for welcoming and supervising me at École Polytechnique Fédérale de Lausanne (EPFL) from February to April 2023. I enjoyed the discussion of new research topic and the beautiful views in Switzerland.

I want to thank everybody at the Department of Mathematical Sciences for various ways you have encouraged or supported me during this project. In particular, I would like to thank JiaLi for academic discussions and the time spending together to cheer each other up during the last year of my project.
I want to thank my family and friends for always sending their understanding and supporting to me from different time zones around the world.

Finally, I want to thank my boyfriend Jakob for always being there for me. Thanks for your care, love and support that always accompany me during the struggling moments and for always making me feel happy about life, regardless of the situation.

Jinwen
August 2023

List Of Papers

Chapter 2: Ye, J., Pantuso, G., Pisinger, D. (2022). Fleet Size Control in First-Mile Ride-Sharing Problems. In: de Armas, J., Ramalhinho, H., Voß, S. (eds) Computational Logistics. ICCL 2022. Lecture Notes in Computer Science, vol 13557. Springer, Cham. https://doi.org/10.1007/978-3-031-16579-5_7

Chapter 3[1]**:** Ye, Jinwen and Pantuso, Giovanni and Pisinger, David, Online Order Dispatching and Vacant Vehicles Rebalancing for the First-Mile Ride-Sharing Problem. *Under revision in EURO Journal on Transportation and Logistics.* Preprint available at: http://dx.doi.org/10.2139/ssrn.4265368

Chapter 4: Ye, Jinwen and Pantuso, Giovanni and Pisinger, David, Adaptive Large Neighborhood Search for Order Dispatching and Vacant Vehicle Rebalancing in First-Mile Ride-Sharing Services. *Under review in Computers and Operations Research.* Preprint available at: http://dx.doi.org/10.2139/ssrn.4517039

[1]The version in Chapter Chapter 3 has been updated compared to the preprint available online.

Contents

Chapter 1

Introduction

1.1 Background

The three chapters that follow, deal with the First-Mile Ride-Sharing Problem (FMRSP). The FMRSP consists of matching vehicles to passengers that need to reach a common destination, such as a transit station (hence the first mile of their journey). The decision process can be illustrated as in Figure 1.1. On the left of this figure, the service provider received 9 customer requests with the same destination (i.e. station). The number of vehicles which are available in the system is 4. The grey lines are the potential routes can be generated for vehicles to traverse. A possible solution is shown in the figure to the right. The four routes for vehicles are colored in green, red, yellow and blue, and all customers are assigned to vehicles.

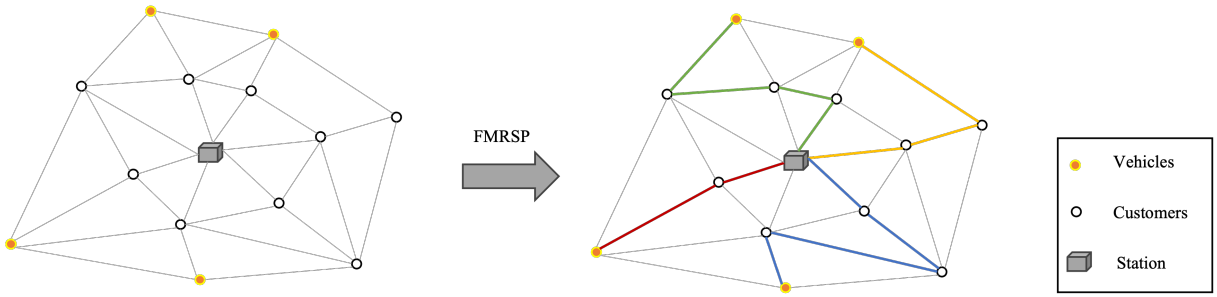


Figure 1.1: FMRSP Decision Process

First-mile ride-sharing has become a valuable concept in transportation systems as it addresses several challenges in urban transportation. First, it provides public transportation accessibility for people that live in suburban areas or at a considerable distance from the transportation hub. Second, ride-sharing has the potential to reduce transportation costs, as the cost of one vehicle can be distributed among multiple customers. Third, ride-sharing has the potential to alleviate congestion and emissions by reducing the number of vehicles on the road.

The study of routing decisions in FMRSP can trace back to the classic Vehicle Routing Problem (VRP) (Dantzig and Ramser, 1959), which considered the routing decision for efficiently dispatching a fleet of vehicles from the depot, while satisfying the goods demand of a set of customer

locations. Variants of the VRP have been explored and several exact and heuristics algorithms have been developed for studying these problems, see e.g., the surveys (Toth and Vigo, 2002; Ritzinger et al., 2016; Braekers et al., 2016). The major difference between variants of the VRP and the FMRSP is that the VRP only considers the trips starting from and returning to the depot, while FMRSP designs the vehicle routes from different origins to the same destination. Given the potential for demand imbalances during peak hours in the context of first-mile on-demand transportation, in the FMRSP it is also important to reason in terms of rebalancing and anticipating demand, compared to VRPs. Furthermore, in classical VRPs, vehicles are loaded at the depot and the goods are delivered to customers, while in FMRSP vehicles collect new customers during routes and all customer are delivered at the destination(station).

In recent years, due to the fast development of technology, ride-sharing service companies (Uber, Lyft, Didi, etc.) based on smart phone and GPS have gained rapid growth. Thus the research attention has also grown considerably. Masoud and Jayakrishnan (2017) built the ride-matching problem as a binary programming problem and proposed a decomposition algorithm to solve the problem. Alonso-mora et al. (2018) developed an efficient algorithm to dynamically solve large scale ride-sharing problem instances in real-time. Stiglic et al. (2018) studied the benefits of integrating ride-sharing and public transport. Their results showed this system can both benefit the mobility sharing and the utilization of public transportation system.

As an extension of the ride-sharing problem, FMRSP, which specifically considers the ride-sharing for the first-mile scenario, has also gained broad attention these years. Yu Shen (2018) studied the integration of first-mile ride-sharing service and public transportation. Chen et al. (2020) explored the first-mile ride-sharing service using autonomous vehicles and developed a cluster-based algorithm for solving this problem. Furthermore, several studies (Bian and Liu, 2019b,a; Bian et al., 2020) included passenger preferences as part of FMRSP model.

1.2 First-mile Ride-sharing Model

A general mathematical model for the FMRSP derived from Chapter 3 can be presented as follows. We consider the operation of a fleet of homogeneous vehicles $\mathcal{K} := \{1, 2, ..., K\}$, each vehicle $k \in \mathcal{K}$ has a fixed capacity Q and is located at its origin $o(k)$. $\mathcal{N} := \{1, 2, ..., N\}$ contains the customers' requests received in the system, each customer $i \in \mathcal{N}$ can be picked up from her/his origin $o(i)$ and all customers have a common destination d (e.g., a station), which is located at $o(d)$. The revenue can be gained from each picked up customer is P_i. The operational cost generated per unit time is C, and the travel time T_{ij} between locations $o(i)$ and $o(j)$ is known in advance, where $i \in \mathcal{N} \cup \mathcal{K}$ and $j \in \mathcal{N} \cup d$. The decisions of the operator can be formulated as follows: x_{ij}^k takes value 1 if vehicle k directly travels between location $o(i)$ and $o(j)$, 0 otherwise, for all $i \in \mathcal{N} \cup \mathcal{K}$, $j \in \mathcal{N} \cup d$, $k \in \mathcal{K}$.

The problem can be formulated as follows with defined decision variables and parameters above.

$$\max \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}\cup\{d\}} P_i x_{ij}^k - \sum_{i\in\{k\}\cup\mathcal{N}} \sum_{j\in\mathcal{N}\cup\{d\}} \sum_{k\in\mathcal{K}} CT_{ij} x_{ij}^k \tag{1.1a}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}\cup\{d\}} \sum_{k\in\mathcal{K}} x_{ij}^k \leq 1 \qquad \forall i \in \mathcal{N} \tag{1.1b}$$

$$\sum_{i\in\mathcal{N}\cup\{k\}} x_{ij}^k = \sum_{i\in\mathcal{N}\cup\{d\}} x_{ji}^k \qquad \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{1.1c}$$

$$\sum_{j\in\mathcal{N}\cup\{d\}} x_{kj}^k = \sum_{j\in\mathcal{N}\cup\{k\}} x_{jd}^k \qquad \forall k \in \mathcal{K} \tag{1.1d}$$

$$\sum_{i\in\{k\}\cup\mathcal{N}} \sum_{j\in\mathcal{N}} x_{ij}^k \leq Q \qquad \forall k \in \mathcal{K} \tag{1.1e}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i \in \{k\}\cup\mathcal{N}, j \in \mathcal{N}\cup\{d\}, k \in \mathcal{K} \tag{1.1f}$$

The goal of the objective function 1.1a of the model is to maximize the profit for the first-mile ride-sharing system. The first term is the accumulated revenue gained from the customers assigned to vehicles, the second term is the summary of the operational cost generated for vehicles to pickup the customers and deliver them to the destination d.

Constraints 1.1b ensure each customer would be assigned to at most one vehicle. Constraints 1.1c state that after the vehicle k visit a customer $j \in \mathcal{N}$, it must move to another customer $i \in \mathcal{N}$ or final destination d. Constraints 1.1d ensure that each vehicle k departing from the original location has to terminate at the station d. Constraints 1.1e are the capacity constraints, which declare that the total customer assigned to a vehicle would not exceed its capacity Q.

This formulation is the core of a FMRSP, and in general it includes various additional service-specific constraints, including maximum waiting time, latest arrival time, and rebalancing. In the following chapters of thesis, we consider a number of these extensions.

1.3 Contributions

In the following chapters we consider variants of the FMRSP for different problem settings. Particularly, we include several problem-specific features including arrival deadlines, waiting times, rebalancing, and fleet management, and we implement the rolling horizon framework for FMRSP. Thus instead of only considering the customer requests received for current re-optimization, the customer requests that were accepted from previous re-optimization phase but have not been picked up would be considered as mandatory customers, which must be assigned to vehicles in current re-optimization phase. Time constraints are also taken into account for the chapters follows. Furthermore, we consider variants of the generalized FMRSP model for different problem settings.

Chapter 2 includes decision variables that denote whether a vehicle is dispatched or not in the current re-optimization phase. We assume that vehicles that were dispatched in previous

optimization phases must be dispatched in subsequent optimization phases. This ensures that drivers are hired for a certain minimum amount of time. Constraints related to the new decision variable and input are added to the model accordingly. A number of numerical experiment are proposed to compare a scenario where the fleet is controlled by the service provider to a scenario where drivers are independent and hired as needed. The result under several revenue sharing mechanisms show the superiority of the fleet control configuration in terms of profit.

Chapter 3 takes into account of the possibility of rebalancing. It aims at identifying locations where to relocate empty vehicles in order to prepare for future demand. For evaluating the efficienty of rebalancing and comparing different methods of identifying rebalancing centers, we include rebalancing benefit in the objective function and add constraints to regulate movements to rebalancing centers. These include constraints that only empty vehicles can be moved to rebalancing centers and bounds on the number of vehicles that can be moved to rebalancing centers.

Chapter 4 extends Chapter 3. For solving large scale FMRSP instances in a rolling horizon framework, an extended Adaptive Large Neighborhood Search meta-heuristic is designed for order dispatching and vacant vehicle rebalancing, five destroy operators and four repair operators are developed for generating problem specific neighborhoods. Numerical experiments are implemented for evaluating the proposed ALNS meta-heuristic. The result demonstrates the advantage of our proposed algorithm over the commecial solver (Gurobi) in both computation time and solution quality. Furthermore, the ALNS was used in a rolling-horizon framework to assess the benefits of rebalancing. The results show a significant service rate improvement in the configuration with rebalancing compare to the configuration without rebalancing.

1.4 Methodology

1.4.1 Mixed Integer Programming

Mixed Integer Linear Programming (MILP) is used in this thesis to model the first-mile ride-sharing problem.. A general MILP is of the form:

$$\min \quad c^\intercal x \tag{1.2a}$$

$$\text{s.t.} \quad Ax = b \tag{1.2b}$$

$$x_i \geq 0, \forall i \in \mathcal{C} \tag{1.2c}$$

$$x_i \in \mathbb{Z}, \forall i \in \mathcal{I} \tag{1.2d}$$

Where $x_i, i \in \mathcal{C}$ is a vector of $|\mathcal{C}|$ continuous variables, $x_i \in \mathbb{Z}, i \in \mathcal{I}$ is a vector of $|\mathcal{I}|$ integer variables, c is a vector of $|\mathcal{C}|+|\mathcal{I}|$ parameters, A is a matrix of appropriate dimension, and b is a vector of parameters. In addition to continuous decision variables, which are normally considered in Linear Programming (LP) models, MILP includes decision variables that are required to take integer values. This enormously increases the modeling power while also increasing the complexity, and in general there is no polynomial-time algorithm known for solving general MILP problems. In other words, MILP is NP-Hard. Particularly, the MILP with 0-1 binary variables as

integer variables are NP-Complete (Floudas, 1995). In practice, modern optimization techniques and solvers are developed to efficiently solve MILP, though solving the large scale instances to optimal can still be challenging.

In our work, we utilize the commercial solver Gurobi, which implements advanced Branch and Cut methods, and develop tailor-made Adaptive Large Neighborhood Search meta-heuristic to address a range of MILP instances with varying scales.

1.4.2 Rolling Horizon Simulation

Rolling Horizon Simulation (RHS) aims at supporting sequential decision-making for complex problems over a planning horizon.

The general process of RHS can be described as follows: First, break down the planning horizon into smaller intervals of time. Second, generate an initial solution with historical or simulated information and optimization methods such as commercial solvers, heuristics, etc. The solution is executed over the first time interval by simulating the real world system operations, to obtain the outcome based on the decision plan from initial solution. Third, the updated outcome will combine with the new input for the second time interval for re-optimization, thus the decision plan can be adjusted and updated for this time interval. Fourth, the process repeats until the last time interval of the planning horizon, during which the re-optimization process continuously updates the decision plan based on the simulated outcome of each previous time interval.

Particularly, in our work, the planning horizon (e.g. 1 hour) can be divided into 12 time intervals, where the range of each time interval is 5 minutes. The initial solution is generated with input information (new customer requests, vehicle locations and status) and using either Gurobi solver or Adaptive Large Neighborhood Search meta-heuristics. After the routing decision is made, the vehicles are dispatched along the updated routes during this time interval (5 mins) simulating the real world setting. At the beginning of second time interval, the status of the vehicles (location, empty seats) are updated. Customers assigned to vehicles in the first time interval but have not been picked up are put into "previous customers" set, in which each customer must be assigned to a vehicle and can not be rejected. With this updated vehicle status and previous customer information, together with new customers requests collected for second time interval, the re-optimization will start and the updated decision plan will be executed during the second time interval. The process repeats until finishing total 12 time interval for the one hour planning horizon.

1.4.3 Adaptive Large Neighborhood Search

Our proposed meta-heuristic is extended from the ALNS introduced in Pisinger and Røpke (2010), which can be presented as Algorithm 1.

Algorithm 1 ALNS algorithm

Input: a feasible solution x
$x^b = x$; $\omega^+ = (1, ..., 1)$, $\omega^- = (1, ..., 1)$
while *NoImprov* $\leq$ *maxNoImprov and* $i \leq$ *maxIter* **do**
  Select a destroy operator $d \in \Omega^+$ randomly using ω^+
  Select a repair operator $r \in \Omega^-$ randomly using ω^-
  $x^t = r(d(x))$
  if *accept*(x^t, x) **then**
    $x = x^t$
  if $c(x^t) < c(x^b)$ **then**
    $x^b = x^t$
  update ω^+ and ω^-
return x^b

The neighborhood of ALNS is defined by destroy and repair operators, which are denote as d and r, respectively. The sets of destroy and repair operators are denoted as Ω^+ and Ω^-, respectively, while $d \in \Omega^+$ and $r \in \Omega^-$. In Chapter 4, five destroy operators are specifically designed for our proposed problem: Nearby Location Removal (NLR), Same Arrival Time Removal (SATR), Free Part of Routes (FPR), Remove Optional (RO) and Destroy Worst (DW). In the meantime, four repair operators are proposed as well Random Insertion Customer First (RICF), Random Insertion Rebalancing First (RIRF),Greedy Insertion (GI), Best Customer Insertion (BCI). Variables $\omega^+ \in \mathbb{R}^{|\Omega^+|}$ and $\omega^- \in \mathbb{R}^{|\Omega^-|}$ store the weights of destroy and repair operators and all destroy and repair operators initiate with same weights at the beginning of the algorithm, respectively. A roulette wheel principle is used for the selection of destroy operator and repair operator, the probability ϕ_d^- of choosing destroy operator d at iteration i is computed using

$$\phi_d^- = \frac{\omega_d^-}{\sum_{j=1}^{\Omega^-} \omega_j} \tag{1.3}$$

and the algorithm also applies for choosing repair operator. The variable x is the current solution, x^b is the best solution obtained so far, and x^t is the temporary solution that can be discarded or accepted in later process depends on the solution quality. Destroy function $d(x)$ returns neighborhood of x which is partially destroyed, and $r(d(x))$ apply repair function on partially destroyed x and returns a new temporary solution x^t.

Algorithm 1 starts by initiating a feasible solution x and global best solution x^b. After the iteration starts, a new temporary solution x^t after applying $r(d(x))$. Then x^t would be evaluated and a score Φ would be updated the corresponding destroy and repair operators depends on quality of x^t according to formula:

$$\Phi = \begin{cases} \alpha_1 & \text{if } x_t \text{ is better than } x_b, \\ \alpha_2 & \text{if } x_t \text{ is better than } x, \\ \alpha_3 & \text{if } x_t \text{ is not better than } x \text{ but accepted}, \\ \alpha_4 & \text{if } x_t \text{ is rejected.} \end{cases} \tag{1.4}$$

Where parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ normally set as $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \alpha_4$ as high score Φ indicates a good method. The weights of operators ω^+ and ω^- are updated using

$$\omega^+ = \lambda\omega^+ + (1-\lambda)\omega^+, \omega^- = \lambda\omega^- + (1-\lambda)\omega^- \tag{1.5}$$

at the end of the each segment l, which is set as $l = 100$ iterations in our work. Where $\lambda \in [0, 1]$ decides how sensitive the weights are to the scores gained in last segment for the destroy and repair operators.

The process repeats until the stopping criteria are met, while the stopping criteria is defined by maximum no improvement iteration number(maxNoImprov), maximum iteration number (maxIter) and Simulated Annealing algorithm in Chapter 4.

Chapter 2

Fleet Size Control in First-mile Ride-sharing Problem

This chapter contains the paper Ye et al. (2022a).

Abstract

The first-mile problem, which refers to the design of transport services that connect passengers to their nearby transit station, has attracted growing attention in recent years. In this paper we consider *first-mile ride-sharing* services and study the problem of optimally determining the fleet size and assigning vehicles to transport requests. We formulate the problem as a mixed-integer program and present a number of numerical experiments based on a small-scale system to analyse different configurations of the service, namely with and without fleet control (FC). Result shows that a configuration with FC is superior in terms of profits while service rates can be higher in a configuration without FC, depending on the revenue-sharing mechanism.

Keywords: Fleet control, First-mile, Ride-sharing

2.1 Introduction

As the size of urban areas and population increases, and with them road congestion and air pollution T. Eiichi (2014), ride-sharing services emerged as a more sustainable urban transportation solution, linked to e.g., a reduction in the number of private cars on the road, emissions and road congestion A. Abubakr O (2019). Particularly, first-mile ride-sharing services, that is transportation services that connect passengers to their nearby transit station using shared vehicles, have attracted growing attention. According to the NYC taxicab data Com (2021), there were 3122731 taxi trips to the Pennsylvania Station in New York City in 2017 that is, on average 8555 taxis traveled to this station every day. However, 70.1% of these trips had only one passenger

on board Bian et al. (2020). This leaves a significant potential for the development of shared trips to transit stations.

In this article, we study the problem of dimensioning a first-mile ride-sharing fleet and optimizing the process of order dispatching, which means assigning transport requests to vehicles. The problem of dimensioning shared fleets has recently been studied in a number of articles. In P. M. Boesch and Axhausen (2016); K. Treleaven and Frazzoli (2013); K. Spieser (2014) the focus is on so-called *ride-hailing* services, where each vehicle satisfies one request at a time (i.e., rides are not shared). In this paper, we focus on *ride-sharing* services, where each vehicle satisfies multiple requests simultaneously. In H. K. R. F. Pinto and Verbas (2019) the size of a fleet of autonomous vehicles is determined. The autonomous vehicles are integrated into a transit network, and run according to predefined patterns. On the contrary, in our case, vehicles do not run according to predefined patters, but react to the arrival of requests. In A. Wallar (2019) the focus is on ride-sharing services. The authors develop a method to determine how many vehicles are needed and where they should be located in order to service all the requests. The method is designed for offline use based on historical demand. In this article, we focus on online optimization and do not require vehicles to be placed at pre-determined locations. Also in K. Winter and Arem (2016) a ride-sharing service is considered. The authors use simulation to estimate the minimal fleet size required for a system that connects a university campus to a train station. This system requires predefined pickup and drop-off locations, while our system does not have this requirement. Moreover, in M. M. Vazifeh and Ratti (2018) the authors propose a vehicle-sharing network model to obtain the optimal and near optimal solution fleet size for the urban-scale data. The method obtains a re-organization of the taxi dispatching, without assuming ride-sharing. Instead we provide an explicit mathematical model for the problem. Finally, unlike in B. A. Beirigo and Schulte (2022), where the authors consider elastic vehicle supply, which considers hiring privately-owned freelance autonomous vehicles, we focus on controlling the fleet size without hiring external fleets.

The main contribution of this work can be stated as follows:

- We propose a mixed-integer programming (MIP) problem for online optimization of order dispatching and fleet size, which accounts for several constraints such as desired arrival time and maximum waiting time. The model is flexible enough to accommodate different ride-sharing business models.
- While existing studies focus mainly on metrics such as idle time, waiting time A. Wallar (2019) and costs K. Winter and Arem (2016), we use the model to assess different configurations of the service. Particularly, we consider a configuration where the service provider owns the fleet and hires drivers as well as configurations where the service provider does not own the fleet but acts as a platform that optimally connects passengers to vehicles. In this case the service provider shares revenues and costs with the drivers and we assess two different sharing schemes.
- We test our model on a number of artificial instances generated to mimic the different configurations of a small-scale service. Our model for online optimization is used in a rolling

horizon procedure with periodic re-optimizations based on the arrival of new requests and updated system information (e.g., vehicles position).

The remainder of this article is organized as follows. In Section 2.2 we provide a formal definition of the problem and introduce the corresponding mathematical model. In Section 2.3 we describe our numerical experiments and present results. Finally, we draw conclusions in Section 2.4.

2.2 Problem Description and Mathematical Model

In this section, we provide a description of the problem followed by a mathematical model.

A set of vehicles $\mathcal{K}$, with each vehicle $k \in \mathcal{K}$ initially available at the respective location $o(k)$, is employed to provide first-mile ride-sharing services to a set of customers from their respective location to a designated station d located at $o(d)$. The set of customers is partitioned into a set of mandatory customers $\mathcal{N}_P$, representing the customers whose transportation request has been accepted during a previous optimization phase but not yet fulfilled, and a set of new customers $\mathcal{N}_C$ whose requests have arrived recently and may or not be accepted. We let $o(i)$ be the location of request $i \in \mathcal{N}_P \cup \mathcal{N}_C$. The company is to decide i) which vehicles to dispatch, if not already dispatched, ii) which new requests to accept and iii) how to assign ride-sharing requests to vehicles.

Each vehicle v has V_k passengers on board at the beginning of the operational period. V_k is an input parameter determined by the requests assigned to vehicle k in previous optimization phases. Let Q be the total capacity of a vehicle (we assume a homogeneous fleet). Parameter U_k is equal to 1 if vehicle k has been dispatched in any of the previous re-optimization phases, 0 otherwise. In the latter case V_k is zero. The company bears a fixed cost $\bar{C}$ for each vehicle dispatched and a cost C per unit of travel time. Picking up new customer $i \in \mathcal{N}_C$ yields a revenue P_i. Such customers may however be rejected. The revenue for customers in $\mathcal{N}_P$ has been collected during a previous optimization phase, when the request was accepted, and these customers are now treated as mandatory.

The operating period starts at time T. The travel time between locations $o(i)$ and $o(j)$, with $i \in \mathcal{K} \cup \mathcal{N}_P \cup \mathcal{N}_C$, $j \in \mathcal{N}_P \cup \mathcal{N}_C \cup \{d\}$, is T_{ij}. Each customer has a requested pickup time T_i^P and arrival time T_i^A and we let $T^L := max_{i \in \mathcal{N}_P \cup \mathcal{N}_C}\{T_i^A\}$. The difference between the requested pickup time and the actual pickup time cannot be larger than Δ. Furthermore, each vehicle has a latest arrival time T_k representing the earliest arrival time among the V_k passengers already on board vehicle k at the beginning of the planning phase.

We introduce the following decision variables. Let x_{ij}^k, for $i \in \{k\} \cup \mathcal{N}_P \cup \mathcal{N}_C, j \in \mathcal{N}_P \cup \mathcal{N}_C \cup \{d\}, k \in \mathcal{K}$, be equal to 1 if vehicle k moves directly between $o(i)$ and $o(j)$, 0 otherwise. Let t_k^A be the actual arrival time of vehicle k to the station, for $k \in \mathcal{K}$. Let t_i^P be the actual pickup time of customer i, for $i \in \mathcal{N}_P \cup \mathcal{N}_C$. Finally, let S_k be equals to 1 if vehicle k is dispatched in the current re-optimization phase, 0 otherwise.

The problem is hence

$$\max \quad \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{N}_C}\sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}} P_i x_{ij}^k - \sum_{k\in\mathcal{K}} \bar{C} S_k \tag{2.1a}$$

$$- \sum_{i\in\{k\}\cup\mathcal{N}_C\cup\mathcal{N}_P}\sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}}\sum_{k\in\mathcal{K}} C T_{ij} x_{ij}^k$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}}\sum_{k\in\mathcal{K}} x_{ij}^k \le 1 \qquad \forall i \in \mathcal{N}_C \tag{2.1b}$$

$$\sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}}\sum_{k\in\mathcal{K}} x_{ij}^k = 1 \qquad \forall i \in \mathcal{N}_P \tag{2.1c}$$

$$\sum_{i\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{k\}} x_{ij}^k = \sum_{i\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}} x_{ji}^k \qquad \forall j \in \mathcal{N}_C\cup\mathcal{N}_P, k \in \mathcal{K} \tag{2.1d}$$

$$\sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}} x_{kj}^k = \sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{k\}} x_{jd}^k \qquad \forall k \in \mathcal{K} \tag{2.1e}$$

$$\sum_{i\in\mathcal{N}_C\cup\mathcal{N}_P\cup\mathcal{K}} x_{id}^k \le 1 \qquad \forall k \in \mathcal{K} \tag{2.1f}$$

$$t_i^P + T_{ij} \le t_j^P + T^L(1 - \sum_{k\in\mathcal{K}} x_{ij}^k) \qquad \forall i,j \in \mathcal{N}_C\cup\mathcal{N}_P \tag{2.1g}$$

$$T + T_{kj} \le t_j^P + T^L(1 - x_{kj}^k) \qquad \forall j \in \mathcal{N}_C\cup\mathcal{N}_P, k \in \mathcal{K} \tag{2.1h}$$

$$t_i^P - T_i^P \le \Delta \qquad \forall i \in \mathcal{N}_C\cup\mathcal{N}_P \tag{2.1i}$$

$$t_k^A \le T_i^A + T^L(1 - \sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{k\}} x_{ji}^k) \qquad \forall i \in \mathcal{N}_C\cup\mathcal{N}_P, k \in \mathcal{K} \tag{2.1j}$$

$$t_k^A \le T_k \qquad \forall k \in \mathcal{K} \tag{2.1k}$$

$$t_j^P + T_{jd} x_{jd}^k \le t_k^A + T^L(1 - x_{jd}^k) \qquad \forall j \in \mathcal{N}_C\cup\mathcal{N}_P, k \in \mathcal{K} \tag{2.1l}$$

$$V_k \le Q \sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}} x_{kj}^k \qquad \forall k \in \mathcal{K} \tag{2.1m}$$

$$\sum_{i\in\{k\}\cup\mathcal{N}_C\cup\mathcal{N}_P}\sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P} x_{ij}^k + V_k \le Q \qquad \forall k \in \mathcal{K} \tag{2.1n}$$

$$S_k \ge \sum_{j\in\mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}} x_{kj}^k \qquad \forall k \in \mathcal{K} \tag{2.1o}$$

$$S_k \ge U_k \qquad \forall k \in \mathcal{K} \tag{2.1p}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i \in \{k\}\cup\mathcal{N}_C\cup\mathcal{N}_P, j \in \mathcal{N}_C\cup\mathcal{N}_P\cup\{d\}, k \in \mathcal{K} \tag{2.1q}$$

$$t_k^A \in \mathbb{R}^+ \qquad \forall k \in \mathcal{K} \tag{2.1r}$$

$$t_i^P \in \mathbb{R}^+ \qquad \forall i \in \mathcal{N}_C\cup\mathcal{N}_P \tag{2.1s}$$

$$S_k \in \{0,1\} \qquad \forall k \in \mathcal{K} \tag{2.1t}$$

The objective function (2.1a) represents the total profit made of the revenue of picking up customers minus dispatching and traveling costs.

Constraints (2.1b) and (2.1c) state that new customers may be picked up at most once, while mandatory customers must be picked up exactly once, respectively. Mandatory customers are those already accepted during a previous re-optimization phase. Constraints (2.1d) are flow conservation constraints, that is whenever a vehicle visits a customer it must also move to

another customer or to the station. Constraints (2.1e) state that, if a vehicle departs from its original location $o(k)$ it must terminate its journey at the station. Constraints (2.1f) ensure that each vehicle travels to the station at most once.

Constraints (2.1g) state that if customers j is picked up by vehicle k immediately after picking up customer i, then the actual picking up time of customer i plus the travel time between customer i and j should be less or equal to customer j's actual pick up time. Constraints (2.1g) may be improved as described by M. Desrochers (1991), Y. Yuan and Semet (2020). Similarly, (2.1h) define the pickup time when the vehicle comes directly from its original location. Constraints (2.1i) ensure that the difference between actual pick up time and the requested pick up time of each customer do not exceed the maximum waiting time Δ. Constraints (2.1j) ensure that the arrival time of vehicle k at the station is earlier than the requested arrival time of any of the customers on board. For instance, if customer i is picked up by vehicle k, the right-hand-side will always be equal to T_i^A, which means the actual arrival time of vehicle k needs to be less than or equal to the requested arrival time of customer i. Constraints (2.1k) ensure that the actual arrival time of vehicle k is earlier than the earliest requested arrival time T_k of the passengers already on board at the beginning of the planning phase. Constraints (2.1l) define the relationship between pickup and arrival time. For example, if j is the last customer picked up by the vehicle k before arrive at the station, then x_{jd}^k takes value 1, the left-hand-side becomes $t_j^P + T_{jd}$ and the right-hand-side becomes t_k^A, which means that the actual pick up time of j plus the travel time between customer j and station should be 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^P and the-right-hand side becomes $t_k^A + T^L$, which always holds. Altogether, constraints (2.1j) and (2.1k) define an upper bound on t_k^A while (2.1l) define a lower bound.

Constraints (2.1m) state that the vehicles that already have customers on board at the beginning of the planning period must be dispatched, while (2.1n) ensure that the total capacity is not violated. Constraints (2.1o) set S_k to 1 as soon as vehicle k is dispatched. Constraints (2.1p) ensure that the vehicles already dispatched in previous re-optimization phases are still available in the current re-optimization phase.

2.3 Numerical Experiment

In this section we report on the results of our numerical experiments. The scope of the experiments is to assess, in terms of profits and service rates, two different configurations of the service which we refer to as *with fleet control* (wFC) and *without fleet control* (woFC). The former refers to a situation where the service provider owns the fleet and bears all costs and profits. All vehicles are initially idle and the provider decides which of them to dispatch, paying a fixed cost for each dispatched vehicle which covers e.g., the salary of the driver, wear and maintenance. In the configuration woFC the service provider does not own the fleet and acts as platform that connect passengers to vehicles. In this case vehicles are considered to be always available so that the provider does not pay a fixed cost upon dispatching a vehicle. However, in this case, revenues are shared between the driver and the service provider. Particularly, we test

two different revenue-sharing schemes inspired by the business configuration adopted by Uber Uber (2020). In the first scheme, which we refer to as woFC-40, the company keeps 40% of the trip fare (thus the driver keeps the remaining 60%) but bears variable costs (e.g., fuel/charge). In the second scheme, which we refer to as woFC-25, the company keeps only 25% Uber (2020) of the trip fare but does not cover variable cost.

The different configurations are tested by using an appropriate setting of the parameters in model (2.1). All problems are solved using the Python libraries of GUROBI 9.5.0 a server equipped with Intel Core i5 CPUs and 16GB of RAM.

2.3.1 Instance Generation

We test our model on a number of artificial randomly generated instances that mimic the different configurations of the service. The instances are generated as follows.

The business area is represented by a 4×3 rectangular area with the station located at the center of the area $(2, 1.5)$. We randomly generate travel requests with their respective pickup location, pickup time, drop-off time, and fare. Requests pickup locations are randomly generated in the 4×3 area. For each request, the requested pickup time T_i^P is randomly generated uniformly between 0 and 5 minutes, the requested arrival time T_i^A is the sum of the requested pickup time, the travel time between the pickup location i and the station d, and a random generated buffer time between 5 and 10 minutes. Since the focal rectangular area is continuous, travel times T_{ij} are calculated using Euclidean distances and assuming an average speed of 36 km/h following Han (2019). The unit cost C of the transportation is set to 11.25\$/h nyc (2008) and trip revenues P_i are computed using an hourly rate set to 56.16\$/h with a base fare of \$2.5 following nyc (2020b).

In the configuration wFC the fixed cost $\bar{C}$ is set to \$2.8 and represents mainly the salary of the driver. This cost is obtained using Lyft salary – \$33.75/h, see nyc (2020a) – as our reference and considering that our method re-optimizes every 5 minutes. Thus, the fixed cost $\bar{C}$ for each re-optimization is \$33.75/60*5 = \$2.8. In Section 2.3.2 we assess the impact of this parameter. In the configurations woFC-40 and woFC-25, the trip fare is reduced to $0.4P_i$ and $0.25P_i$, respectively to mimic the corresponding revenue-sharing scheme. In the configuration woFC-40 the company covers variable costs using the unit transportation cost C cost described above, while in the configuration woFC-25 the company does not cover transportation cost, so $C = 0$. In both cases the dispatching cost is $\bar{C} = 0$.

We generate instances with different number of customers and vehicles. Particularly, we create instance classes named $V|\mathcal{K}|C|\mathcal{N}_C|$ with vehicles in $|\mathcal{K}| \in \{8, 10, 12, 20\}$, and customers, $|\mathcal{N}_C| \in \{6, 8, 12\}$. As an example, $V8C6$ indicates a class of instance with 8 vehicles available for dispatch and 6 new customers in each re-optimization phase. Such instances are perhaps representative of a potential service in a small peripheral station or in a small urban context. Finally, for each instance class we randomly generate 3 instances.

Finally, we consider a one-hour planning horizon with online re-optimization every 5 minutes, leading to a total of 12 re-optimizations for each instance. It should be noted that at each re-

optimization we update the status of the system (i.e, vehicles positions and customers on board) and randomly select the corresponding number of new customers, while considering previously accepted customers as mandatory.

Figure 2.1 illustrates how information is updated between re-optimizations on an instance with three vehicles and four new customers for each re-optimization phase. Figure 2.1(a) illustrates the initial position of vehicles (yellow squares), customers (blue circles) and customers that have been assigned to vehicles in the previous re-optimization phases but have not yet been picked up (red circles) – the station is identified by the black triangle. Figure 2.1(b) shows the routes computed for the vehicles in the corresponding re-optimization phase. It can be noticed that four customers (three new and one mandatory) have been assigned to three vehicles in Route1, Route2, Route3 (their request has been accepted) while one customer has not (the request has not been accepted). Figure 2.1(c) shows the location and remaining portion of the route of the vehicles before the next re-optimization phase, as well as the location of four new customers (green circles) arrived in the system in the mean time and whose request will be handled in the next re-optimization phase. The four customers assigned to vehicles in the previous optimization phase are now on board the vehicles, while the customer whose request was rejected has left the system. Thus the three vehicles currently have customers on board and are on their way to the station. Nevertheless, their routes may change in the next re-optimization phase in order to pickup new customers.

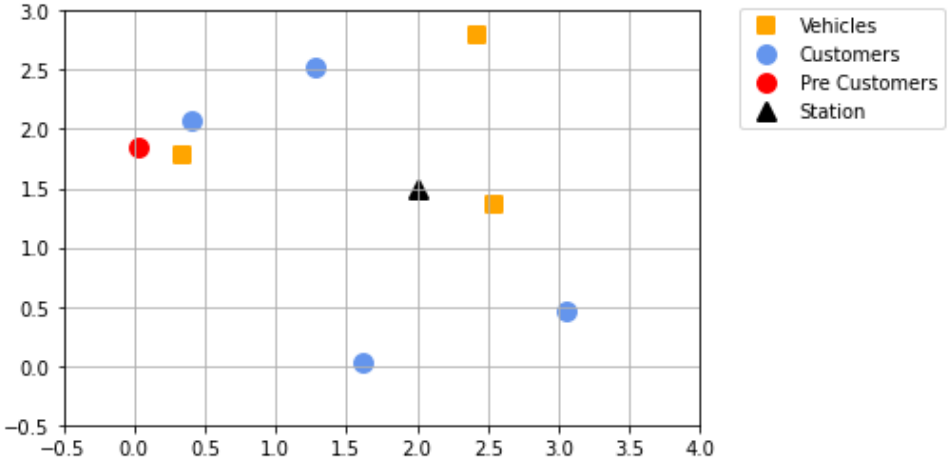
2.3.2 Results

We report now on the performance of the different configurations of the service.

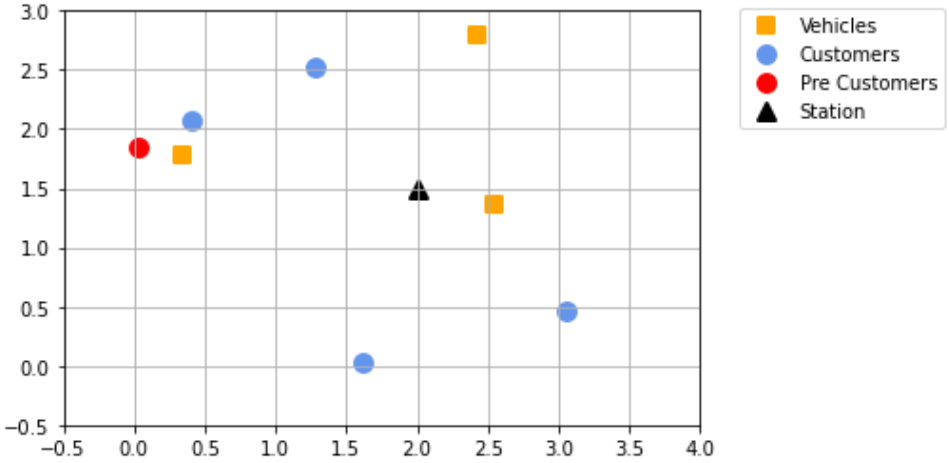
Figure 2.2 reports the dispatch rate, that is the percentage of vehicles dispatched. It can be noticed that in the case wFC, as intuition suggests, the dispatch rate generally increases with the number of customers and decreases with the size of the fleet. When the fleet counts 20 vehicles, there is no case in which the entire fleet is dispatched, indicating that the fleet is larger than needed. On the contrary, in the configurations woFC all vehicles are dispatched, meaning that they are not only available, but actually move from their original location to perform some transportation task. It appears, therefore, that configurations woFC may lead to using more vehicles than are actually necessary.

Figure 2.3 reports the service rate obtained with the different configurations, that is the number of requests satisfied over the total number of requests received in the one-hour planning horizon. It can be noticed that configuration woFC-25 competes and outperforms the configuration wFC. In the configuration woFC-25 the company is insensitive to transportation costs and does not bear fixed costs, therefore also solutions require a long travel time or would perhaps require an additional vehicle, thus potentially inefficient in a model for woFC-25 to become profitable. Such solutions become even more unappealing in the configuration woFC-40 where transportation costs are still born by the company in exchange for a higher revenue share.

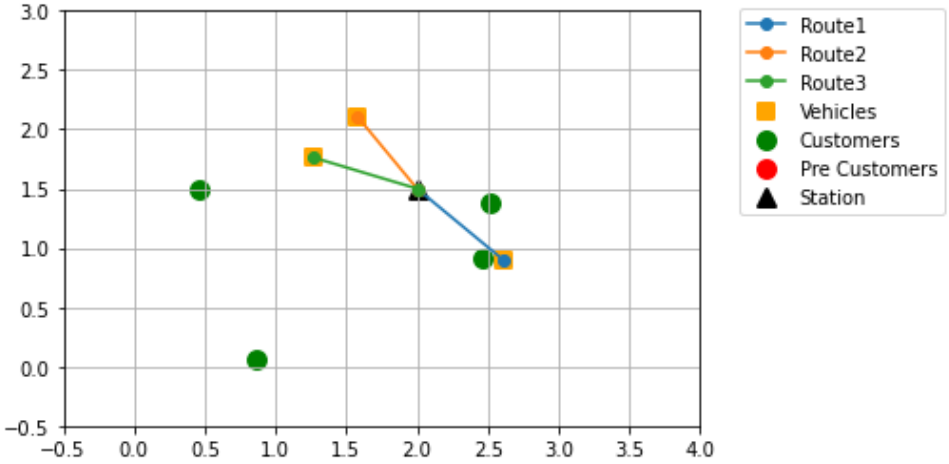
Finally, Figure 2.4 illustrates that the configuration wFC leads to the highest profits for the company, despite a slightly lower service rate compared to the configuration woFC-25 (see Fig-



(a) Distribution



(b) Optimized Routes



(c) Movement

Figure 2.1: Illustration of the solving process

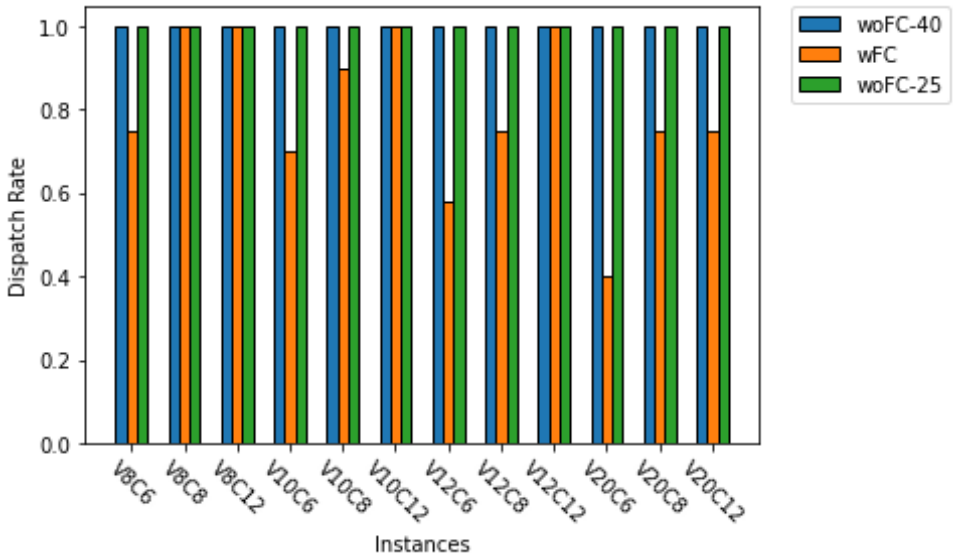


Figure 2.2: Dispatch rate for the configuration wFC and woFC. For each instance type the results are the average over three different instances.

ure 2.3). Both configurations woFC lead to worse profits regardless of the revenue-sharing mechanism.

We report now on the impact of the fixed dispatch cost $\bar{C}$ in the configuration wFC. This in turn sheds light on the effect of using different types of vehicles, maintenance contracts, or salaries. Particularly, we assess dispatch rate, service rate and profit with different values of the fixed dispatch cost $\bar{C} \in \{1.4, 2.1, 2.8, 3.5\}$. Results are reported in Figures 2.5 to 2.7.

Figure 2.5 shows the intuitive pattern that, as the dispatch cost increases, the dispatch rate decreases. We also observe that the dispatch rate increases with the number of customers. Similarly, in Figure 2.6 it can be observed that the service rate increases as the fixed dispatch cost decreases. Nevertheless, the service rate remains rather high for all values of the fixed cost, illustrating that the model is able to find cost-efficient solutions also with fewer vehicles dispatched. Finally, in Figure 2.7 we also observe a similar trend, where profits are negatively affected by fixed costs. The effect appears more severe as the number of customers grows.

2.3.3 Solution time

Finally, we investigate the solution time. In our numerical experiments we solve each instance in a rolling horizon process. The average solution time and optimality gaps for different instances are reported in Table 2.1. Average values are computed over three randomly generated instances obtained with same combination of $|\mathcal{N}_C|$ and $|\mathcal{K}|$, 12 re-optimization phases for each instances, and three configurations, namely wFC, woFC-40 and woFC-25. Table 2.1 illustrates that all instances can be solved to provable optimality within 0.5 seconds. There is thus room for scaling up the problems to larger instances which represent more closely a real-life scenario. To this end, Table 2.2 provides some insights on how the model handles bigger instances. It can be noted that, with a fleet of 20 vehicles, the quality of the solutions decreases with the number of

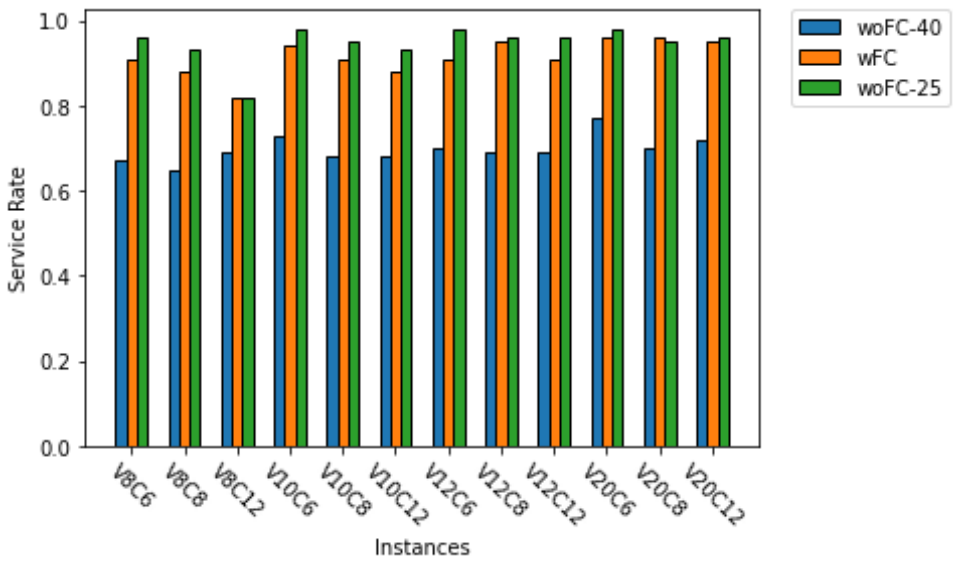


Figure 2.3: Service Rate for the configuration wFC and woFC. For each instance type the results are the average over three different instances.

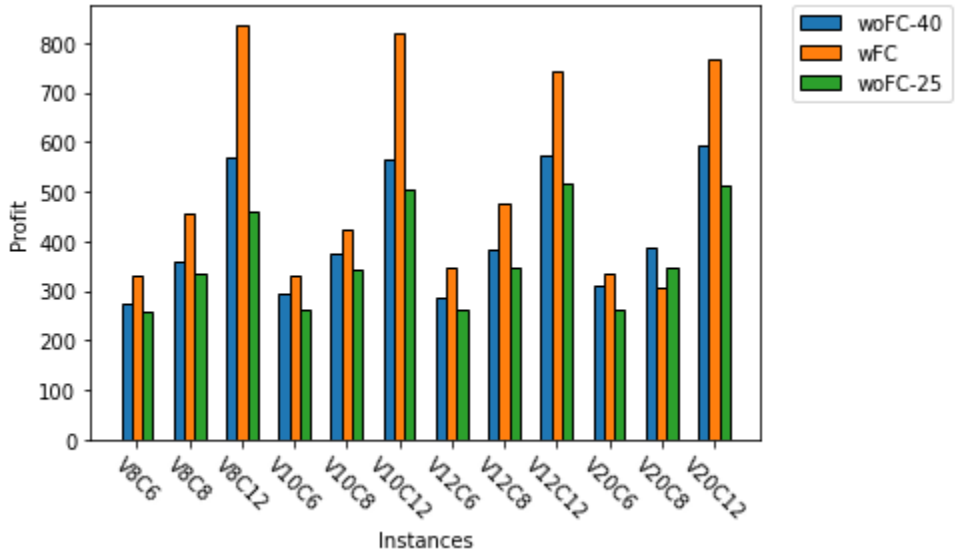


Figure 2.4: Profit for the configuration wFC and woFC. For each instance type the results are the average over three different instances.

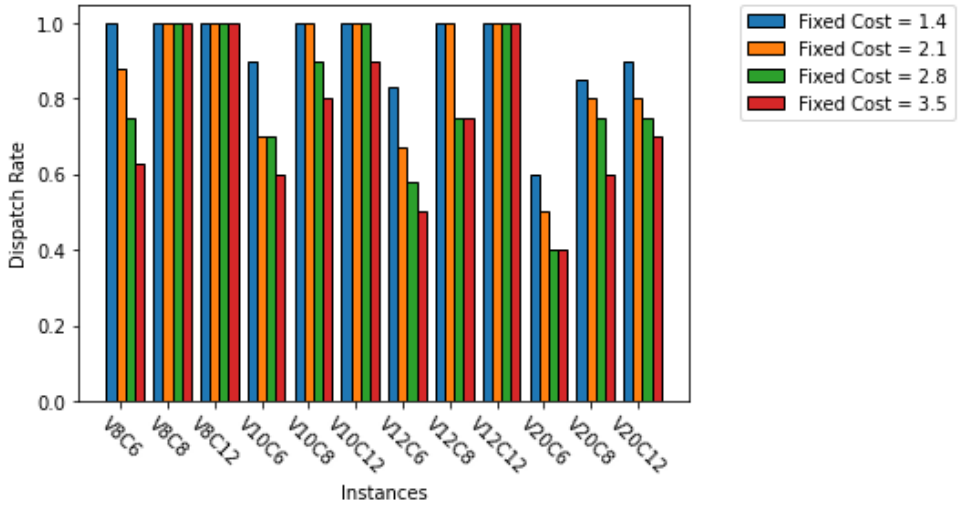


Figure 2.5: Dispatch rate with different fixed dispatch costs $\bar{C}$. For each instance type the results are the average over three different instances.

Table 2.1: Optimality gap and solution time

Instance	**Solution time [s]**	**Optimality Gap**
V8C6	0.03	0%
V8C8	0.04	0%
V8C12	0.11	0%
V10C6	0.03	0%
V10C8	0.05	0%
V10C12	0.14	0%
V12C6	0.04	0%
V12C8	0.06	0%
V12C12	0.18	0%
V20C6	0.06	0%
V20C8	0.08	0%
V20C12	0.41	0%

customers.

2.4 Conclusion

In this paper we proposed a MIP model for optimal order dispatching and fleet size control in a first-mile ride-sharing service. The model was used in a set of numerical experiments, where we compared different configurations of the service (i.e., business models), with and without fleet control and with different revenue-sharing schemes. Results shows that the configuration with fleet control (i.e., where the company owns the fleet) has a relatively high service rate and outperforms the configurations without fleet control in terms of profits. Results also show that fixed dispatch costs have a critical impact on both service rate and profits. We can observe that, as the fixed cost increases, the number of vehicles dispatched decreases and profits shrink. Nevertheless, service rates remain rather high, showing that the model is able to find cost efficient solutions with fewer vehicles. Finally, the tests illustrate that the model is flexible

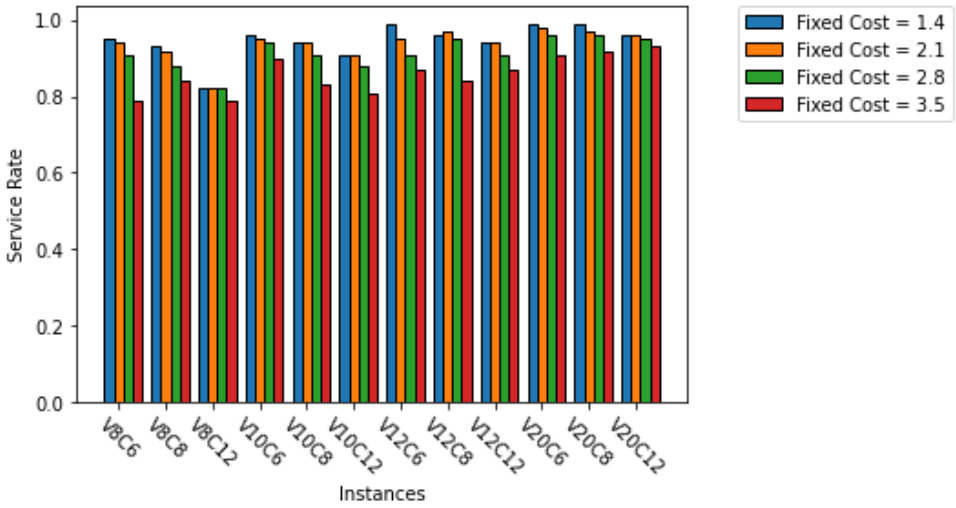


Figure 2.6: Service rate with different fixed dispatch costs $\bar{C}$. For each instance type the results are the average over three different instances.

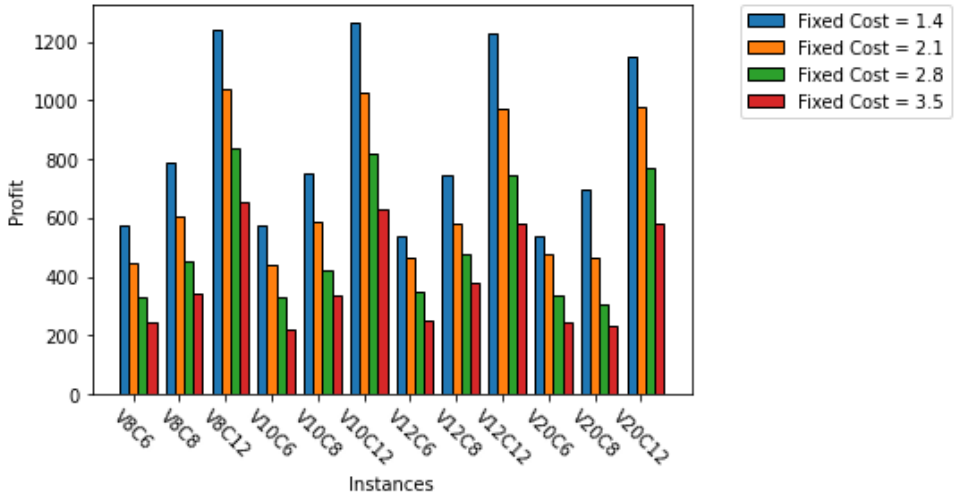


Figure 2.7: Profit with different fixed dispatch costs $\bar{C}$. For each instance type the results are the average over three different instances.

Table 2.2: Optimality gap and solution time on larger instances. For each instance size, maximum and average are taken over three randomly generated instances.

Instance	**Avg. sol. time**	**Avg. Gap**	**Max sol. time**	**Max Gap**
V20C20	15.59	0%	300	14%
V20C25	39.54	2%	300	26%
V20C30	96.44	4%	300	41%

enough to adapt to different configurations of the service and thus may serve real-life analysis of ride-sharing services. As an example, the model could help assess revenue-sharing mechanisms other than the one studied in this article.

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.

Chapter 3

Order Dispatching and Vacant Vehicles Rebalancing for the First-mile Ride-sharing Problem

This chapter contains the paper Ye et al. (2022b).

Abstract

Given a set of transport requests to a transit station and a set of homogeneous vehicle, both geographically dispersed in a business area, the First-Mile Ride-Sharing Problem (FMRSP) consists of finding least cost vehicle routes to transport passengers to the station by shared rides. In this paper we formulate the problem as a mathematical optimization problem and study the effectiveness of preventive movements of idle vehicles (i.e., rebalancing) in order to anticipate future demand. That is, we identify promising rebalanacing locations based on historical data and give the model incentives to assign vehicles to such location. We then assess the effectiveness of such movements by simulating online usage of the mathematical model. The results show that rebalancing is consistently preferable both in terms of profits and service rate. Particularly, in operating contexts where the station is not centrally located, rebalancing movements increase both profits and service rates by around 30% on average.

Keywords: First-mile, Ride-sharing, Order dispatching, Rebalancing, Rolling Horizon

3.1 Introduction

Ride-sharing services, which are linked to a reduction of the number of private cars on the road, emissions and congestion (A. Abubakr O, 2019), have emerged as a potential solution to the increase in road congestion and air pollution generated by growing urban areas and population

(a) All Trips (b) Shared Trips

Figure 3.1: Taxi trips to Penn Station. Data from Commission and Limousine (2020).

(T. Eiichi, 2014). Such services have yet significant potential for development. As an example, according to the NYC taxicab data (Commission and Limousine, 2020), during January 2020 there were 363,874 taxi trips to the Pennsylvania Station, a fairly busy transit station in New York City see Figure 3.1a, that is, on average 12,129 taxis trips daily to the station. Of these, only 6% were shared by multiple passengers, see Figure 3.1b, which leaves significant margins for more efficient connections to the station.

An effective implementation of ride-sharing services requires adequate responses to potentially frequent changes in demand patterns during the day that may determine geographical mismatches between demand as supply. Figure 3.2 illustrates the location of the requests of transportation to Pennsylvania Station during January 2014, showing that the majority of the requests arrive from the North-East area, whereas much fewer requests arrive from the remaining zones of the city. This suggests implementing mechanisms that prepare the geographical distribution of the fleet in such a way to anticipate demand and perhaps reduce waiting and response times as well as service rate.

The existing literature study various aspects of ride-sharing services, including pricing mechanisms (Bian and Liu, 2019a,b; Bian et al., 2020; Chen and Wang, 2018), integration with public transport (Yu Shen, 2018), order dispatching and vehicle routes (Wang, 2019; Chen et al., 2020). Conversely, strategies for anticipating demand through, e.g. preventive or rebalancing movements (Wen et al., 2018), remain, to a large extent, an open research question. Particularly, efficient ways to simultaneously determine both dispatching and rebalancing movements have, to the best of our knowledge, been neglected.

We contribute to filling this gap by providing a mathematical programming model for joint order dispatch and rebalancing decisions in a first-mile ride-sharing service which transports passengers from their initial location to a common destination (e.g., a transit station). We will

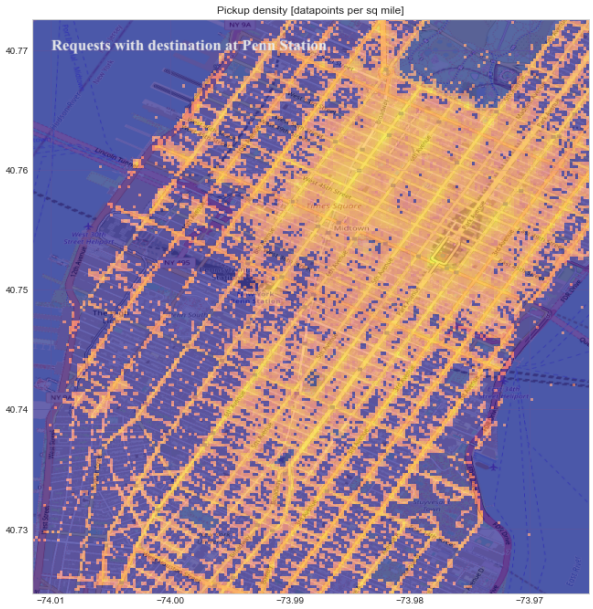


Figure 3.2: Distribution of trips to Penn Station

refer to this decision problem as the First-Mile Ride-Sharing Problem (FMRSP). In addition, we propose two strategies for identifying promising locations where to rebalance empty vehicles. The model and rebalancing strategies are tested in a simulation framework which replicates on-line usage.

The rest of this paper is structured as follows. In Section 3.2 we review the related literature and underline the contribution of this article. In Section 3.3 we formally introduce the problem and the corresponding mathematical programming model. In Section 3.4 we describe two methods for deciding where to relocate vehicles in anticipation of future demand. In Section 3.5 we describe the simulation framework and the numerical experiments we performed with the model and illustrate the results. Finally, we draw conclusions in Section 3.6.

3.2 Literature review

The routing decisions considered in the FMRSP share similarities with those involved in well studied routing problems. Among these we find the Vehicle Routing Problem (VRP). Starting from the seminal paper of (Dantzig and Ramser, 1959), several exact and heuristics algorithms were proposed to solve VRPs (Bräysy and Gendreau, 2005; Bertsimas et al., 2019) and several flavors of the problem have been studied, see e.g., the surveys (Kumar and Panneerselvam, 2012; Pillac et al., 2013; Lin et al., 2014; Ritzinger et al., 2016; Braekers et al., 2016). One of the major differences between the FMRSP and the different variants of the VRP is that the VRP consists of designing tours returning to the depot, while the FMRSP designs open paths from the vehicle origins to a common destination. The FMRSP focuses on picking up and transporting customers from multiple locations to the destination (station) while the VRP delivers service or goods to customers. Furthermore, in the VRP the vehicles typically operate from a common depot, though some variants considering multiple depots do exist.

Particularly, the FMRSP shares features with the Dial-a-ride Problem (DARP) and the Pick-up-and-delivery Problem (PDP) (Cordeau and Laporte, 2003; Ropke and Cordeau, 2009; Berbeglia et al., 2010), which are generalizations of the VRP. A comprehensive review of DARP and PDP can be found in Ho et al. (2018). The goal of the DARP is to minimize the cost/time to transport a set of passenger by means of a fixed fleet of vehicles. Requests have different pickup and delivery locations, and the vehicles can pick up more than one passengers at a time. Also for the DARP different variants can be found, such as where the objective is to minimize the detour for the customers on board the vehicles Pfeiffer and Schulz (2022). The DARP can be considered as a variant of the PDP. The PDP typically deals with the transportation of goods while the DARP deals with passenger transportation (Parragh et al., 2008). Thus, the difference between DARP and PDP is usually expressed in terms of additional constraints or objectives that explicitly take user (in)convenience into account (e.g., time window and vehicle capacity constraints). The FMRSP can be seen as a special case of DARP where passengers travel to a common depot (station) and with additional service-specific constraints. Particularly, in the FMRSP we address in this paper, we focus on on-line dispatch and rebalancing decisions. That is, we consider the allocation of customers requests to vehicles as they arrive, and while vehicles are busy with other transportation requests. This entails dealing two types of customers. First, we find customers whose request has been accepted in previous decision epochs and have not yet picked up. These customers requests must be satisfied. Second, we find new customers whose request may or may not be accepted, similarly to a price-collecting TSP (Balas, 1989). Finally, empty vehicles may be moved to rebalancing centers in order to prepare for future demand.

Due to the fast development of GPS and widespread use of smartphones, ride-sharing services enabled by smartphones applications have attracted broad attentions. Commercial companies such as Uber and Didi have implemented versions of the service (Xu et al., 2018; Lin et al., 2018) and the attention of the research community has grown providing both optimization methods (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) and reinforcement learning methods (Xu et al., 2018; Lin et al., 2018; Li et al., 2019; Tang et al., 2019; Qin et al., 2019).

The FMRSP is, however, a relatively new problem and the corresponding literature is somewhat sparser. Yu Shen (2018) study the integration of a first-mile autonomous vehicle (AV) ride-sharing service into public transportation. The idea is to preserve high demand bus routes while using shared AVs as an alternative for low demand routes. They assess the performance of this system using simulation. Particularly, they test different fleet sizes, ride-sharing preferences and dispatching algorithms. The results show that the integrated system has the potential of enhancing service quality and utilizing bus services more efficiently. The article does not provide the mathematical model that drives the underlying planning decisions. Wen et al. (2018) propose a reinforcement learning method to adaptively move idle vehicles in a shared mobility-on-demand systems. They test their solution method on a first-mile service in the city of London. Results show that the proposed method outperforms the local anticipatory method by reducing the fleet size by 14%, and the computational speed is 2.5 times faster for close

to optimal solution. Chen et al. (2020) provide a mixed-integer linear programming (MILP) model to make AV dispatch and ride-sharing scheme decisions with the objective of minimizing operational costs. The authors devise a cluster-based solution method to deal with large-scale instances. Numerical experiments show that the proposed method can efficiently reduce the fleet size, travel miles and fulfill the requirement of online computation.

Rebalancing has become an important aspect of shared-mobility services and has been widely studied in recent years. Several recent papers show the benefit of rebalancing in the context of carsharing (Illgen and Höck, 2019) and ride-sharing (Wallar et al., 2018; Wen et al., 2018; Ma et al., 2019; Alonso-mora et al., 2018) problems. Particularly for ride-sharing, Wallar et al. (2018) propose algorithms for partitioning the operating area into zones, estimating the real-time demand and rebalancing idle vehicles. Wen et al. (2018) use reinforcement learning for predicting demand and rebalancing vehicles. Alonso-mora et al. (2018) design a matching algorithm for on-demand ride-sharing and incorporates rebalancing for remained idle vehicles after matching. Ma et al. (2019) propose a queueing-theoretic vehicle dispatch and idle vehicle relocation algorithm. Except for Alonso-mora et al. (2018), the above mentioned studies partition the operating area into zones and identify zones where to rebalance. In this paper, we propose a a clustering-based method which identifies relocation points by clustering demand. Furthermore, our work aims at assigning vehicles to customers and rebalancing vacant vehicles to rebalancing centers simutaneously, while the available literature considers assigning customers and rebalancing in a sequential order. To the best of our knowledge, this work is the first paper considering rebalancing and order dispatching simutaneously in first-mile ride-sharing services.

Our work is an extension of the methods provided in Chen et al. (2020) and Alonso-mora et al. (2018). The main differences between our work and Chen et al. (2020) is that our work combines rebalancing and order dispatching decisions, while Chen et al. (2020) focus only on order dispatching. In addition, Chen et al. (2020) solve the dispatching problem by working on groups of requests while we optimize considering individual requests. Finally, Chen et al. (2020) consider reservations made 15-60 min before departure, while we focus on online decisions which take into account new requests on a shorter notice and capture the real-time location and status of the vehicles. Alonso-mora et al. (2018) propose an efficient algorithm to generate optimal routes and match large groups of riders to vehicles in real-time and rebalance the idle vehicles to areas of high demand, and the algorithm is validated on the data provided by Commission and Limousine (2020). In this work we focus on modeling the first-mile ride-sharing problem mathematically. Furthermore, we address order dispatching and vehicle rebalancing decisions simultaneously, such that we move vehicles to high demand areas beforehand and avoid myopic decisions.

Summarizing, the contribution of this work can be stated as follows:

1. We provide a mathematical model for order dispatching and vehicle rebalancing in a first-mile ride-sharing service. The problem is formulated as a MILP.

2. We consider a rolling horizon optimization. This entails that a subset of the customers (those whose request has been accepted in previous decision problems) must be serviced,

while the remaining customers (those newly arrived) may be picked-up if feasible and profitable.

3. We propose a clustering method to identify the number and locations of rebalancing centers.

4. We test our model on randomly generated instances to assess the solutions delivered by the model and, particularly, the advantage provided by rebalancing activities.

3.3 The First-mile Ride-sharing Problem

In this section, we formally introduce the First-Mile Ride-Sharing Problem. We start, in Section 3.3.1, by providing a general introduction to the problem. Following, in Section 3.3.2, we introduce a mathematical model for the FMRSP, and in Section 3.3.3 we provide a simple example to illustrate possible solutions to the problem. An overview of notations can be found in Section 3.6.

3.3.1 Problem Statement

We consider the operator of a fleet of vehicles $\mathcal{K} := \{1, \ldots, K\}$ 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". At each re-optimization phase, the available customers can be partitioned in two sets, namely $\mathcal{N}_P := \{1, \ldots, N_P\}$ which contains the customers whose transportation request had already been accepted during a previous optimization phase, and $\mathcal{N}_C := \{1, \ldots, N_C\}$ which contains newly arrived customer requests which have not been considered in previous optimization phases. We assume that the customers in $\mathcal{N}_C$ may be either accepted (and thus assigned to a vehicles) or rejected, while the customers in $\mathcal{N}_P$ must be picked up (thus we assume acceptance decisions are binding). For convenience we set $\mathcal{N}_U := \mathcal{N}_C \cup \mathcal{N}_P$.

All customers travel to a common destination d located in position $o(d)$ (e.g., a transit station) and for each customer i, the operator knows the requested pick-up time T_i^P, the requested arrival time T_i^A and the origin $o(i)$. We let Δ be the maximum waiting time (i.e., difference between actual pick-up time and requested pick-up time). Similarly, at the beginning of the re-optimization phase, denoted T, each vehicle k is located at $o(k)$ as a result of previous deployment 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 the passengers already on board vehicle k. The operator needs to ensure that vehicles with customers on board terminate their journey to the station. Conversely, vehicles with no customers on board may be sent to a rebalancing point or stay at their origin location $o(k)$. A set $\mathcal{R} := \{1, \ldots, R\}$ of potential rebalancing points in the operating

area is available. For each rebalancing point r we let B_r denote an upper bound on the number of vehicles that can be dispatched to the rebalancing point.

The operator bears transportation costs generated by vehicle movements. Particularly, we assume travel times are known, with T_{ij} being the travel time between locations $o(i)$ and $o(j)$ with $i \in \mathcal{K} \cup \mathcal{N}_U$, $j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}$ and cost C is born for each unit of travel time. The operator collects a revenue P_i when picking up customer i, for $i \in \mathcal{N}_C$. Note that the revenue is collected only when picking up new customers as we assume the revenue for the customers in $\mathcal{N}_P$ has been collected during previous optimization phases. Furthermore, E_r denotes the expected revenue collected for each vehicle relocated to rebalancing center $i \in \mathcal{R}$. Expected future revenues from rebalancing activities are discounted using a parameter β that denotes the weight of the rebalancing reward.

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 $i \in \{k\} \cup \mathcal{N}_U, j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K}$. Furthermore, we let t_k^A denote the actual arrival time of vehicle k to the station, for $k \in \mathcal{K}$ and t_i^P denote the actual pick-up time of customer i, for $i \in \mathcal{N}_U$.

Thus, we use a 3-index formulation of size $O(|\mathcal{N}_C||\mathcal{K}||\mathcal{R}|)$. The FMRSP is NP-hard, as it contains the prize-collecting TSP (Balas (1989)) as a special case.

3.3.2 Mathematical Model

Having defined all decision variables and parameters, we may formulate the problem as follows.

$$\max \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}_C} \sum_{j \in \mathcal{N}_U \cup \{d\}} P_i x_{ij}^k - \sum_{i \in \{k\} \cup \mathcal{N}_U} \sum_{j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}} \sum_{k \in \mathcal{K}} C T_{ij} x_{ij}^k + \beta \sum_{i \in \mathcal{R}} \sum_{k \in \mathcal{K}} x_{ki}^k E_i \tag{3.1a}$$

$$\text{s.t.} \sum_{j \in \mathcal{N}_U \cup \{d\}} \sum_{k \in \mathcal{K}} x_{ij}^k \leq 1 \qquad \forall i \in \mathcal{N}_C \quad (3.1b)$$

$$\sum_{j \in \mathcal{N}_U \cup \{d\}} \sum_{k \in \mathcal{K}} x_{ij}^k = 1 \qquad \forall i \in \mathcal{N}_P \quad (3.1c)$$

$$\sum_{i \in \mathcal{N}_U \cup \mathcal{K}} x_{id}^k \leq 1 \qquad \forall k \in \mathcal{K} \quad (3.1d)$$

$$\sum_{i \in \mathcal{N}_U \cup \{k\}} x_{ij}^k = \sum_{i \in \mathcal{N}_U \cup \{d\}} x_{ji}^k \qquad \forall j \in \mathcal{N}_U, k \in \mathcal{K} \quad (3.1e)$$

$$\sum_{j \in \mathcal{R} \cup \mathcal{N}_U \cup \{d\}} x_{kj}^k = \sum_{j \in \mathcal{N}_U \cup \{k\}} \sum_{i \in \mathcal{R} \cup \{d\}} x_{ji}^k \qquad \forall k \in \mathcal{K} \quad (3.1f)$$

$$\sum_{i \in \{k\} \cup \mathcal{N}_U} \sum_{j \in \mathcal{N}_U} x_{ij}^k + V_k \leq Q \qquad \forall k \in \mathcal{K} \quad (3.1g)$$

$$\sum_{k \in \mathcal{K}} x_{kj}^k \leq B_j \qquad \forall j \in \mathcal{R} \quad (3.1h)$$

$$V_k \leq Q(1 - \sum_{j \in \mathcal{R}} x_{kj}^k) \qquad \forall k \in \mathcal{K} \quad (3.1i)$$

$$V_k \leq Q \sum_{j \in \mathcal{N}_U \cup \{d\}} x_{kj}^k \qquad \forall k \in \mathcal{K} \quad (3.1j)$$

$$t_i^P + T_{ij} \leq t_j^P + T^L(1 - \sum_{k \in \mathcal{K}} x_{ij}^k) \qquad \forall i, j \in \mathcal{N}_U \quad (3.1k)$$

$$T + T_{kj} \leq t_j^P + T^L(1 - x_{kj}^k) \qquad \forall j \in \mathcal{N}_U, k \in \mathcal{K} \quad (3.1l)$$

$$t_i^P - T_i^P \leq \Delta \qquad \forall i, j \in \mathcal{N}_U \quad (3.1m)$$

$$t_k^A \leq T_i^A + T^L(1 - \sum_{j \in \mathcal{N}_U \cup \{k\}} x_{ji}^k) \qquad \forall i \in \mathcal{N}_U, k \in \mathcal{K} \quad (3.1n)$$

$$t_k^A \leq T_k \qquad \forall k \in \mathcal{K} \quad (3.1o)$$

$$t_j^P + T_{jd} x_{jd}^k \leq t_k^A + T^L(1 - x_{jd}^k) \qquad \forall j \in \mathcal{N}_U, k \in \mathcal{K} \quad (3.1p)$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i \in \{k\} \cup \mathcal{N}_U, j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K} \quad (3.1q)$$

$$t_k^A \in \mathbb{R}^+ \qquad \forall k \in \mathcal{K} \quad (3.1r)$$

$$t_i^P \in \mathbb{R}^+ \qquad \forall i \in \mathcal{N}_U \quad (3.1s)$$

Objective function (3.1a) represents the profit for the operator. The first term represents the revenue generated by picking up customers, the second term the total cost born of the vehicles movements and, finally, the third term is the discounted expected profit obtained in the rebalancing centers.

Constraints (3.1b) and (3.1c) state that new customers may be picked up at most once and customers already accepted must be picked up exactly once, respectively. Observe, in (3.1b) and (3.1c), 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. Constraints (3.1d) ensure that vehicles travel to the station at most once. Constraints (3.1e) 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 arrive at a customer location j either from another customer or from the vehicle's original location $o(k)$. Constraints (3.1f) 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 (3.1g) ensure that the capacity of the vehicles is not exceeded, while constraints (3.1h) 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 (3.1i) state that only empty vehicles may be dispatched to rebalancing centers. For instance, if vehicle k is dispached 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 $\mathcal{N}_U$ and rebalancing points are automatically forbidden by the absence of the corresponding x_{ij}^k variables. Constraints (3.1j) 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 (3.1k) state that if customer j is picked up by vehicle k immediately after picking up customer i, then the actual picking 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^L: $= max_i\{T_i^A\}$ for $i \in \mathcal{N}_U$ is an upper bound on the requested arrival time. Similarly, constraints (3.1l) denote the pick-up time for the first customers in the route. Constraints (3.1m) ensure that the difference between the actual pick up time and the requested pick up time of the customer does not exceed the maximum waiting time Δ. Constraints (3.1n) ensure that the actual arrival time of vehicle k must be 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^A enforcing that the actual arrival time of vehicle k is before T_i^A. Constraints (3.1o) 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 (3.1p) 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_{jd}^k takes value 1, the left-hand-side becomes $t_j^P + T_{jd}$, 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^P, the right-hand-side becomes $t_k^A + T^L$, which always holds.

Finally, constraints (3.1q)-(3.1s) define the domain of the decision variables.

3.3.3 Illustrative Example

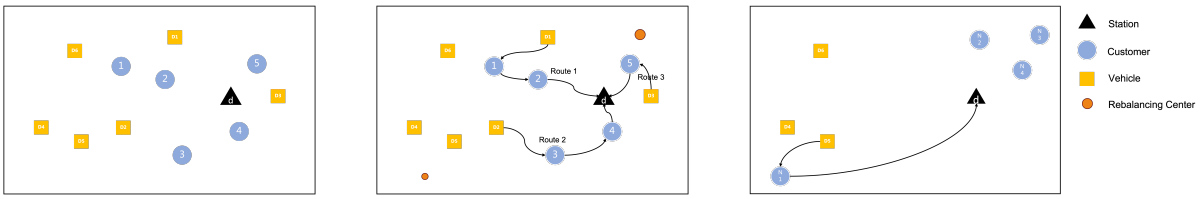
In what follows we provide two examples to illustrate how the model in Section 3.3.2 works. The online FMRSP model re-optimizes the vehicles dispatch and rebalancing decisions at fixed time intervals. Assume that the current re-optimization phase start at $T = T_1$, and a snapshot of the system, which includes vehicles and customers positions, is shown in Figures 3.3a and 3.4a. The blue circles denote customers while the yellow squares denote vehicles. Assume all vehicles

have a capacity $Q = 2$, and that are all empty except for $D3$ that has currently one customer on board ($V_{D3} = 1$). The triangle denotes the station. Particularly, Figure 3.3 describes a scenario without rebalancing, while Figure 3.4 describes a scenario where rebalancing is permitted.

The solution for the case without rebalancing, is illustrated in Figure 3.3b where the orange circles denote rebalancing centers and their size reflects the expected demand at their location. The solution for the scenario with rebalancing is provided in Figure 3.4b. Consider the solution to the first scenario. We observe that the model provides three routes and that the rebalancing centers are not visited. The route for vehicle $D1$ (Route 1) starts from the current location of the vehicle, $o(D1)$, and visits customer 1 (thus $x^{D1}_{D1,1} = 1$) and customer 2 ($x^{D1}_{1,2} = 1$), in this order, before arriving at the station ($x^{D1}_{2,d} = 1$). In this solution, vehicles $D4$, $D5$ and $D6$ stay idle at their current position and wait for the next re-optimization phase.

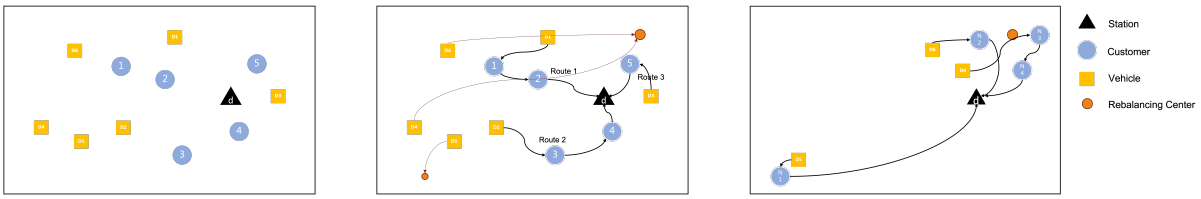
Figure 3.3c depicts a new snapshot of the system at the new re-optimization phase (say $T = T_2$), and the corresponding solution. Customers 1 through 5 from the previous optimization phase are not in the system anymore because they have been by the time picked up. Similarly, vehicles $D1$, $D2$ and $D3$ are not in the system because they are currently on the way to the station and have no extra capacity ($D1$ and $D2$ have filled their two seats and $D3$ has the remaining seat available). New customers $N1$-$N4$ appear in the system. The solution suggests picking up only customer $N1$. The remaining customers are, in fact, too distant from the available vehicles.

The solution for the scenario with rebalancing (Figure 3.4b) differs only in the fact that vehicles $D4$, $D5$, and $D6$ are dispatched to rebalancing centers. Therefore, at the next re-optimization phase (Figure 3.4c), $D4$ and $D6$ are much closer to the new requests $N2$, $N3$, and $N4$, and can pick them up ensuring their requested arrival time is satisfied.



(a) The system at $T = T_1$ (b) Solution for $T = T_1$ (c) Solution for $T = T_2$

Figure 3.3: Example problem in a scenario where rebalancing in not permitted



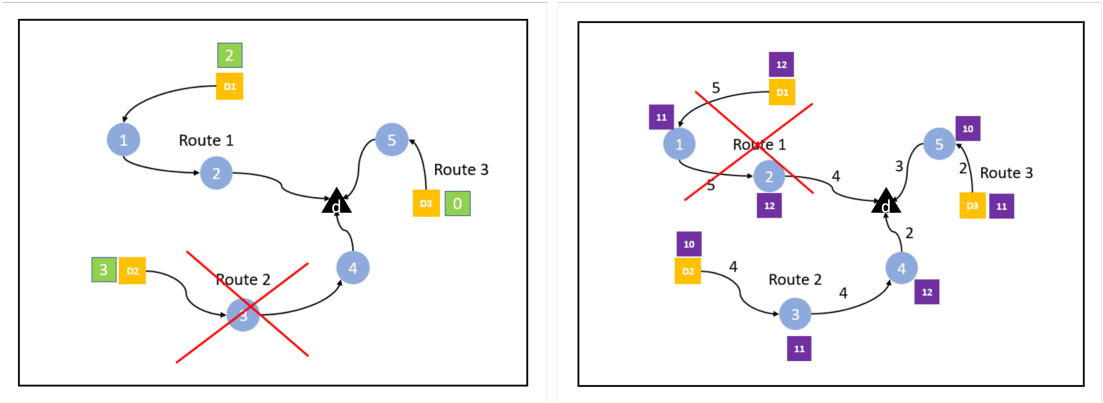
(a) The system at $T = T_1$ (b) Solution for $T = T_1$ (c) Solution for $T = T_2$

Figure 3.4: Example problem in a scenario where rebalancing is permitted

The example in Figure 3.5a illustrates how capacity constraints work. The original number of customers on board are shown in green boxes next to the vehicle. Assume the capacity of each vehicle is $Q = 4$. We observe that routes 1 and 3 are feasible with respect to the capacity

constraints as in both cases the number of customers on board does not exceed Q. Route 2, on the other hand, will be infeasible, since there are already 3 customers on board vehicle $D2$ at the beginning of the optimization phase, and the route assigns two additional customers to the vehicle, thus violating the capacity constraints.

Figure 3.5b illustrates how requested arrival time constraints are enforced. Assume the re-optimization phase starts at time $T = 0$. The requested arrival time of the vehicles and customers are shown in purple boxes while travel times are shown on top of the arcs. For route 1, the total travel time is $5 + 5 + 4 = 14$, which violates the requested arrival time of the passengers already on board of $D1$, which is 12. Thus the route is infeasible, even if the arrival times of the two customers in route 1 are respected. Routes 2 and 3 are instead feasible as the arrival times of both customers and vehicles are respected.



(a) Capacity constraints (b) Arrival time Constraints

Figure 3.5: Illustration of capacity and arrival time constraints

3.4 Finding Rebalancing Centers

In this section, we introduce a clustering-based methods for identifying rebalancing centers. We refer to the method as the *K-means Clustering* (KC) method. The method identifies both the location and demand of the rebalancing centers, and this in turn allows us to determine the upper bound on the number of vehicles dispatchable to the different rebalancing centers.

Given a number k of rebalancing centers to find, the KC method finds rebalancing centers by partitioning all requests received in the current re-optimization phase into $k = |\mathcal{R}|$ clusters. Clusters are created in such a way as to minimize the total distance between the points allocated to the cluster and the centroid of the cluster. The centroids of the clusters will then be used as rebalancing centers. The expected demand (number of requests) of the rebalancing centers will be set equal to the number of requests in the corresponding cluster. We let D_r be the demand of rebalancing point $r \in \mathcal{R}$.

The KC method is compared to a random selection method (hereafter named RS method) which consists of randomly selecting $|\mathcal{R}|$ points in the operating area as rebalancing centers. To each rebalancing center is assigned a demand equal to the number of customer requests of current re-optimization phase within a certain distance (e.g., 1 Km) of the rebalancing center.

For both the KC and RS methods, the upper bound B_r of the number vehicles that can be dispatched to each rebalancing center r is defined as $B_r = D_r/\bar{Q}$, where $\bar{Q}$ denotes the average number of customer on board a vehicle during one trip.

3.5 Numerical Experiments

In this section, we report the results of our numerical experiments. The scope of the experiments is to assess, in terms of profits and service rates, two different configurations of the service which we refer to as *without rebalancing* (`woR`) and *with rebalancing* (`wR`). The configuration `woR` refers to the situation where the service provider dispatches the vehicles only based on customer requests of the current re-optimization phase. For the model without rebalancing, we simplify our model in Section 3.3 by having an empty set of rebalancing centers. In the configuration `wR`, the service provider makes the dispatching decision based both on current customer requests and on predicted demand using rebalancing centers. In this case, we test the model with two different method for finding rebalancing centers. In the first case, which we refer to as `wRKC`, the company use KC to obtain the location and demand of the rebalancing centers. In the second case, which we refer to as `wRRS`, the company uses the RS method to find the locations and demands of the rebalancing centers.

The different configurations are tested on a set of random instances introduced in Section 3.5.2. All problems are solved with the Python libraries of GUROBI 9.5.0 and a server equipped with Intel Core i5 and 16GB of RAM.

3.5.1 Simulation Framework

We test our model in a simulation framework, with a planning horizon of one hour. We assume online re-optimization happens every 5 minutes. This means that every simulation requires the solution of 12 optimization problems (Model 3.1). At each re-optimization we update the status of the system, and randomly generate (as explained in the next section) a number of new customers. Particularly,

- At the initial optimization phase, say $T = 0$, we assume that the $\mathcal{N}_P$ is empty. This means that there is no customer whose request had already been accepted in a previous optimization phase. We generate a number of new customers $\mathcal{N}_C$, rebalancing centers $\mathcal{R}$, and initial vehicle positions (as explained in the next section) and solve the resulting model (3.1).
- We then step five minutes forward in time, say optimization phase $T = 1$, and assume the solution to the model for $T = 0$, has been implemented. This entails the vehicle followed the routes determined by the previous optimization model for five minutes, moving either to customers locations or to rebalancing centers. This provides their updated location for the new re-optimization phase. We then partition the customers of the previous optimization phase into three groups:
    1. The first group contains those customers that had been assigned to a vehicle but have

not yet been picked up in the five minutes interval between the two re-optimizations (i.e., the route of the vehicle assigned to the customer did not stop by the customer within the five-minute interval between re-optimizations). These customers form the set of mandatory customers $\mathcal{N}_P$ in the new re-optimization phase and must be picked up by some vehicle (possibly different from the one assigned in the previous re-optimization phase).

2. The second group contains those customers that had been assigned to a vehicle in the previous re-optimization phase and the route of the assigned vehicle stopped by the customer within the five-minute interval between re-optimizations. In the new re-optimization phase these customers represents occupied seats (V_k) in the vehicles to which they were assigned. Therefore, these customers represent fulfilled requests and do not appear in $\mathcal{N}_P$ in the new re-optimization phase.

3. The third group contains those customers that had not been assigned to a vehicle in the previous re-optimization phase. These represent customers whose request has been rejected and will not show up in the new re-optimization phase.

Observe, that in the new re-optimization phase, vehicles may find themselves into one of the following situations. (A) The vehicle is empty at a given position and was on the way to pick-up customers or to a rebalancing centers. (B) The vehicle has passengers on board at a given position and was on the way to pick-up additional customers or to the station. In case (A), in the new re-optimization phase the vehicles may be assigned to new customers or to a rebalancing center, independently of the decision made in the previous re-optimization phase. That is, it is possible that the vehicle is assigned to a set of customers different from the ones previously assigned to the vehicle. In case (B) the vehicle cannot be sent to a rebalancing center but may be assigned to new customers or sent directly to the station. Following, we generate new customers $\mathcal{N}_C$ for the new re-optimization phase and resolve a problem (3.1).

The procedure continues stepping five minutes forward in time until the end of the one-hour planning horizon. Thus, we are able to collect statistics on the performance of the service. Particularly, the profit is computed as follows. At the end of each re-optimization phase, we collect the fee for all the customers that have been accepted (i.e., a vehicle has been assigned to them) and picked up (i.e., the vehicle has arrived at their location during the five-minute interval between re-optimizations) and subtract the cost of the movements the vehicles have done during the five-minute interval between re-optimizations. The final total profit is then the sum of the individual profits made during the one-hour planning horizon. The procedure is explained by the following example.

- Assume that at $T = 0$, $\mathcal{N}_P$ is empty and $V_k = 0$ for all vehicles $k \in \{D_1, D_2, D_3\}$. That is, there is no customer whose request had already been accepted in a previous optimization phase. We generate a number (say four) of new customers $\mathcal{N}_C := \{C_1^0, C_2^0, C_3^0, C_4^0\}$, rebalancing centers $\mathcal{R} := R_1^0$, and initial vehicle positions $o(D_1), o(D_2), o(D_3)$ and solve the resulting model (3.1). Assume that the solution determines the following routes for the

three vehicles: $\texttt{Route1}^0 := \{D_1, C_1^0, C_3^0, d\}$, $\texttt{Route2}^0 =: \{D_2.C_2^0, d\}$, $\texttt{Route3}^0 := \{D_3, R_1^0\}$ and that customer C_4^0 is rejected.

- The solution computed at $T = 0$ is implemented and the vehicle follow the respective routes for five minutes. This provides updated system information for optimization phase $T = 1$. That is, we updated the location of the vehicles $\{o(D_1), o(D_2), o(D_3)\}$. We observe that,

  1. For $Route1^0$, the vehicle D_1 already picked up customer C_1^0, and is still on the way to pickup C_3^0, so we can delete C_1^0 from the system, update $V_{D_1} = 1$ and move C_3^0 to mandatory customer set $\mathcal{N}_P := \{C_3^0\}$.

  2. For $Route2^0$, vehicle D_2 already picked up customer C_2^0 and is still on the way to station, so we can delete C_2^0 from the system and update $V_{D_2} = 1$.

  3. For $Route3^0$, vehicle D_3 arrived at the rebalancing center R_1^0, thus $V_{D_3} = 0$.

  In the new optimization phase $T = 1$, since the vehicles D_1, D_2 already have customers on board, they cannot be sent to a rebalancing center but may be assigned to new customers or sent directly to the station. However, vehicle D_3 may be assigned to new customers or to a rebalancing center as it is still empty. Following, we generate new customers $\mathcal{N}_C := C_1^1, C_2^1, C_3^1, C_4^1$ for the new re-optimization phase $T = 1$ and resolve the problem (3.1).

3.5.2 Instance Generation

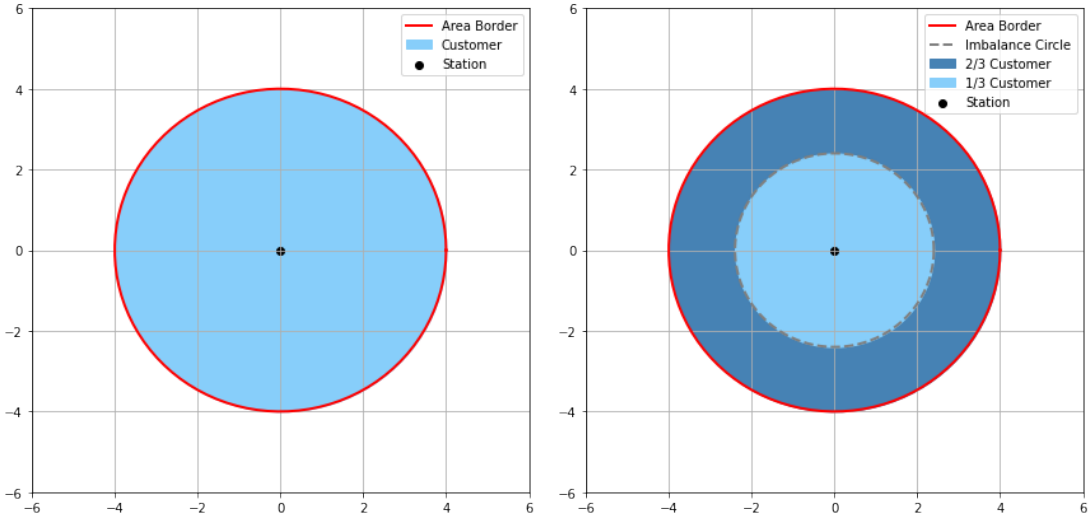
We generate a number of artificial and randomly generated instances that mimic real-life operating scenarios for the service. Particularly, we assume a fleet of homogeneous vehicles of capacity $Q = 4$. The position of the vehicles for the first re-optimization phase is generated randomly in the business area (defined below) and initially vehicles are assumed to have no passenger on board. For the re-optimization phases other than the first, the position of the vehicles, and the number of passengers on board is computed as the result of previous optimization phases.

We consider two different geographies of the business area. In the first geography, depicted in Figure 3.6, the station is located at the center of the business area, and the business area itself is represented by a circle of radius $R = 4Km$. In the second geography, see Figure 3.7, the station is located in a corner and the business area is a quarter of a circle of radius of $R = 8Km$. The second geography is meant to represent urban contexts where the demand is concentrated only on one side of the station due to, e.g., physical barriers such as rivers or harbors.

For each geography, and for each re-optimization phase, customer request are generated in the following two different scenarios. In the first scenario, referred to as the *balanced scenario*, pickup locations are randomly scattered in the whole business area, see Figure 3.6a and Figure 3.7a. In the second scenario, referred to as the *imbalanced scenario*, one third of the requests arrives, randomly, from inside the inner circle of radius $R^I = 0.6R$ (where R is the radius of the outer circle), while the remaining requests arrive, randomly, from outer portion of the circle, see Figure 3.6b and Figure 3.7b. We obtain, in total, four configurations namely

1. station in the center and balanced demand (`BCt`),
2. station in the center and imbalanced demand (`iBCt`),
3. station in the corner and balanced demand (`BCn`),
4. station in the corner and imbalanced demand (`iBCn`).

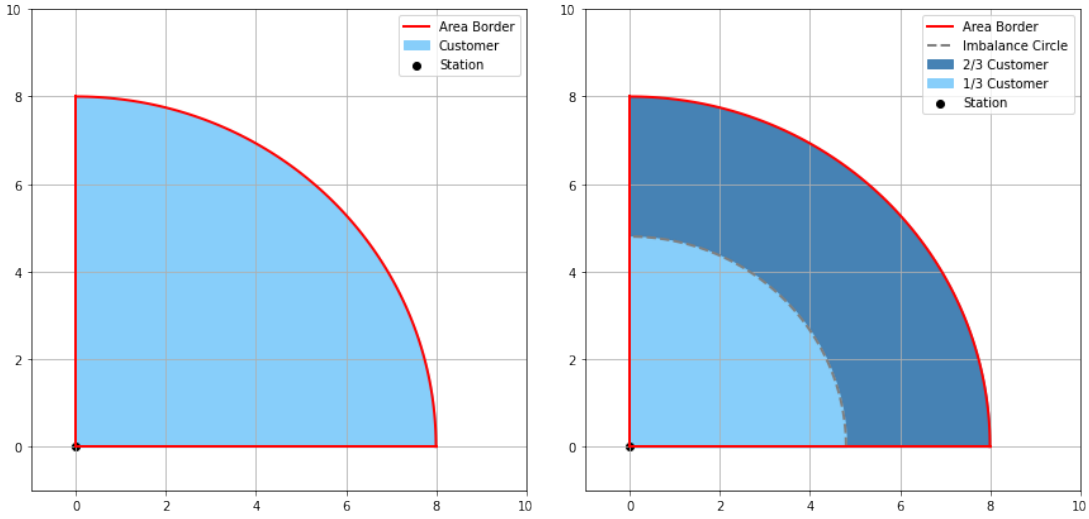
For each request, the requested pickup time (T_i^P) is randomly generated uniformly between 0 and 3 minutes after the beginning of the planning horizon, and the requested arrival time (T_i^A) is set as the sum of requested pickup time, T_i^P, travel time between customer i and the station d, and a buffer time randomly generated between 5 and 8 minutes. Travel time T_{ij} are calculated using Euclidean distances and assuming an average speed of 36 Km/h (Commission and Limousine, 2020). The unit transportation cost C is set to \$11.25/h (English, 2008) and trip revenues P_i are computed using a fare of \$2.59 per Km traveled plus \$0.74 per minute traveled, with a minimum fare of \$8 following the setting in INSHUR.



(a) Balanced distribution of customers (b) Imbalanced distribution of customers

Figure 3.6: Balanced and imbalanced distribution of customers, station located in the center

We set the value of β to 0.1 in the objective function, unless otherwise specified. Observe that β determines the impact of rebalancing movements. High values will increase the potential benefit of rebalancing and might lead to rejecting current customers while low values might lead the model to provide myopic solutions. The impact of different values of β will be assessed through sensitivity analysis in Section 3.5.4. The net profit of rebalancing center E_i is calculated as $\bar{P}_i - CT_{id}$, where $\bar{P}_i$ is the revenue of dispatching a vehicle to rebalancing center $i \in \mathcal{R}$, and is calculated as $\bar{P}_i = Q\ P_i^A$, where P_i^A the average revenue for the requests in the cluster where i is the centroid and Q is the capacity of the vehicles. C is the unit transportation cost, set as above, and T_{id} is the distance between rebalancing point i and the station. To obtain the upper bound on the number of vehicles disptachable to a rebalancing center (B_r, see Section 3.4) we set the average number of customers on board during one trip of a vehicle $\bar{Q}$ equal to half of the capacity $Q = 4$.

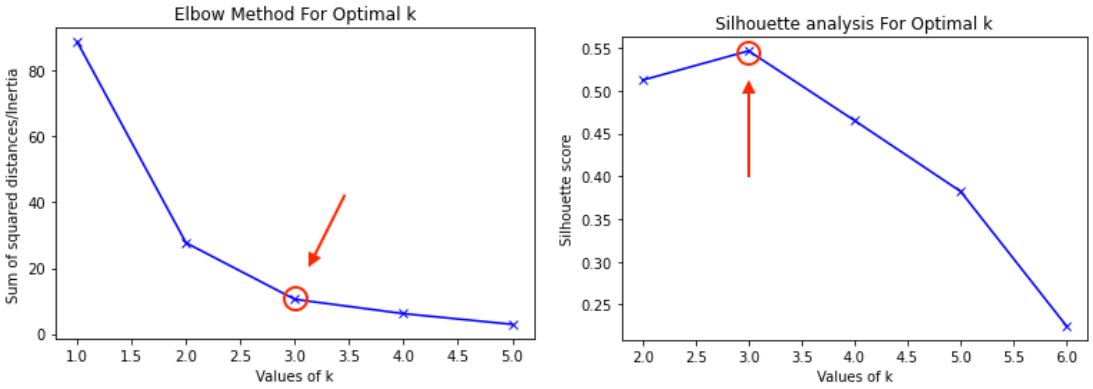


(a) Balance Corner (b) Imbalance Corner

Figure 3.7: Balanced and imbalanced distribution of customers, station located in a corner

For each configuration, we generate different instances varying in the number of vehicles and customers that appear at each new re-optimization phase. Particularly, we create instance classes named $C|\mathcal{N}_C|V|\mathcal{K}|$ with number of customers $|\mathcal{N}_C| \in \{6, 7, 8\}$, and number of vehicles, $|\mathcal{K}| \in \{10, 12, 14\}$. As an example, $C8V10$ indicates a class of instances with 8 new customers in each re-optimization phase and 10 vehicles available for dispatching for the whole planning period. For each instance class we randomly generate 3 different instances. Observe, however, that for each instance we solve 12 different optimization problems in our simulation framework.

We set the number of rebalancing centers $|\mathcal{R}|$ to 3 in all instances. This number is found using the *Elbow Method* (EM) and the *Silhouette Analysis Method* (SAM) (Mahendru, 2019). Particularly, we use the instances with $|\mathcal{N}_C| = 8$ as a reference case to find a suitable value for $k = |\mathcal{R}|$. First we randomly generate 8 customers in the operating area, then we use the EM approach to show the performance of the KC method for different values of k. For each k we consider the *Within-Cluster-Sum of Squared Errors* (WSS). We then plot the WSS versus k, and choose the value of k for which the WSS flattens. This point is referred to as an "Elbow", see Figure 3.8a.



(a) Elbow Method (b) Silhouette Analysis

Figure 3.8: Identifying a suitable number of clusters

Nevertheless, in some cases, the EM does not give precise answers. In such cases, we will use the SAM to make a decision. The silhouette score is a measure of how similar a customer request is to its own rebalancing cluster compared to other rebalancing clusters. To be more precise, the silhouette score of one customer request i can be calculate as below:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3.2}$$

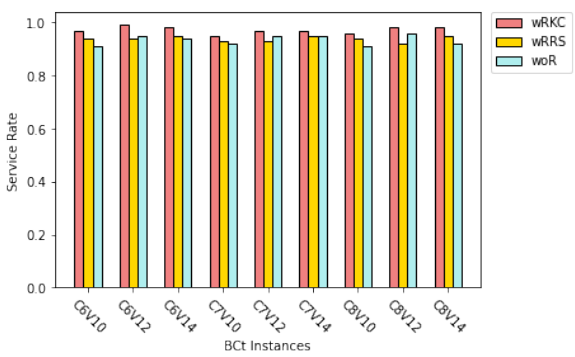
where $b(i)$ is the average of the minimum euclidean distance between customer i and the customers in clusters other than the one customer i belongs to. Parameter $a(i)$ is the average euclidean distance from customer i to other customer requests inside customer i's own rebalancing cluster. For each k, we sum $s(i)$ for all customer requests, and we plot it against k, see a qualitative description in Figure 3.14. We then choose the value of k for which the sum is the highest (the higher $s(i)$ the higher is the difference between a point i and the clusters other than the one it belongs to). In our case, the best value of k was found to be $k = 3$ and therefore we use this value in the computational study.

3.5.3 Managerial Insights

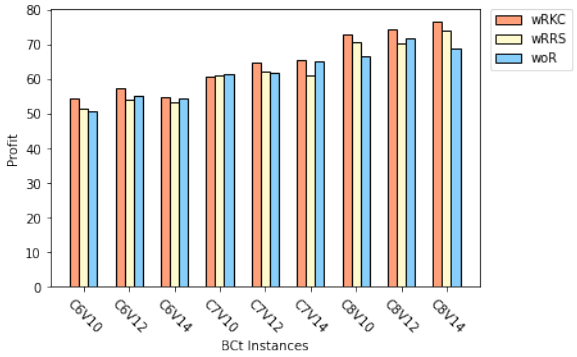
In this subsection, we assess the solutions provided by the model in terms of service rates and profits. The service rate is computed as the ratio between the total number of customers transported during the whole planning period over the total number of requests received in the same period. The profit consists of the cumulative profit over the entire simulated period (thus one hours with re-optimization every five minutes). Particularly, we compare three different strategies, namely no rebalancing (`woR`), rebalancing to random rebalancing centers (`wRRS`), and rebalancing to rebalancing centers found with the KC method(`wRKC`), see Section 3.4. These three strategies are assessed on different configurations of the service, namely `BCt`, `iBCt`, `BCn`, `iBCn`, see Section 3.5.2.

Figures 3.9 to 3.12 report the profit and service rate for the different strategies and configurations of the service. We notice, in Figure 3.9a and Figure 3.10a that the settings `wRKC`, `wRRS` and `woR` provide a 90% or higher service rate in the configurations `BCt` and `iBCt`, respectively. The profits are likewise relatively similar as shown in Figure 3.9b and Figure 3.10b. However, we observe that both service rate and profits in the `wRKC` case are systematically higher than in the case `woR`, illustrating that rebalancing activities pay off. The figures also illustrate that rebalancing with the clustering method (`wRKC`) is consistently preferable to rebalancing to random locations (`wRRS`) and that rebalancing to random locations is not necessarily better than having no rebalancing activities (`woR`). In general, when the station is in the center of the business area, we do not observe significant changes between the case with balanced demand (`BCt`) and imbalanced demand (`iBCt`).

Figures 3.11 and 3.12 report the results for the configuration with the station in a corner with balanced (`BCn`) and unbalanced (`iBCn`) demand, respectively. In these cases, we observe a much more marked difference in performance between the setting with and without rebalancing. Particularly, we notice that the setting `wRKC` improves both profits and service rates by approx-

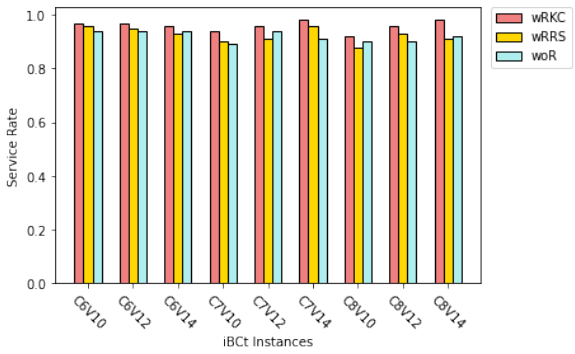


(a) Service Rate

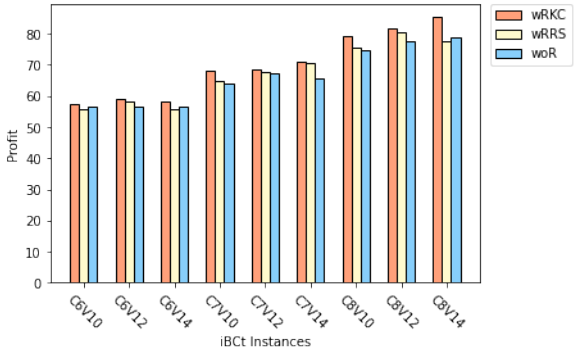


(b) Profit

Figure 3.9: Service rate and profit for the `BCt` configuration



(a) Service Rate



(b) Profit

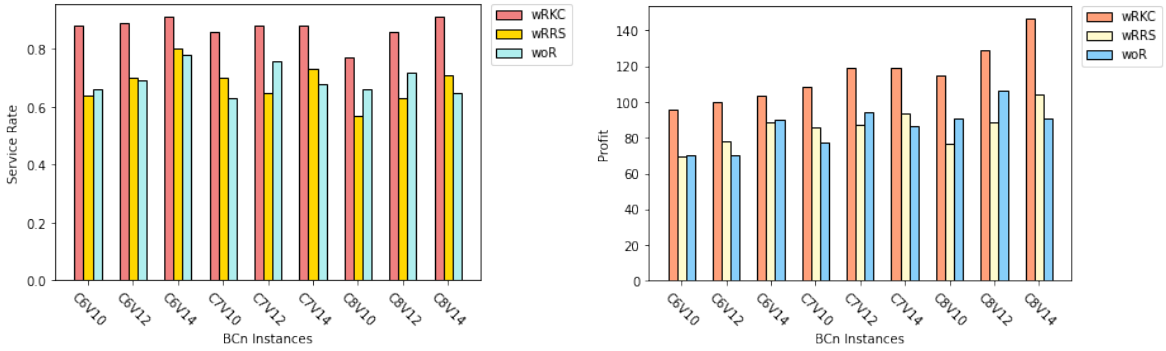
Figure 3.10: Service rate and profit for the `iBCt` configuration

imately 30%, compared with `woR` in the case `BCn`. In the configuration `iBCn` the improvement is even more marked with approximately 50% higher in service rates and 60% higher profits. Random rebalancing (`wRRS`) is still slighly preferable to no rebalancing (`woR`) but lags significantly behind compared to the strategy of using clustering-based rebalancing centers (`wRKC`).

It can be further observed in Figure 3.11 that, as the number of customers increases, the gap between `wRKC` and `woR` increases even with the same fleet size. In Figure 3.12 we observe that, in the `wRKC` case, both profits and service rates improve as the number of vehicles increases, keeping fixed the number of customers. Compared to Figure 3.11 we notice that for the `woR` case, the highest service rate for the *C8V** instances is lower than 60%, while the lowest service rate the same instances in the `BCn` scenario is above 60%.

In conclusion, `wRKC` outperform both `woR` and `wRRS` in all scenarios. When the station is in the center (`BCt` and `iBCt`), the model reaches a service rate higher than 90% regardless of whether and how rebalancing is done. Nevertheless, profits and service rates are consistently higher when rebalancing centers are chosen with the clustering method (`wRKC`). When the station is in a corner(`BCn` and `iBCn`), rebalancing to centers chosen with the clustering method (`wRKC`) yields a much more marked improvement both in profits and service rate, especially with the demand is not evenly spread in the service area(`iBCn`). Furthermore, as we observe in this case, keeping fixed number of customers, the profit increases with the size of the fleet. As more vehicles

provide better opportunities for rebalancing and anticipating future requests. This also shows that rebalancing to appropriately chosen locations (e.g., `wRKC`) can improve the utilization of empty vehicles.



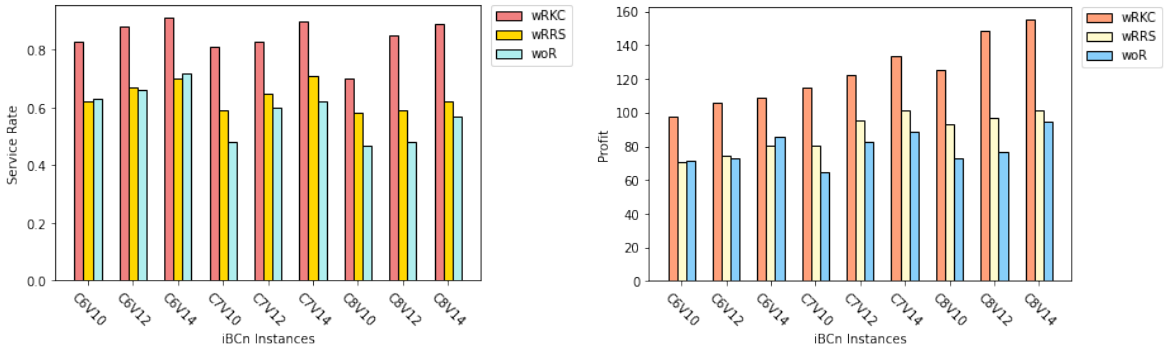
(a) Service Rate (b) Profit

Figure 3.11: Service rate and profit for `BCn` instances

3.5.4 Sensitivity Analysis

We now analyze how the efficiency of rebalancing is affected by the β parameter, see (3.1). With the other parameters unchanged, for each instance we generate nine variants with different values of β, namely 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9. We conduct experiments on all instances assuming an imbalanced configurations (`iBCn` and `iBCt`) to assess the impact on profits of the different rebalancing strategies under different weights β.

Figure 3.13 illustrates the results for the `iBCn` configuration. We observe that the performance of `woR`, `wRRS`and `wRKC` are stable under different setting of β. Similarly, Figure 3.14 reports the results under the `iBCt` configuration. Here we observe that when the ratio of customers to vehicles is relatively high, i.e. *C6V10*, *C7V10*, *C8V10* and *C8V12*, the performance of the `wRKC` is sensitive to β. Particularly, for instances *C8V10* and *C8V12*, when β increases, the performance of `wRKC` can drop below that of `wRRS` or even below `woR` while the `woR` and `wRRS`



(a) Service Rate (b) Profit

Figure 3.12: Service rate and profit for `iBCn` instances

are relatively stable to the change of β. When the customers to vehicles ratio is relatively low, the results appear insesitive to β.

Figure 3.15 reports the proportion of vehicles going to rebalancing centers under the configuration `iBCt` for instances *C8V10* and *C8V14*. The proportion is computed by the ratio between the average number of vehicles going to rebalancing centers of 3 randomly generated instances in the 12 re-optimization phases and the total number of vehicles. When β increases, the proportion of dispatched vehicles increases significantly for instance *C8V10*. The proportion of vehicles going to rebalancing centers is relatively stable for *C8V14*, which is consistent with the results in Figure 3.14.

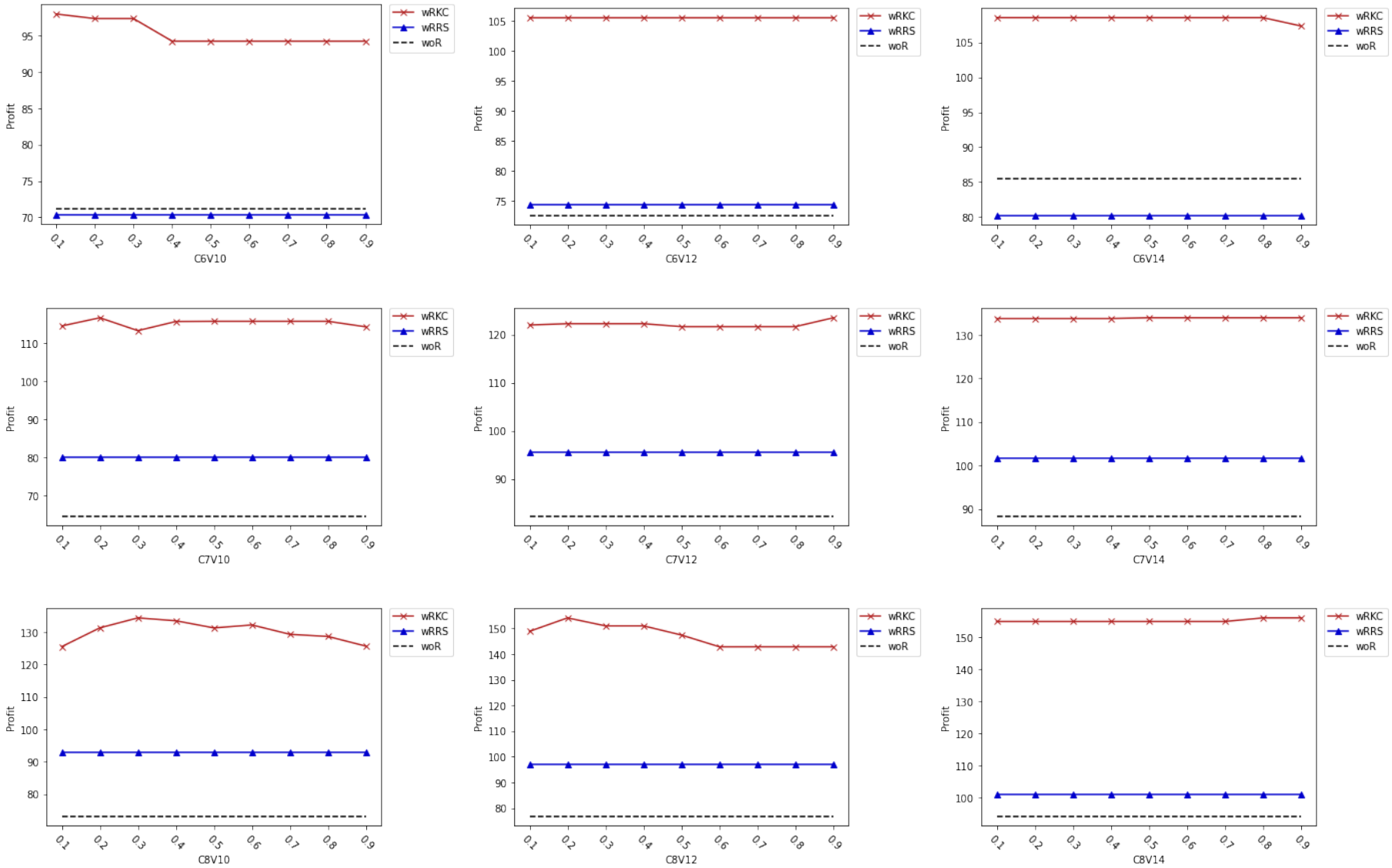


Figure 3.13: Sensitivity analysis for the `iBCn` configuration

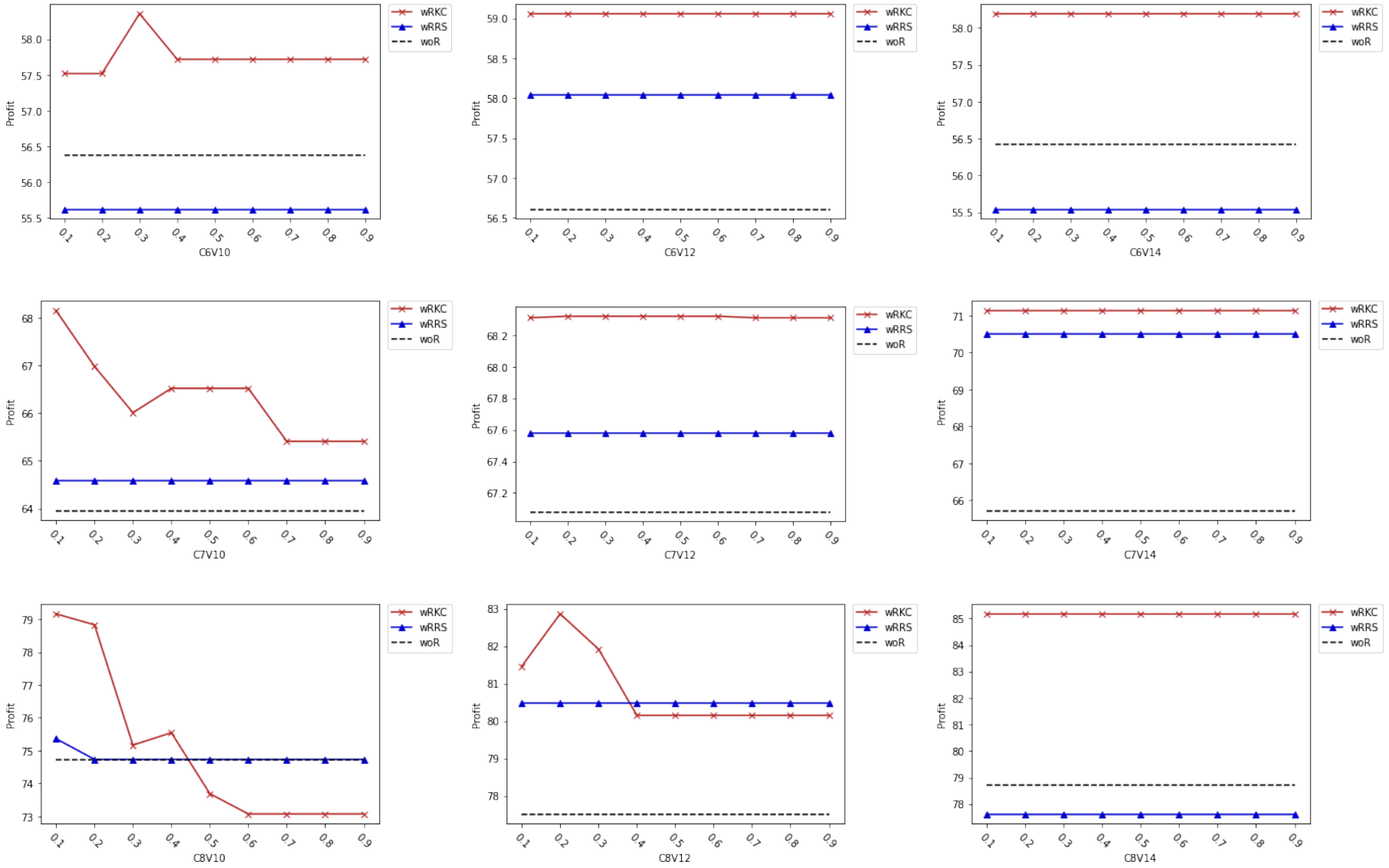


Figure 3.14: Sensitivity analysis for the `iBCt` configuration

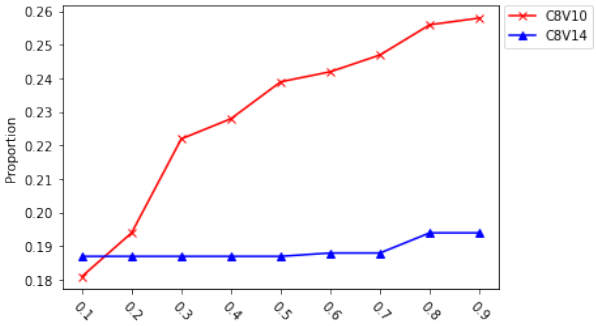
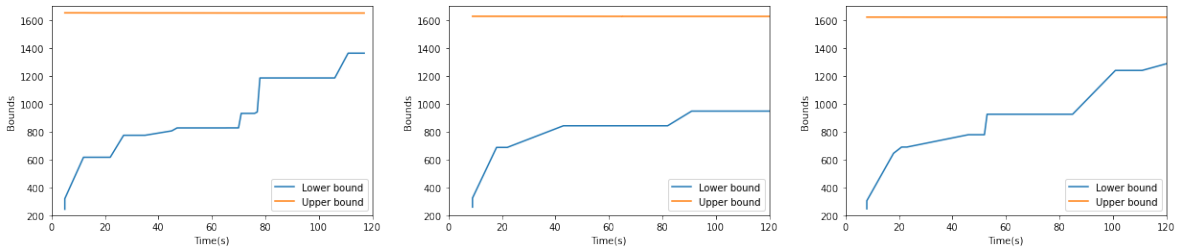


Figure 3.15: Proportion of vehicles dispatched to rebalancing centers

3.5.5 Complexity of the models

For the numerical experiments presented above, we choose to use relatively small instances in order to obtain results that are not affected by optimality gaps. All the instances presented could be solved, in all re-optimization phases, to provable optimality within less than a second in average (45 seconds in the worst case).

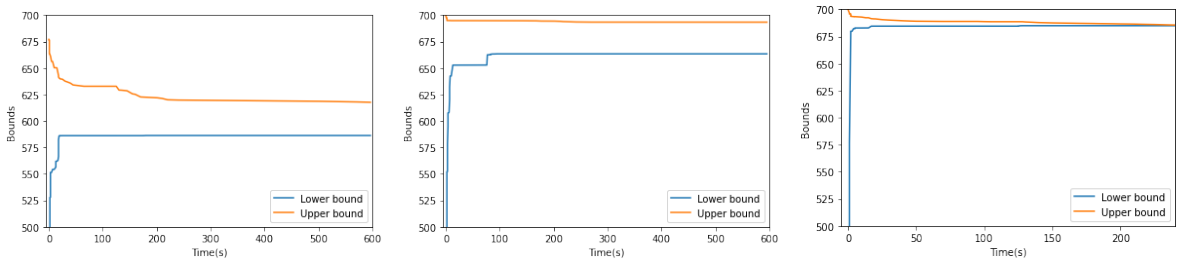
Nevertheless, in practical situations one may encounter instances significantly larger than the ones we used. In this section, we assess how solution times and optimality gaps scale with the size of the instance. Particularly, we create instance classes named $C|\mathcal{N}_C|V|\mathcal{K}|$ for $|\mathcal{N}_C| \in \{60\}$, and $|\mathcal{K}| \in \{40, 50, 60\}$. As an example, *C60V40* indicates a class of instances with 60 new customers in each re-optimization phase and 40 vehicles available for dispatch for the whole planning period. Figure 3.16 reports the progression of upper and lower bounds for instances *C60V40*, *C60V50* and *C60V60*. The optimality gaps of the three instances are 21.1%, 71.7% and 26.0%, respectively, after 120 seconds (an amount of time which is sufficiently large if the operator re-optimizes, e.g., every 5 minutes). We observe significant optimality gaps and, particularly, that the primal bound improves slowly while the dual bound remains steady.



(a) *C60V40* (b) *C60V50* (c) *C60V60*

Figure 3.16: Progression of upper and lower bounds for the instances *C60V**

Figure 3.17 provides the same information for three smaller instances, namely *C25V15*, *C25V20* and *C25V25*. In this case, the instances are small enough to observe an improvement of the upper bound. Nevertheless, while a good primal solution is found rather quickly, the upper bound improves slowly. This suggests new lines of research that provide both tighter formulations and method, perhaps heuristic, to quickly find primal solutions in large scale instances.



(a) *C25V15* (b) *C25V20* (c) *C25V25*

Figure 3.17: Progression of upper and lower bounds for the instances *C25V**

3.6 Conclusion

In this paper we developed a MILP model for online order dispatching and vehicle rebalancing in a first-mile ride-sharing service.

The model was used in a rolling-horizon simulation framework based on constructed instances, to assess whether rebalancing is advantageous and whether a rebalancing strategy based on a clustering method is preferable to a random rebalancing strategy. The results show that rebalancing using a clustering-based strategy consistently outperforms strategies based on rebalancing to random locations or without rebalancing. Particularly, rebalancing strategies perform dramatically better in a context where the station is not centrally located (e.g., the business area is entirely on one side).

Our tests also show that finding high quality solutions and bounds to the problem presented becomes problematic as the size of the problem increases, suggesting future research on both efficient solution methods and tighter formulations.

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.

Appendix

Notations

Sets:

$\mathcal{N}_P := \{1, \ldots, N_P\}$ is the set of customers that have been assigned to vehicles but haven't been picked up yet in previous re-optimization phase, in which customers can be reassigned but can't be rejected in current re-optimization phase.

$\mathcal{N}_C := \{1, \ldots, N_C\}$ is the set of new customers for current re-optimization phase, in which customers can be accepted or rejected

$\mathcal{N}_U := \mathcal{N}_C \cup \mathcal{N}_P$ is the combined set of assigned and new customers.

$\mathcal{R} := \{1, \ldots, R\}$ is the set of rebalancing centers

$\mathcal{K} := \{1, \ldots, K\}$ is the set of vehicles

Parameters:

d: Denotes the destination/station node

$o(k)$: Denotes the original location of vehicle k, for $k \in \mathcal{K}$

$o(i)$: Denotes the location of request $i \in \mathcal{N}_U$

P_i: Denotes the profit of picking up customer i, for $i \in \mathcal{N}_U$

C: Denotes the unit travel time cost of the vehicle.

β: Denotes the weight parameter of rebalancing reward

V_k: Denotes the number of customers on board of vehicle k at the beginning of the operational period, for $k \in \mathcal{K}$

Q: Denotes the capacity of the vehicle

T: Denotes the start time of the operational period

T_{ij}: Denotes the travel time between locations $o(i)$ and $o(j)$, for $i \in \mathcal{K} \cup \mathcal{N}_U$, $j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}$

T_i^A: Denotes the requested arrival time of customer i, for $i \in \mathcal{N}_U$

T^L: $= max_i\{T_i^A\}$ for $i \in \mathcal{N}_U$, is the upper bound of the requested arrival time of all the customers

T_k: Denotes the requested arrival time of vehicle k, for $k \in \mathcal{K}$

T_i^P: Denotes the requested pick up time of the customer i, for $i \in \mathcal{N}_P \cup \mathcal{N}_C$

B_i: Denotes an upper bound on the number of vehicles that can move to rebalancing center $i \in \mathcal{R}$

E_i: Denotes the net profit for a vehicle to go to the rebalancing center i, for $i \in \mathcal{R}$

Δ: Denotes the maximum waiting time. Waiting time here is represented by the difference between actual pick up time and requested pick up time of the customer

$\bar{Q}$: Denotes the average number of customer on board during one trip of a vehicle

Decision variables:

x_{ij}^k: Equal to 1 if vehicle k moves directly between $o(i)$ and $o(j)$, 0 otherwise, for $i \in \{k\} \cup \mathcal{N}_U, j \in \mathcal{N}_U \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K}$.

t_k^A: Denotes the actual arrival time of vehicle k to the station, for $k \in \mathcal{K}$

t_i^P: Denotes the actual pick up time of customer i, for $i \in \mathcal{N}_U$

Chapter 4

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

This chapter contains the paper Ye et al. (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

4.1 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 (T. Eiichi, 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; Braekers 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 onboard 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 (2019a) 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., 2022b), 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 $10 \times 10\ km^2$ area.

The remainder of this paper is organized as follows. Section 4.2 describes the problem in detail and formulates it as a MILP problem. Section 4.3 presents the ALNS algorithm. Section 4.4 reports on the experiments that illustrate the performance of the algorithm. Section 4.5 describes a simulation framework which assesses the utilization of the algorithm in a realistic scenario. Finally, Section 4.6 draws final conclusions.

4.2 Problem Description and Mathematical Model

We start by formally introducing the problem in Section 4.2.1 and subsequently, in Section 4.2.2 we express it as a MILP problem extended from Ye et al. (2022b).

4.2.1 Problem Description

We consider the operator of a fleet of vehicles $\mathcal{K} := \{1, \dots, K\}$ 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 $\mathcal{N}_P$ and $\mathcal{N}_C$. The set $\mathcal{N}_P := \{1, \dots, N_P\}$ 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 $\mathcal{N}_C := \{1, \dots, N_C\}$ contains newly arrived customers whose request may or may not be accepted. For convenience we set $\mathcal{N} := \mathcal{N}_C \cup \mathcal{N}_P$.

All customers travel to a common destination d located at position $o(d)$ (e.g., a transit station) and for each customer $i \in \mathcal{N}$, the operator knows the requested arrival time T_i^A 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 $i \in \mathcal{K} \cup \mathcal{N}$, $j \in \mathcal{N} \cup \mathcal{R} \cup \{d\}$. The operator collects a revenue P_i when picking up customer i, for $i \in \mathcal{N}_C$. We assume the revenue for the customers in $\mathcal{N}_P$ has already been collected.

Finally, a set $\mathcal{R} := \{1, \dots, R\}$ 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 $i \in \mathcal{R}$ 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 β.

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 $i \in \{k\} \cup \mathcal{N}, j \in \mathcal{N} \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K}$. Furthermore, we let t_k^A denote the actual arrival time of vehicle k to the station, for $k \in \mathcal{K}$ and t_i^P denote the actual pick-up time of customer i, for $i \in \mathcal{N}$.

4.2.2 Mathematical Model

Using the notation above we formulate the FMRSP as follows:

$$\max \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}_C} \sum_{j\in\mathcal{N}\cup\{d\}} P_i x_{ij}^k - \sum_{i\in\{k\}\cup\mathcal{N}} \sum_{j\in\mathcal{N}\cup\mathcal{R}\cup\{d\}} \sum_{k\in\mathcal{K}} CT_{ij} x_{ij}^k + \beta \sum_{i\in\mathcal{R}} \sum_{k\in\mathcal{K}} x_{ki}^k E_i \tag{4.1a}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}\cup\{d\}} \sum_{k\in\mathcal{K}} x_{ij}^k \leq 1 \qquad \forall i \in \mathcal{N}_C \tag{4.1b}$$

$$\sum_{j\in\mathcal{N}\cup\{d\}} \sum_{k\in\mathcal{K}} x_{ij}^k = 1 \qquad \forall i \in \mathcal{N}_P \tag{4.1c}$$

$$\sum_{i\in\mathcal{N}\cup\mathcal{K}} x_{id}^k \leq 1 \qquad \forall k \in \mathcal{K} \tag{4.1d}$$

$$\sum_{i\in\mathcal{N}\cup\{k\}} x_{ij}^k = \sum_{i\in\mathcal{N}\cup\{d\}} x_{ji}^k \qquad \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{4.1e}$$

$$\sum_{j\in\mathcal{R}\cup\mathcal{N}\cup\{d\}} x_{kj}^k = \sum_{j\in\mathcal{N}\cup\{k\}} \sum_{i\in\mathcal{R}\cup\{d\}} x_{ji}^k \qquad \forall k \in \mathcal{K} \tag{4.1f}$$

$$\sum_{i\in\{k\}\cup\mathcal{N}} \sum_{j\in\mathcal{N}} x_{ij}^k + V_k \leq Q \qquad \forall k \in \mathcal{K} \tag{4.1g}$$

$$\sum_{k\in\mathcal{K}} x_{kj}^k \leq D_j \qquad \forall j \in \mathcal{R} \tag{4.1h}$$

$$V_k \leq Q(1 - \sum_{j\in\mathcal{R}} x_{kj}^k) \qquad \forall k \in \mathcal{K} \tag{4.1i}$$

$$V_k \leq Q \sum_{j\in\mathcal{N}\cup\{d\}} x_{kj}^k \qquad \forall k \in \mathcal{K} \tag{4.1j}$$

$$t_i^P + T_{ij} \leq t_j^P + T^L(1 - \sum_{k\in\mathcal{K}} x_{ij}^k) \qquad \forall i, j \in \mathcal{N} \tag{4.1k}$$

$$T + T_{kj} \leq t_j^P + T^L(1 - x_{kj}^k) \qquad \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{4.1l}$$

$$t_k^A \leq T_i^A + T^L(1 - \sum_{j\in\mathcal{N}\cup\{k\}} x_{ji}^k) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{4.1m}$$

$$t_k^A \leq T_k \qquad \forall k \in \mathcal{K} \tag{4.1n}$$

$$t_j^P + T_{jd} x_{jd}^k \leq t_k^A + T^L(1 - x_{jd}^k) \qquad \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{4.1o}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i \in \{k\} \cup \mathcal{N}, j \in \mathcal{N} \cup \mathcal{R} \cup \{d\}, k \in \mathcal{K} \tag{4.1p}$$

$$t_k^A \in \mathbb{R}^+ \qquad \forall k \in \mathcal{K} \tag{4.1q}$$

$$t_i^P \in \mathbb{R}^+ \qquad \forall i \in \mathcal{N} \tag{4.1r}$$

The objective function (4.1a) 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 (4.1b) and (4.1c) state that new customers may be picked up at most once and customers already accepted must be picked up exactly once, respectively. Observe, in (4.1b) and (4.1c), 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 (4.1d) ensure that vehicles travel to the station at most once. Constraints (4.1e) 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 (4.1f) 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 (4.1g) ensure that the capacity of the vehicles is not exceeded, while constraints (4.1h) 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 (4.1i) state that only empty vehicles may be dispatched to rebalancing centers. For instance, if vehicle k is dispached 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 (4.1j) 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 (4.1k) 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 $T^L := max_i\{T_i^A\}$ for $i \in \mathcal{N}$ is an upper bound on the requested arrival time. Similarly, constraints (4.1l) denote the pick-up time for the first customers in the route. Constraints (4.1k)-(4.1l) prevent subtours. Constraints (4.1m) 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^A enforcing that the actual arrival time of vehicle k is before T_i^A. Constraints (4.1n) 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 (4.1o) 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_{jd}^k takes value 1, the left-hand-side becomes $t_j^P + T_{jd}$, 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^P, the right-hand-side becomes $t_k^A + T^L$, which always holds.

Finally, constraints (4.1p)-(4.1r) define the domain of the decision variables.

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

4.3 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 2. 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 (4.1a).

The algorithm receives as input integers `maxIter` and `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 π_k and θ_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 α_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 $\alpha_1 > \alpha_2$.

If no improvement is obtained, the counter `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 α_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 π_k and frequency θ_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 2 ALNS algorithm

Input:

`maxIter`, `maxNoImprov` ▷ Maximum # iterations, total and with no improvements

$\mathcal{O}^D$, $\mathcal{O}^R$ ▷ Destroy and repair operators

r, $\alpha_1, \ldots, \alpha_4$ ▷ Constants used in the adjustment of the weights

Initialize:

$i \leftarrow 0$ ▷ Iteration counter

`NoImprov` $\leftarrow 0$ ▷ Counter of the iterations without improvement

$w_k \leftarrow 1/|\mathcal{O}^D|$ for all $k \in \mathcal{O}^D$ ▷ Initial weight of the destroy operators

$w_k \leftarrow 1/|\mathcal{O}^R|$ for all $k \in \mathcal{O}^R$ ▷ Initial weight of the repair operators

$\pi_k \leftarrow 0$ for all $k \in \mathcal{O}^D \cup \mathcal{O}^R$ ▷ Initial score of the operators

$\theta_k \leftarrow 0$ for all $k \in \mathcal{O}^D \cup \mathcal{O}^R$ ▷ Counters of the operators

$x_i \leftarrow$ `generateInitialSolution()` ▷ Initial solution

$x_c, x_b \leftarrow x_i$ ▷ Current and current best solution

while *NoImprov* $\leq$ *maxNoImprov* *and* $i \leq$ *maxIter* **do**

  $i \leftarrow i + 1$

  $x' \leftarrow x_c$

  Select a destroy operator k' randomly with probabilities $p_{k'} = \frac{w_{k'}}{\sum_{i=1}^{k} w_i}$ and apply it to x'

  $\theta_{k'} \leftarrow \theta_{k'} + 1$

  Select a repair operator k'' randomly with probabilities $p_{k''} = \frac{w_{k''}}{\sum_{i=1}^{k} w_i}$ 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**

    `NoImprov` $\leftarrow 0$

    if $f(x_c) > f(x_b)$ **then**

      $x_b \leftarrow x_c$

      $\pi_{k'} \leftarrow \pi_{k'} + \alpha_1$ ▷ k' and k'' are rewarded for a new best solution

      $\pi_{k''} \leftarrow \pi_{k''} + \alpha_1$

    else

      $x_c \leftarrow x'$

      $\pi_{k'} \leftarrow \pi_{k'} + \alpha_2$ ▷ k' and k'' are rewarded for an improving solution

      $\pi_{k''} \leftarrow \pi_{k''} + \alpha_2$

    end

  else

    `NoImprov` $\leftarrow$ `NoImprov` $+ 1$

    if *x' is accepted by simulated annealing criteria* **then**

      $x_c \leftarrow x'$

      $\pi_{k'} \leftarrow \pi_{k'} + \alpha_3$ ▷ k' and k'' are rewarded for an accepted non-improving solution

      $\pi_{k''} \leftarrow \pi_{k''} + \alpha_3$

    else

      $\pi_{k'} \leftarrow \pi_{k'} + \alpha_4$ ▷ k' and k'' are rewarded for a non-improving solution

      $\pi_{k''} \leftarrow \pi_{k''} + \alpha_4$

    end

  end

  if i *modulo* $100 = 0$ **then**

    for $k \in \mathcal{O}^D \cup \mathcal{O}^R$ **do**

      if $\theta_k > 0$ **then**

        $w_k \leftarrow w_k(1 - r) + r\frac{\pi_k}{\theta_k^l}$ ▷ Updates the weight

        $\pi_k, \theta_k \leftarrow 0$ ▷ Resets scores and counters

      end

    end

  end

end

4.3.1 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 3. 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 4) 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 3 Allocation of previous customer

Input:
$\mathcal{N}_P, \mathcal{K}$
for $c \in \mathcal{N}_P$ **do**
  Let $\mathcal{K}^O$ be a randomly ordered list of the vehicles with residual capacity
  for $v \in \mathcal{K}^O$ **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_i^v be the cost of T
      else
        $C_i^v = \infty$
      end
    end
    if $min_{i=1,\ldots,Q_v}\{C_i^v\} < \infty$ **then**
      Assign customer c to position $i = \arg\min_{i=1,\ldots,Q_v}\{C_i^v\}$ of the tour of vehicle v
      Move on to next customer
    end
  end
end

Algorithm 4 Allocation of new customers

Input:
$\mathcal{N}_C$, $\mathcal{K}$
for $c \in \mathcal{N}_C$ **do**
  Let $\mathcal{K}^O$ be a randomly ordered list of the vehicles with residual capacity
  for $v \in \mathcal{K}^O$ **do**
    for $i = 1, .., Q_v$ **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 $i = \arg\min_{i=1,...,Q_v}\{C_i^v\}$ of the tour of vehicle v
        Move on to next customer
      end
    end
  end
end

Figure 4.1 illustrates the construction heuristic in a small fictitious example. We have $\mathcal{N}_P := \{P1\}$, $\mathcal{N}_C := \{C1, C2, C3\}$, $\mathcal{R} := \{R1\}$, $\mathcal{K} : \{D1, D2, D3\}$. 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 $R1$.

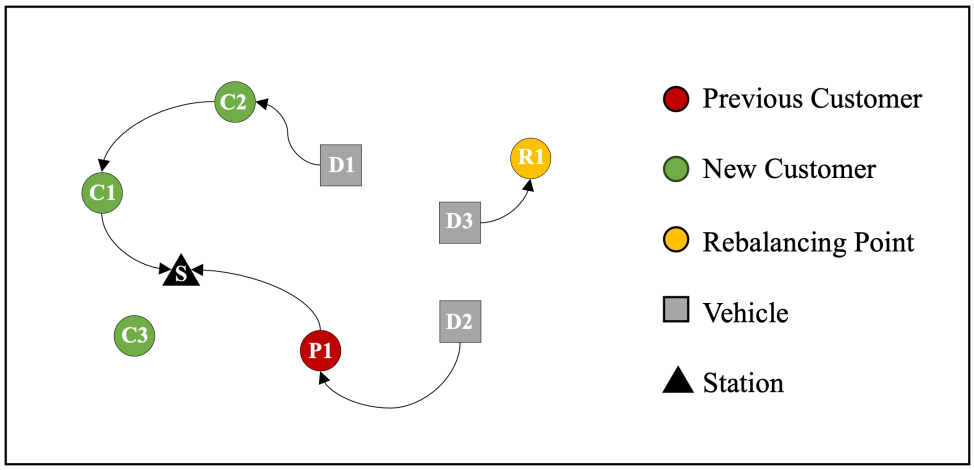


Figure 4.1: Example construction of a feasible solution.

4.3.2 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 $T := T(1 - i/\texttt{maxIter})$. During parameter configuration, we conduct numerical experiments to identify the optimal value of M.

4.3.3 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.

4.3.4 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.

4.4 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 10×10 km^2 for a 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.

4.4.1 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 $V|\mathcal{K}|-C|\mathcal{N}_C|-P|\mathcal{N}_P|-R|\mathcal{R}|$, 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 10×10 Km^2 area, see Figure 4.2. 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 4.2. 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 $k \in \mathcal{K}$. 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 $\beta = 0.1$.

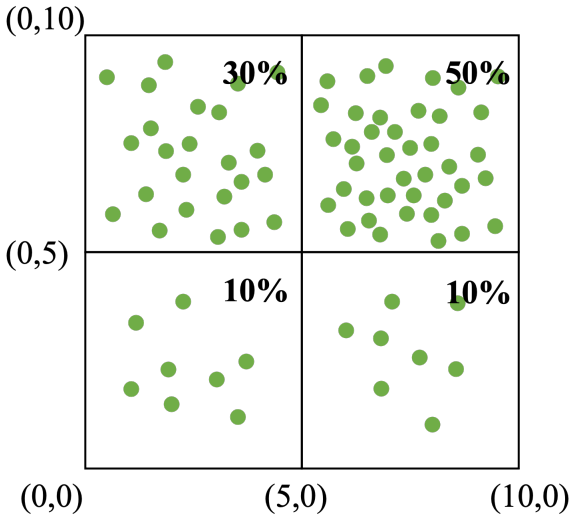


Figure 4.2: Distribution of New Customers

4.4.2 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 4.1. 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.

4.4.3 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 4.3 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.

Table 4.1: Parameter Tuning

Instance	Temperature Percentage		
	0.2	**0.5**	**0.8**
V50-C150-P45-R3-1	3611.78	3607.15	3604.88
V50-C150-P45-R3-2	3633.15	3634.14	3631.71
V50-C150-P45-R3-3	3670.70	3683.53	3671.77
Average	3638.54	**3641.61**	3636.12
V100-C300-P50-R3-1	7588.80	7623.01	7623.96
V100-C300-P50-R3-2	7487.07	7474.69	7470.90
V100-C300-P50-R3-3	7428.76	7411.41	7410.98
Average	7501.54	**7503.04**	7501.95
V150-C450-P75-R3-1	11282.46	11282.46	11282.46
V150-C450-P75-R3-2	11255.32	11275.27	11257.27
V150-C450-P75-R3-3	11132.66	11140.82	11127.39
Average	11223.48	**11232.85**	11222.37
V200-C600-P100-R3-1	15195.61	15200.96	15191.71
V200-C600-P100-R3-2	14987.19	14992.08	14983.87
V200-C600-P100-R3-3	14805.90	14828.38	14815.52
Average	14996.24	**15007.14**	14997.03
Total Average	9339.95	**9346.16**	9339.37

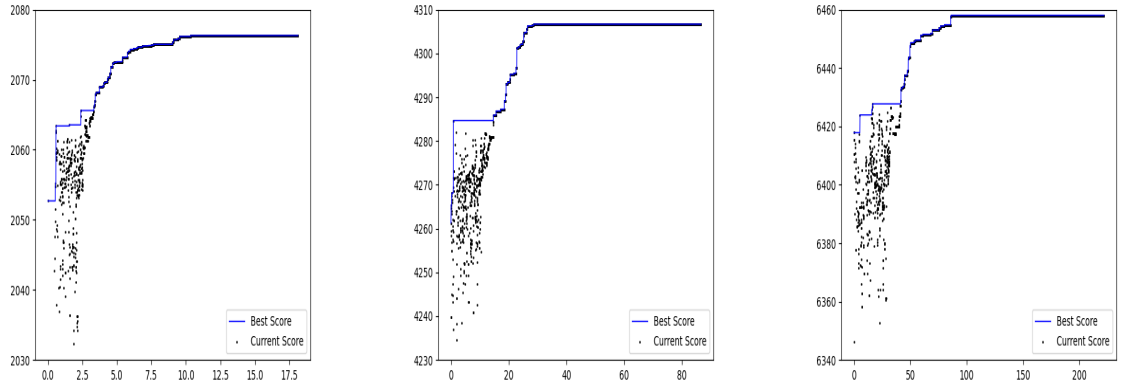
This often happens in significant jumps after the initial phases where many worse solutions were evaluated. It appears that the initial exploration phase is followed by an exploitation phase in more promising regions of the solution space.

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 4.2 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 able to deliver feasible solutions. In contrast, the ALNS algorithm delivers feasible solutions with an average optimality gap of 10% within a 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.



(a) V50-C100-P45-R3-3 (b) V100-C200-P50-R3-3 (c) 150-C300-P75-R3-3

Figure 4.3: ALNS Process

4.5 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 4.4.1. 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 customers 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 $V40 - C|\mathcal{N}|$, where $|\mathcal{N}|$ 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 4.3 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.

4.6 Conclusion

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.

Table 4.2: Comparison of ALNS and Gurobi. Optimality gaps are computed as (`bestUpperBound-objective`)/(`bestUpperBound`+ϵ), where `bestUpperBound` is the best upper bound delivered by Gurobi (when available) and ϵ is a small constant. A dash − indicates that no feasible solution was found. *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.

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

Table 4.3: 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.

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

Bibliography

High fuel prices impoverish new york city taxi drivers. 2008. doi: https://www.wsws.org/en/articles/2008/09/taxi-s04.html.

Taxi driver salary in new york. 2020a. doi: https://www.indeed.com/career/taxi-driver/salaries/New-York--NY.

Taxi rate new york city. 2020b. doi: https://www.taxi-calculator.com/taxi-rate-new-york-city/259.

New york city taxi and limousine commission. 2021. URL https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

A. A. Abubakr O, G. Arnob. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2019. doi: 10.1109/tits.2019.2931830.

D. R. A. Wallar, J. Alonso-Mora. Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand. *International Conference on Robotics and Automation, ICRA, IEEE*, pages 3853–3859, 2019. doi: 10.1109/ICRA.2019.8793685.

J. Alonso-mora, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences of the United States of America*, 115(3):E555, 2018. ISSN 10916490. doi: 10.1073/pnas.1721622115.

J. A.-M. B. A. Beirigo, R. R. Negenborn and F. Schulte. A business class for autonomous mobility-on-demand: Modeling service quality contracts in dynamic ridesharing systems. *Transportation Research Part C: Emerging Technologies*, 136, 2022. doi: https://doi.org/10.1016/j.trc.2021.103520.

E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

G. Berbeglia, J. F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.04.024.

D. Bertsimas, P. Jaillet, and S. Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 67(1):143–162, 2019. doi: 10.1287/opre.2018.1763. URL https://doi.org/10.1287/opre.2018.1763.

Z. Bian and X. Liu. Mechanism design for first-mile ridesharing based on personalized requirements part II: Solution algorithm for large-scale problems. *Transportation Research Part B: Methodological*, 120:172–192, 2019a. ISSN 01912615. doi: 10.1016/j.trb.2018.12.014. URL https://doi.org/10.1016/j.trb.2018.12.014.

Z. Bian and X. Liu. Mechanism design for first-mile ridesharing based on personalized requirements part I: Theoretical analysis in generalized scenarios. *Transportation Research Part B: Methodological*, 120:147–171, 2019b. ISSN 01912615. doi: 10.1016/j.trb.2018.12.009. URL https://doi.org/10.1016/j.trb.2018.12.009.

Z. Bian, X. Liu, and Y. Bai. Mechanism design for on-demand first-mile ridesharing. *Transportation Research Part B: Methodological*, 138:77–117, 2020. ISSN 01912615. doi: 10.1016/j.trb.2020.03.011. URL https://doi.org/10.1016/j.trb.2020.03.011.

K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers and Industrial Engineering*, 99:300–313, 2016. ISSN 03608352. doi: 10.1016/j.cie.2015.12.007. URL http://dx.doi.org/10.1016/j.cie.2015.12.007.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005. doi: 10.1287/trsc.1030.0056. URL https://doi.org/10.1287/trsc.1030.0056.

S. Chen, H. Wang, and Q. Meng. Solving the first-mile ridesharing problem using autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, 35(1):45–60, 2020. ISSN 14678667. doi: 10.1111/mice.12461.

Y. Chen and H. Wang. Pricing for a Last-Mile Transportation System. *Transportation Research Part B: Methodological*, 107:57–69, 2018. ISSN 01912615. doi: 10.1016/j.trb.2017.11.008.

N. Y. C. T. Commission and Limousine. New York City Taxi and Limousine Commission. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2020. [Online; accessed 10-Oct-2022].

J. F. Cordeau and G. Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4OR*, 1(2):89–101, 2003. ISSN 16142411. doi: 10.1007/s10288-002-0009-8.

G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1): 80–91, 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.80.

S. English. High fuel prices impoverish New York City taxi drivers. https://www.wsws.org/en/articles/2008/09/taxi-s04.html, 2008. [Online; accessed 10-Oct-2022].

C. A. Floudas. Nonlinear and mixed-integer optimization: Fundamentals and applications. 1995. URL https://api.semanticscholar.org/CorpusID:118032488.

H. S. M. H. K. R. F. Pinto, M. F. Hyland and I. O. Verbas. Joint design of multimodal transit networks and shared autonomous mobility fleets. *Transportation Research Part C: Emerging Technologies*, pages 2–20, 2019.

J. Han. Looking through the taxi meter - analysis of the nyc green taxi data. 2019.

S. C. Ho, W. Y. Szeto, Y.-h. H. Kuo, J. M. Y. Leung, M. Petering, and T. W. H. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, 2018. ISSN 01912615. doi: 10.1016/j.trb.2018.02.001. URL https://doi.org/10.1016/j.trb.2018.02.001.

Y. Huang, F. Bastani, R. Jin, and X. S. Wang. Large scale realtime ridesharing with service guarantee on road networks. In *Proceedings of the VLDB Endowment*, 2014. doi: 10.14778/2733085.2733106.

S. Illgen and M. Höck. Literature review of the vehicle relocation problem in one-way car sharing networks. *Transportation Research Part B: Methodological*, 120:193–204, 2019. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2018.12.006. URL https://www.sciencedirect.com/science/article/pii/S0191261518300547.

INSHUR. How Much Does an Uber Driver Make in New York City? https://inshur.com/blog/how-much-does-an-uber-driver-make-in-new-york-city/. [Online; accessed 10-Oct-2022].

R. Z. E. F. D. M. M. P. K. Spieser, K. Treleaven. Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in singapore. *Road vehicle automation*, page 229– 245, 2014.

M. P. K. Treleaven and E. Frazzoli. Asymptotically optimal algorithms for one-to-one pickup and delivery problems with applications to transportation systems. *IEEE Transactions on Automatic Control*, 58:2261–2276, 2013.

G. H. d. A. C. K. Winter, O. Cats and B. V. Arem. Designing an automated demand-responsive transport system: Fleet size and performance analysis for a campus—train station service. *Transportation Research Record: Journal of the Transportation Research Board*, pages 75–83, 2016.

S. N. Kumar and R. Panneerselvam. A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, 04(03):66–74, 2012. ISSN 2160-5912. doi: 10.4236/iim.2012.43010.

X. Li, J. Zhang, J. Bian, Y. Tong, and T.-Y. Liu. A Cooperative Multi-Agent Reinforcement Learning Framework for Resource Balancing in Complex Logistics Network. *International Foundation for Autonomous Agents and Multiagent Systems(AAMAS '19)*, page 980–988, 2019.

C. Lin, K. L. Choy, G. T. Ho, S. H. Chung, and H. Y. Lam. Survey of Green Vehicle Routing Problem: Past and future trends. *Expert Systems with Applications*, 41(4 PART 1):1118–1138, 2014. ISSN 09574174. doi: 10.1016/j.eswa.2013.07.107.

K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. *Proceedings of the ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, pages 1774–1783, 2018. doi: 10.1145/3219819.3219993.

G. L. M. Desrochers. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

G. R. S. H. S. M. M. Vazifeh, P. Santi and C. Ratti. Addressing the minimum fleet problem in on-demand urban mobility. *Nature*, 557:534, 2018.

T.-y. Y. Ma, S. Rasulkhani, J. Y. J. Chow, and S. Klein. A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. *Transportation Research Part E: Logistics and Transportation Review*, 128(December 2018):417–442, 2019. ISSN 13665545. doi: 10.1016/j.tre.2019.07.002. URL https://doi.org/10.1016/j.tre.2019.07.002.

K. Mahendru. How to Determine the Optimal K for K-Means? https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb, 2019. [Online; accessed 10-Oct-2022].

N. Masoud and R. Jayakrishnan. A decomposition algorithm to solve the multi-hop Peer-to-Peer ride-matching problem. *Transportation Research Part B: Methodological*, 99(May):1–29, 2017. ISSN 01912615. doi: 10.1016/j.trb.2017.01.004.

N. Masoud, R. Lloret-Batlle, and R. Jayakrishnan. Using bilateral trading to increase ridership and user permanence in ridesharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 2017. ISSN 13665545. doi: 10.1016/j.tre.2017.04.007.

A. Mourad, J. Puchinger, and C. Chu. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, 123:323–346, 2019. ISSN 01912615. doi: 10.1016/j.trb.2019.02.003. URL https://doi.org/10.1016/j.trb.2019.02.003.

F. C. P. M. Boesch and K. W. Axhausen. Autonomous vehicle fleet sizes required to serve different levels of demand. *Transportation Research Record: Journal of the Transportation Research Board*, 2542:111–119, 2016.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal fur Betriebswirtschaft*, 58 (2):81–117, 2008. ISSN 03449327. doi: 10.1007/s11301-008-0036-4.

C. Pfeiffer and A. Schulz. An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. *OR Spectrum*, 44(1):87–119, 2022. ISSN 14366304. doi: 10.1007/s00291-021-00656-7. URL https://doi.org/10.1007/s00291-021-00656-7.

V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013. ISSN 03772217. doi: 10.1016/j.ejor.2012.08.015. URL http://dx.doi.org/10.1016/j.ejor.2012.08.015.

D. Pisinger and S. Røpke. Large Neighborhood Search. *Handbook of Metaheuristics(2 ed.)*, pages 339–420, 2010.

Z. Qin, X. Tang, Y. Jiao, F. Zhang, C. Wang, and Q. Li. Deep reinforcement learning for ride-sharing dispatching and repositioning. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus:6566–6568, 2019. ISSN 10450823. doi: 10.24963/ijcai.2019/958.

U. Ritzinger, J. Puchinger, and R. F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016. ISSN 1366588X. doi: 10.1080/00207543.2015.1043403.

S. Ropke and J.-F. F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009. ISSN 15265447. doi: 10.1287/trsc.1090.0272.

S. Ropke, D. Pisinger, S. Ropke, and D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. (April 2019), 2006. doi: 10.1287/trsc.1050.0135.

M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological*, 82:36–53, 2015. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2015.07.025. URL https://www.sciencedirect.com/science/article/pii/S0191261515002088.

M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar. Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers and Operations Research*, 90:12–21, 2018. ISSN 03050548. doi: 10.1016/j.cor.2017.08.016. URL http://dx.doi.org/10.1016/j.cor.2017.08.016.

Y. T. Eiichi, T. Russell G. Recent trends and innovations in modelling city logistics. 125:4–14, 2014. doi: 10.1016/j.sbspro.2014.01.1451.

X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, J. Ye, A. I. Labs, D. Chuxing, X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1780–1790, 2019. doi: 10.1145/3292500.3330724.

P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898718515. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.

Uber. Uber charges partners 25% fee on all fares. 2020. doi: https://www.uber.com/gh/en/drive/basics/tracking-your-earnings/.

A. Wallar, M. Van Der Zee, J. Alonso-Mora, and D. Rus. Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing. *IEEE International Conference on Intelligent Robots and Systems*, pages 4539–4546, 2018. ISSN 21530866. doi: 10.1109/IROS.2018.8593743.

H. Wang. Routing and scheduling for a last-mile transportation system. *Transportation Science*, 53(1):131–147, 2019. ISSN 15265447. doi: 10.1287/trsc.2017.0753.

X. Wang, N. Agatz, and A. Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 52(4):850–867, 2018. ISSN 15265447. doi: 10.1287/trsc.2017.0768.

J. Wen, J. Zhao, and P. Jaillet. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-March(October):220–225, 2018. doi: 10.1109/ITSC.2017.8317908.

Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, page 905–913, 2018. doi: 10.1145/3219819.3219824. URL https://doi.org/10.1145/3219819.3219824.

M. O. Y. Yuan, D. Cattaruzza and F. Semet. A note on the lifted miller–tucker–zemlin subtour elimination constraints for routing problems with time windows. *Operations Research Letters*, 48:167–169, 2020.

J. Ye, G. Pantuso, and D. Pisinger. Fleet size control in first-mile ride-sharing problems. *Computational Logistics. ICCL 2022. Lecture Notes in Computer Science*, 13557, 2022a. doi: https://doi.org/10.1007/978-3-031-16579-5_7.

J. Ye, G. Pantuso, and D. Pisinger. Online order dispatching and vacant vehicles rebalancing for the first-mile ride-sharing problem. 2022b. doi: http://dx.doi.org/10.2139/ssrn.4265368.

J. Ye, G. Pantuso, and D. Pisinger. Adaptive large neighborhood search for order dispatching and vacant vehicle rebalancing in first-mile ride-sharing services. 2023. doi: http://dx.doi.org/10.2139/ssrn.4517039.

J. Z. Yu Shen, Hongmou Zhang. Integrating Shared Autonomous Vehicle in Public Transportation System: A Supply-Side Simulation of the First-Mile Service in Singapore. *Transportation Research Part A*, 113:125–136, 2018. ISSN 0965-8564. doi: 10.1016/j.tra.2018.04.004. URL https://doi.org/10.1016/j.tra.2018.04.004.