OPERATING ON-DEMAND RIDE-SHARING SERVICES

A Dissertation
Presented to
The Academic Faculty

By

Connor Riley

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
Department of Industrial and Systems Engineering

Georgia Institute of Technology
August 2022

© Connor Riley 2022
OPERATING ON-DEMAND RIDE-SHARING SERVICES

Thesis committee:

Dr. Pascal Van Hentenryck  
Industrial and Systems Engineering  
*Georgia Institute of Technology*

Dr. Mathieu Dahn  
Industrial and Systems Engineering  
*Georgia Institute of Technology*

Dr. Pinar Keskinocak  
Industrial and Systems Engineering  
*Georgia Institute of Technology*

Dr. Antoine Legrain  
Mathematical and Industrial Engineering  
*Polytechnique Montréal*

Dr. Alan Erera  
Industrial and Systems Engineering  
*Georgia Institute of Technology*

Date approved: July 27, 2022
For my Father
ACKNOWLEDGMENTS

First and foremost, I have to thank my advisor, Pascal Van Hentenryck, without whom none of this would be possible. Pascal’s ideas, advice, and support throughout this process were invaluable. I would also like to express my gratitude to the members of my thesis committee for their help in preparation of this work, especially Antoine Legrain, who was instrumental in developing my understanding of Operations Research and vehicle routing and whose ideas helped shape chapter 2. I am deeply grateful to Enpeng Yuan for his insight and especially for his contribution of the prediction algorithm used in chapter 3. I would like to offer a special thanks to Kevin Dalmeijer for his friendship and advice throughout the later years of my PhD candidacy and for helping me fine-tune my thesis defense presentation and to Anthony Trasatti without whom MARTA Reach would not exist. I would also like extend my sincere thanks to Laurent Michel, whose belief in me provided the catalyst for this journey.

Writing a dissertation requires more than academics, and I would like to express my gratitude and appreciation for all my family and friends whose support made this work possible. Special thanks to Tom Logan and Thomas Chen, whose friendship made Ann Arbor feel like home and to Julie Lee, for going beyond the expectations of a brief friendship and supporting me through the most trying time in my life. I am deeply indebted to Ciara Beaulieu, whose empathy knows no bounds, and to Colin Gerrity, whose love and support I depend on every day. You are both family, and I could not have done this without you. I cannot begin to express my thanks to my Mother and sister for their unwavering belief in me. And lastly, to my Father, for everything. This dissertation stands as a testament to your love and encouragement.
# TABLE OF CONTENTS

Acknowledgments ................................................................. iv  
List of Tables ................................................................. ix  
List of Figures ................................................................. x  
List of Acronyms ............................................................... xiii  
Summary .............................................................................. xiv  
Chapter 1: Introduction and Background ................................. 1  
Chapter 2: Column Generation for Real-Time Ride-Sharing Operations ........ 4  
  2.1 Introduction ................................................................. 4  
  2.2 Related Work ............................................................... 6  
  2.3 Overview of the Approach ................................................. 7  
  2.4 The Static Problem ......................................................... 7  
  2.5 The Column-Generation Algorithm ................................. 11  
      2.5.1 The Master Problem ................................................. 11  
      2.5.2 The Pricing Problem ............................................... 12  
      2.5.3 The Column Generation ........................................... 12  
  2.6 The Real-Time Problem ................................................ 15
2.7 Experimental Results

2.7.1 Instance Description

2.7.2 Virtual Stops

2.7.3 Algorithmic Setting

2.7.4 Wait Times

2.7.5 Trip Deviation

2.7.6 The Impact of the Fleet Size

2.7.7 The Impact of Vehicle Capacity

2.7.8 The Impact of the Penalty

2.7.9 Final Vehicle Assignments

2.7.10 The Impact of Column Generation

2.7.11 The Impact of Pruning

2.7.12 The Impact of Ride Sharing

2.7.13 Comparison with Prior Work

2.8 Conclusion

Chapter 3: Real-Time Dispatching of Large-Scale Ride-Sharing Systems: Integrating Optimization, Machine Learning, and Model Predictive Control

3.1 Introduction

3.2 Problem Statement

3.3 Related Work

3.4 Overall Organization

3.4.1 The Optimization Problem
4.3 Idle Vehicles and Relocation ........................................... 48
  4.3.1 Determining the Desired Number of Vehicles per Stop ......... 48
  4.3.2 Vehicle Relocation .................................................. 49
  4.3.3 Tie-breaking ......................................................... 50
4.4 Experimental Results .................................................... 52
  4.4.1 Reduction in Waiting Times ....................................... 53
  4.4.2 Vehicle Information ................................................ 56
  4.4.3 Relocation ........................................................... 56
4.5 Conclusion ............................................................... 58

Appendices ................................................................. 59

References ................................................................. 60

Vita .......................................................................... 63
LIST OF TABLES

3.1 Average Waiting Times by Instance Sizes. . . . . . . . . . . . . . . . . . . 38
LIST OF FIGURES

2.1 The Static Formulation of the Dial-A-Ride Problem. . . . . . . . . . . . . . 9
2.2 The Master Problem Formulation. . . . . . . . . . . . . . . . . . . . . . . 12
2.3 The Pricing Problem Formulation for Vehicle $v$. . . . . . . . . . . . . . 13
2.4 The Penalty Function for Unserved Customers. . . . . . . . . . . . . . . . 16
2.5 The Histogram of Wait Times (Log Scale). . . . . . . . . . . . . . . . . . 18
2.6 The Histogram of Trip Deviations (Log Scale). . . . . . . . . . . . . . . . 18
2.7 The Impact of the Fleet Size on the Average Wait Times and Average De-
viations on All Instances. . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
2.8 The Impact of the Vehicle Capacity on the Average Wait Times and the
Average Trip Deviations on All Instances. . . . . . . . . . . . . . . . . . . 20
2.9 The Impact of the Penalty on Average Wait Times. . . . . . . . . . . . . . 20
2.10 Times Until Final Vehicle Assignments. . . . . . . . . . . . . . . . . . . 20
2.11 The Number of Generated Columns as a Percentage of Possible Combin-
ations of Requests/Vehicles. The x-Axis value are scaled by $10^{-3}$. . . . . . 21
2.12 Optimization Times With and Without Pruning. . . . . . . . . . . . . . . 21
2.13 The Impact of the Fleet Size on the Average Vehicle Utilization and Idle
Time on All Instances. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
3.1 The Anticipatory Real-time Ride-sharing (A-RTRS) Architecture for Real-
Time Dispatching. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
3.2 The Restricted Master Problem Formulation. . . . . . . . . . . . . . . . . 32
3.3 The MPC-MIP Model for Zone Balancing. ........................................... 36
3.4 The Vehicle Relocation Optimization (VR-MIP). ............................. 36
3.5 Histograms of Waiting Times for A-RTRS and Myopic Real-time Ride-sharing (M-RTRS) (Logarithmic Y-Scale). ................................. 39
3.6 Histograms of Waiting Times for A-RTRS and Oracle Anticipatory Real-time Ride-Sharing (OA-RTRS) (Logarithmic Y-Scale). ...................... 39
3.7 Waiting Times for A-RTRS and M-RTRS. ......................................... 40
3.8 Waiting Times for A-RTRS and OA-RTRS. ...................................... 40
3.9 Percentage Improvement in Waiting Times by Origin. ....................... 41
3.10 Total Number of Requests By Zone (over all instances). .................... 41
3.11 Average Waiting Times by Zone. .................................................. 41
3.12 Average Number of Passengers per Vehicle. .................................. 42
3.13 Average Number of Passengers per Vehicle. .................................. 42
3.14 Idling Times per Vehicle. ............................................................ 43
3.15 Relocation Time per Vehicle. ....................................................... 44
3.16 Average Times Vehicles Spend Relocating. ................................... 44
4.1 The Ratio Rebalancing Optimization (VR-Ratio). ........................... 49
4.2 The Vehicle Relocation Optimization (RVR-MIP). ............................. 50
4.3 The Vehicle Relocation Optimization with tie-breaking (TBVR-MIP) .... 52
4.4 Histograms of Waiting Times for M-RTRS and Ratio Rebalancing Real-time Ride-sharing (RR-RTRS) (Logarithmic Y-Scale). ...................... 53
4.5 Histograms of Waiting Times for A-RTRS and RR-RTRS (Logarithmic Y-Scale). ......................................................... 54
4.6 Histograms of Waiting Times for RR-RTRS (min) and RR-RTRS (median) (Logarithmic Y-Scale). ......................................................... 54
4.7 Histograms of Waiting Times for RR-RTRS (min) and RR-RTRS (max) (Logarithmic Y-Scale). ................................. 55

4.8 Waiting Times for M-RTRS, A-RTRS, and RR-RTRS by instance size. . . . 55

4.9 Waiting Times for RR-RTRS (min), RR-RTRS (median), and RR-RTRS (max) by instance size. ................................. 56

4.10 Idling Times per Vehicle. ........................................... 57

4.11 Relocation Time per Vehicle. ....................................... 57

4.12 Average Times Vehicles Spend Relocating. ...................... 58
LIST OF ACRONYMS

**A-RTRS** Anticipatory Real-time Ride-sharing

**AR** Autoregression

**BRT** bus rapid transit

**CPAIOR** Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research

**DRT** demand-responsive transport

**IJCAI** International Joint Conferences on Artificial Intelligence

**M-RTRS** Myopic Real-time Ride-sharing

**MARTA** Metropolitan Atlanta Rapid Transit Authority

**MIP** Mixed Integer Program

**MPC** Model Predictive Control

**NYCTLC** New York City Taxi and Limousine Commission

**OA-RTRS** Oracle Anticipatory Real-time Ride-Sharing

**RMP** restricted master problem

**RR-RTRS** Ratio Rebalancing Real-time Ride-sharing

**TNCs** Transportation Network Companies

**VAR** Vector Autoregression
SUMMARY

Public transit agencies are increasingly exploring mobility options to supplement their traditional rail, bus, and streetcar offerings [1, 2]. One such option is demand-responsive transport (DRT), “any non-fixed route system of transporting individuals that requires advanced scheduling by the customer” [3]. DRT presents challenging design and operations problems, including fleet sizing, network design, and dispatching. In this thesis, we present optimization techniques to address operational challenges in demand responsive transit.

In Chapter 2, we review the real-time dial-a-ride problem, a vehicle routing problem with pickups and deliveries, deviation, and capacity constraints, and present a dispatching algorithm, M-RTRS, which provides service guarantees, serving all customers with a small number of vehicles while minimizing wait times. In a computational study, we show that this algorithm scales to over 30,000 requests per hour, providing an effective way to support large-scale ride-sharing services in dense cities.

In Chapter 3, we introduce an approach for vehicle dispatching, A-RTRS, that tightly integrates a state-of-the-art dispatching algorithm, a machine-learning model to predict zone-to-zone demand over time, and a model predictive control optimization to relocate idle vehicles. This is shown to decrease the average wait time of passengers in a computational study.

In Chapter 4, we present a relocation algorithm designed to address two challenges faced when deploying a real-world real-time dial-a-ride service. The first, a lack of historic data, because in a real-world deployment, initial adoption may be slow, and thus accumulating the amount of data needed for the machine learning approach to demand prediction presented in Chapter 3 may be impractical. The second, that vehicles may be restricted in the locations that they may idle, which must be considered when relocating them. In a computational study, we show this approach yields similar average wait time decreases to A-RTRS.
CHAPTER 1
INTRODUCTION AND BACKGROUND

In the past decade mobility options in cities around the world have dramatically increased. Transportation Network Companies (TNCs) like Uber and Lyft have fundamentally transformed mobility, providing on-demand door-to-door transportation through mobile applications. E-scooters and E-bike companies like Lime and Bird have increased micro-mobility options, while Zipcar and Getaround have pioneered car-sharing. In public transit, agencies are exploring mobility options such as bus rapid transit (BRT) and DRT to supplement their traditional rail, bus, and streetcar offerings. All of these options present potential society benefits by reducing the reliance on personal vehicles and thus decreasing congestion, greenhouse emissions, and trip prices. However, many of these options introduce new challenges to be considered when designing infrastructure, and when planning and operating these services.

This thesis considers problems encountered when operating on-demand ride-sharing services, like DRT. One such problem is the dial-a-ride problem, which consists of defining the routes for a set of vehicles while ensuring that a given set of transportation requests are served. Variants of this problem include imposing constraints on the amount of time passengers can wait to be picked up and limiting the amount of time passengers can spend in transit relative to the time it would take to complete a direct trip from the passengers origin to their destination. Dial-a-ride problems can also be solved in real-time, meaning vehicle routes must be changed dynamically as requests come into the system. Another problem is the real-time vehicle rebalancing problem, which takes the set of vehicles that are not serving customers and attempts to relocate them to areas of higher demand.

Variants of both the real-time dial-a-ride problem, and the vehicle rebalancing problem are encountered in ride-hailing, and ride-sharing domains, such as Uber (Pool) and Lyft,
and also in a public setting when operating paratransit and DRT service. However, the
algorithms used by both public and commercial ride-sharing services rarely use state-of-the-art techniques, which decreases system efficiency and reduces potential positive impact.
In this thesis, we develop new algorithms to solve both the real-time dial-a-ride problem and the real-time vehicle rebalancing problem. These algorithms are specifically designed for settings where driver behavior is not a factor, including public DRT and paratransit, and a potential future commercial ride-sharing service using autonomous vehicles.

The rest of this thesis is structured as follows. Chapter 2 details the online dial-a-ride problem, and presents a novel approach for solving such problems, M-RTRS. This was previously published as “Column Generation for Real-Time Ride-Sharing Operations” in the 2019 proceedings of the Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR) [4]. It is joint work with Antoine Legrain and Pascal Van Hentenryck and, with the exception of formatting changes to conform to the style requirements of a Georgia Institute of Technology PhD thesis, appears here unaltered. Chapter 3 introduces an end-to-end framework for dispatching and relocating idle vehicles in an online dial-a-ride system, A-RTRS. This was previously published as “Real-Time Dispatching of Large-Scale Ride-Sharing Systems: Integrating Optimization, Machine Learning, and Model Predictive Control” in the 2020 proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI) [5]. It is joint work with Enpeng Yuan and Pascal Van Hentenryck and, with the exception of formatting changes to conform to the style requirements of a Georgia Institute of Technology PhD thesis, appears here unaltered. Chapter 4 explores a separate idle vehicle relocation method, designed to address the specific requirements of Metropolitan Atlanta Rapid Transit Authority (MARTA) Reach [1]. The first, a lack of historic data, because in a real-world deployment, initial adoption may be slow, and thus accumulating the amount of data needed for the machine learning approach to demand prediction presented in Chapter 3 may be impractical. The second, that vehicles may be restricted in the locations that they may idle, which must be
considered when relocating them.
CHAPTER 2
COLUMN GENERATION FOR REAL-TIME RIDE-SHARING OPERATIONS

2.1 Introduction

In the past decade, commercial ride-hailing services such as Didi, Uber, and Lyft have decreased reliance on personal vehicles and provided new mobility options for various population segments. More recently, ride-sharing has been introduced as an option for customers using these services. Ride-sharing has the potential for significant positive impact since it can reduce the number of cars on the roads and thus congestion, decrease greenhouse emissions, and make mobility accessible to new population segments by decreasing trip prices. However, the algorithms used by commercial ride-sharing services rarely use state-of-the-art techniques, which reduces the potential positive impact. Recent research by Alonso-Mora et al. [6] has shown the benefits of more sophisticated algorithms. Their algorithm uses shareability graphs and cliques to generate all possible routes and a Mixed Integer Program (MIP) model to select the routes. They impose significant constraints on waiting times (e.g., 420 seconds), which reduces the potential riders to consider for each route at the cost of rejecting customers.

This paper considers large-scale ride-sharing services where customers are always guaranteed a ride, in contrast to prior work. The Myopic Real-Time Dial-A-Ride System (M-RTRS) divides the days into short time periods called epochs, batches requests in a given epoch, and then schedules customers to minimize average waiting times. M-RTRS makes a number of modeling and solving contributions. At the modeling level, M-RTRS has the following innovations:

1. M-RTRS follows a Lagrangian approach, relaxing the constraint that all customers must be served in the static optimization problem of each epoch. Instead, M-RTRS asso-
associates a penalty with each rider, representing the cost of not serving the customer.

2. To balance the minimization of average waiting times and ensure that the waiting time
   of every customer is reasonable, M-RTRS increases the penalty of an unserved customer in
   the next epoch, making it increasingly harder not to serve the waiting rider.

3. M-RTRS exploits a key property of the resulting formulation to reduce the search space
   explored for each epoch.

4. To favor ride-sharing, M-RTRS uses the concept of virtual stops used in the RITMO
   project [7] and being adopted by ride-sourcing services.

M-RTRS solves the static optimization problem for each epoch with a column-generation
algorithm based on the three-index MIP formulation [8]. The main innovation here is the
pricing problem which is organized as a series of waves, first considering all the insertions
of a single customer, before incrementally adding more customers.

M-RTRS was evaluated on historic taxi trips from the New York City Taxi and Limousine
Commission [9], which contains large-scale instances with more than 30,000 requests
an hour. The results show that M-RTRS can provide service guarantees while improving
the state-of-the-art results. For instance, for a fleet of 2,000 vehicles of capacity 4, M-RTRS
obtains an average wait of 2.2 minutes and an average deviation from the shortest path of
0.62 minutes. The results also show that large-occupancy vehicles (e.g., 8-passenger vehi-
cles) provide additional benefits in terms of waiting times with negligible increases in in-
vehicle time. M-RTRS is also shown to generate a small fraction of the potential columns,
explaining its efficiency. The Lagrangian modeling also helps in reducing computation
times significantly.

The rest of this paper is organized as follows. Section 2.2 presents the related work
in more detail. Section 2.3 describes the real-time setting. Section 2.4 specifies the static
problem and gives the MIP formulation. Section 2.5 describes the column generation.
Section 2.6 specifies the real-time operations. Section 2.7 presents the experimental results and section 2.8 concludes the paper.

2.2 Related Work

Dial-a-ride problems have been a popular topic in operations research for a long time. Cordeau and Laporte [8] provided a comprehensive review of many of the popular formulations and the starting point of M-RTRS’s column generation is their three-index formulation. Constraint programming and large neighborhood search were also proposed for dial-a-ride problems (e.g., [10] [11]). Progress in communication technologies and the emergence of ride-sourcing and ride-sharing services have stimulated further research in this area. Rolling horizons are often used to batch requests and were used in taxi pooling previously [12, 13]. In addition, stochastic scenarios along with waiting and reallocation strategies have been previously explored in [14, 15]. Bertsimas et al. [16] explored the taxi routing problem (without ride-sharing) and introduced a “backbone” algorithm which increases the sparsity of the problem by computing a set of candidate paths that are likely to be optimal. Alonso-Mora et al. [6] proposed an anytime algorithm which uses cliques to generate vehicle paths combined with a vehicle rebalancing step to move vehicles towards demand. Their “results show that 2,000 vehicles (15% of the taxi fleet) of capacity 10 or 3,000 of capacity 4 can serve 98% of the demand within a mean waiting time of 2.8 min and mean trip delay of 3.5 min.” [6]. Both Alonso-Mora et al. [6] and Bertsimas et al. [16] use hard time windows to reject riders when they cannot serve them quickly enough (e.g., 420 seconds in the aforementioned results). This decision significantly reduces the search space as only close riders can be served by a vehicle. In contrast, M-RTRS provides service guarantees for all riders, while still reducing the search space through a Lagrangian reformulation. The results show that M-RTRS is capable of providing these guarantees while improving prior results in terms of average waiting times. Indeed, for 2,000 vehicles of capacity 4, M-RTRS provides an average waiting time of 2.2 minutes with a standard
deviation of 1.24 and a mean trip deviation of 0.62 minutes (standard deviation 1.13). For 3,000 vehicles of capacity 4, the average waiting time is further reduced to 1.81 minutes with a standard deviation of 1.03 and an average trip deviation of 0.23 minutes.

2.3 Overview of the Approach

M-RTRS divides time into epochs, e.g., time periods of 30 seconds. During an epoch, M-RTRS performs two tasks: It batches incoming requests and it solves the epoch optimization problem for all unserved customers from prior epochs. The epoch optimization takes, as inputs, these unserved customers and their penalties, as well as the first stop of each vehicle after the start of the epoch: Vehicle schedules prior to this stop are committed since, for safety reasons, M-RTRS does not allow a vehicle to be re-routed once it has departed for its next customer. These first stops are called departing stops in this paper. All customers served before and up to the departing stops of the vehicles are considered served. All others, even if they were assigned a vehicle in the prior epoch optimization, are considered unserved.

Once the epoch is completed, a new schedule and a new set of requests are available. The schedule commits the vehicle routes for the entire next epoch and determines their next departing stops. The customer penalties are also updated to make it increasingly harder not to serve them. M-RTRS then moves to the next epoch.

2.4 The Static Problem

This section defines and presents the static (generalized) dial-a-ride problem solved for each epoch. Its objective is to schedule a set of requests on a given set of vehicles while ensuring that no customer deviates too much from their shortest trip time.

The inputs consist primarily of the vehicle and request data. The set of vehicles is denoted by $V$ and each vehicle $v \in V$ is associated with a tuple $(u^v_0, w^v_0, I_v, T^B_v, T^E_v, Q_v)$, where $u^v_0$ is the time the vehicle arrives at its departing stop for the epoch, $w^v_0$ is the num-
ber of passengers currently in the vehicle, $I_v$ is the set of dropoff requests for on-board passengers, $T^B_v$ is the vehicle start time, $T^E_v$ is the vehicle end time, and $Q_v$ is the capacity of the vehicle. In other words, a vehicle $v$ can only insert new requests after time $u^v_0$ and it must serve the dropoffs in $I_v$. The request data is given in terms of a complete graph $G = (N, A)$, which contains the nodes for each possible pickup and delivery. There are five types of nodes: the pickup nodes $P = \{1, \ldots n\}$, their associated dropoff nodes $D = \{n + 1, \ldots 2n\}$, the dropoff nodes $I = \bigcup_{v \in V} I_v$ of the passengers inside the vehicles, the source 0, and the sink $s$ (the last node in terms of indices). Each node $i$ is associated with a number of people $q_i$ to pick up ($q_i > 0$) or drop off ($q_i < 0$) and the time $\Delta_i \geq 0$ it takes to perform them. If $i \in P$, then the corresponding delivery node is $n + i$ and $q_i = -q_{n+i}$. Also, $q_i$ and $\Delta_i$ are zero for the source and the sink. Each node $i \in P$ is associated with a request, which is a tuple of the form $(e_i, o_i, d_i, q_i)$ where $e_i$ is the earliest possible pickup time, $o_i$ is the pickup location, $d_i$ is the dropoff location, and $q_i$ is the number of passengers. Every request $i$ in $I$ is associated with the time $u^P_i$ on which the request was picked up. Every request $i \in P \cup I$ is associated with the shortest time $t_i$ from the request origin to its destination. Finally, the input contains a matrix $(t_{i,j})_{(i,j) \in A}$ of travel times from any node $i$ to any node $j$ satisfying the triangle inequality, the constants $\alpha$ and $\beta$ which constrain the deviation from the shortest path, and the penalty $p_i$ of not serving the request $i \in P$.

A MIP model for the static problem is presented in Figure 2.1. The MIP variables are as follows: $u^v_i$ represents the time at which vehicle $v$ arrives at node $i$, $w^v_i$ the number of people in vehicle $v$ when $v$ leaves node $i$, $x^v_{ij}$ denotes whether edge $(i, j)$ is used by vehicle $v$, and $z_i$ captures whether request $i \in P$ is served. Objective (2.1a) balances the minimization of wait times for every pickups with the penalties incurred by unserved riders. Note that the wait times for riders in $I$ are not included in the objective because these riders are already in vehicles: only the constraints on their deviations must be satisfied. Constraints (2.1b) ensure that only one vehicle serves each request and that, if the request is not served, $z_i$
\[
\min \sum_{i \in P} \sum_{v \in V} (u^v_i - e_i) + \sum_{i \in P} p_i z_i \quad (2.1a)
\]

subject to

\[
\left( \sum_{v \in V} \sum_{j \in \mathcal{N}} x^v_{ij} \right) + z_i = 1 \quad \forall i \in P \quad (2.1b)
\]

\[
\sum_{j \in \mathcal{N}} x^v_{ij} = \sum_{j \in \mathcal{N}} x^v_{ji} \quad \forall i \in \mathcal{N} \setminus \{0, s\}, \forall v \in V \quad (2.1c)
\]

\[
\sum_{j \in \mathcal{N}} x^v_{0j} = 1 \quad \forall v \in V \quad (2.1d)
\]

\[
\sum_{j \in \mathcal{N}} x^v_{j,s} = 1 \quad \forall v \in V \quad (2.1e)
\]

\[
\sum_{j \in \mathcal{N}} x^v_{ij} - \sum_{j \in \mathcal{N}} x^v_{n+i,j} = 0 \quad \forall i \in P, \forall v \in V \quad (2.1f)
\]

\[
\sum_{i \in \mathcal{N}} x^v_{ij} = 1 \quad \forall j \in I_v, \forall v \in V \quad (2.1g)
\]

\[
u^v_j \geq (u^v_i + \Delta_i + t_{ij}) x^v_{ij} \quad \forall i, j \in \mathcal{N}, \forall v \in V \quad (2.1h)
\]

\[
u^v_0 \geq T^B_v \quad \forall v \in V \quad (2.1i)
\]

\[
u^v_s \leq T^E_v \quad \forall v \in V \quad (2.1j)
\]

\[
u^v_i \geq e_i \quad \forall i \in P, v \in V \quad (2.1k)
\]

\[
t_i \leq u^v_{n+i} - (u^v_i + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \quad \forall i \in P, \forall v \in V \quad (2.1l)
\]

\[
t_i \leq u^v_i - (u^v_i + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \quad \forall i \in I_v, \forall v \in V \quad (2.1m)
\]

\[
w^v_j \geq (u^v_i + q_j) x^v_{ij} \quad \forall i, j \in \mathcal{N}, \forall v \in V \quad (2.1n)
\]

\[
0 \leq w^v_i \leq Q_v \quad \forall i \in \mathcal{N}, \forall v \in V \quad (2.1o)
\]

\[
x^v_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{N}, \forall v \in V \quad (2.1p)
\]

Figure 2.1: The Static Formulation of the Dial-A-Ride Problem.

is set to 1 to activate the penalty in the objective. Constraints (2.1c) are flow balance constraints. Constraints (2.1d) and (2.1e) are flow constraints for the source and the sink. Constraints (2.1f) ensure that every request is dropped off by the same vehicle that picks it up. Constraints (2.1g) ensure that every passenger currently in a vehicle is dropped off. Constraints (2.1h) define the arrival times at the nodes. Constraints (2.1i) and (2.1j) ensure that the vehicle is operational during its working hours. Constraints (2.1k) ensure that each
rider is picked up no earlier than its lower bound. Constraints (2.1l) ensure that the travel
time of each served passenger does not deviate too much from the shortest path between
its origin and destination. Passengers are allowed to spend either $\alpha * t_i$ (a percentage of
the shortest path), or $\beta + t_i$ (a constant deviation time from the shortest path) traveling in
the vehicle, whichever is larger. Constraints (2.1m) do the same for passengers already in
a vehicle. Constraints (2.1n) define the vehicle capacities. Lastly, constraints (2.1o) ensure
that the vehicle capacities are not exceeded. Constraints (2.1h) and (2.1n) can be linearized
using a Big $M$ formulation.

The following theorem provides a way to prune the search space significantly. It shows
that, in an optimal solution, a rider cannot be picked up by a vehicle $v$ if the smallest
possible wait time incurred using $v$ is greater than her penalty.

**Theorem 1.** A feasible solution where rider $l$ is assigned to vehicle $v$ such that $u_0^v + t_{0,l} -
eq e_l > p_l$ is suboptimal.

**Proof.** Suppose that there exists a feasible solution (I) that serves a passenger $l$ such that
$u_0^v + t_{0,l} - e_l > p_l$. Let $r$ be the route of vehicle $v$ (i.e., a sequence of edges in $\mathcal{A}$).
Removing the pickup and dropoff of rider $l$ from route $r$ produces a new feasible route $\hat{r}$
since the deviation time cannot increase by the triangular inequality and the number of
riders in $v$ decreases. Solution (II) is derived from solution (I) by replacing the route $r$ by
route $\hat{r}$ and fixing $z_l$ to 1. Using $\hat{u}$ and $\hat{z}$ to denote the variables of solution (II), the cost
Equation 2.2a is just the definition of the objective of solution (II). Inequality (2.2b) is induced by the hypothesis. Inequality (2.2c) is induced by the triangular inequality on the travel times. Inequality (2.2d) just factors the equation to get the objective of solution (I). Solution (I) is thus suboptimal.

2.5 The Column-Generation Algorithm

2.5.1 The Master Problem

The restricted master problem (RMP), presented in Figure 2.2, selects a route for each vehicle. In order for a route to be assigned to a vehicle, the route must contain dropoffs for every current passenger of that vehicle. The set of routes is denoted by $R$ and its subset of routes that can be assigned to vehicle $v$ is denoted $R_v$. The variables in the master problem are the following: $y_r \in [0, 1]$ is set to 1 if potential route $r$ is selected for use and variable $z_i \in [0, 1]$ is set to 1 if request $i$ is not served by any of the selected routes. The constants are as follows: $c_r$ is the sum of the wait time incurred by customers served by route $r$, $p_i$ is the cost of not scheduling request $i$ for this period, and $a_i^r = 1$ if request $i$ is served by route $r$. The objective minimizes the waiting times incurred by all customers on each route and the penalties for the customers not scheduled during the current period. Constraints (2.3c) ensure that $z_i$ is set to 1 if request $i$ is not served by any of the selected
\begin{align*}
\min & \quad \sum_{r \in R} c_r y_r + \sum_{i \in P} p_i z_i \tag{2.3a} \\
\text{subject to} & \quad \left( \sum_{r \in R} y_r a_{ri}^i \right) + z_i = 1 \quad \forall i \in P \quad (\pi_i) \tag{2.3b} \\
& \quad \sum_{r \in R_v} y_r = 1 \quad \forall v \in V \quad (\sigma_v) \tag{2.3c} \\
& \quad z_i \in \mathbb{N} \quad \forall i \in P \quad (\lambda_i) \tag{2.3d} \\
& \quad y_r \in \{0, 1\} \quad \forall r \in R \quad (\rho_r) \tag{2.3e}
\end{align*}

Figure 2.2: The Master Problem Formulation.

routes and constraints (2.3d) ensure that only one route is selected per vehicle. The dual variables associated with each constraint are specified in between parentheses next to the constraint in the model.

### 2.5.2 The Pricing Problem

The routes for each vehicle \( v \) are generated via a pricing problem depicted in Figure 2.3. The pricing problem is defined for a given vehicle \( v \). Theorem 1 makes it possible to remove some passengers from the set \( P \) to obtain the subset \( P_v \) and thus a new graph \( G_v = (N_v, A_v) \). The pricing problem minimizes the reduced cost of the route being generated. Constraints (2.4b) – (2.4o) correspond to constraints (2.1c) – (2.1p) in the static problem.

### 2.5.3 The Column Generation

In traditional column generation for dial-a-ride problems, the pricing problem is formulated as a resource-constrained shortest-path problem and solved using dynamic programming. However, the minimization of waiting times, i.e., \( \sum_{i \in P} (u_i - e_i) \), is particularly challenging, as it cannot be formulated as a classical resource-constrained shortest-path problem. One option is to discretize time and use time-expanded graphs. However, this raises significant computational challenges for large instances. As a result, this paper solves the pricing
\[
\begin{align*}
\text{min} & \quad \sum_{i \in P_v} (u_i - e_i) - \sum_{i \in P_v} \sum_{j \in N_v} x_{ij} \pi_i - \sigma_v \\
\text{subject to} & \quad \sum_{j \in N_v} x_{ij} = \sum_{j \in N_v} x_{ji} \quad \forall i \in N_v \setminus \{0, s\} \\
& \quad \sum_{j \in N_v} x_{0j} = 1 \\
& \quad \sum_{j \in N_v} x_{js} = 1 \\
& \quad \sum_{j \in N_v} x_{ij} - \sum_{j \in N_v} x_{n+i,j} = 0 \quad \forall i \in P_v \\
& \quad \sum_{i \in N_v} x_{ij} = 1 \quad \forall j \in I_v \\
& \quad u_j \geq (u_i + \Delta_i + t_{ij}) x_{ij} \quad \forall i, j \in N_v \\
& \quad u_0 \geq T_B^v \\
& \quad u_s \leq T_E^v \\
& \quad u_i \geq e_i \quad \forall i \in P_v \\
& \quad t_i \leq u_{n+i} - (u_i + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \quad \forall i \in P_v \\
& \quad t_i \leq u_i - (u_i^P + \Delta_i) \leq \max\{\alpha t_i, \beta + t_i\} \quad \forall i \in I_v \\
& \quad w_j \geq (w_i + q_j) x_{ij} \quad \forall i, j \in N_v \\
& \quad 0 \leq w_i \leq Q_v \quad \forall i \in N_v \\
& \quad x_{ij} \in \{0, 1\} \quad \forall i, j \in N_v
\end{align*}
\]
Algorithm 1: COLUMN GENERATION

while true do
  C ← GENERATE_COLUMNS()
  if C = ∅ then
    break;
  Solve RMP after adding C

Function GENERATE_COLUMNS() :
  k ← 1
  while k ≤ |P| do
    C ← GENERATE_SIZE_COLUMNS(k)
    if C ≠ ∅ then
      return C
    else
      k++

Function GENERATE_SIZE_COLUMNS(k) :
  Q ← \{R ⊆ P | |R| = k\}
  forall v ∈ |V| ordered by decreasing σ_v
  R_v ← arg min_{R ∈ Q} PRICING(v, R)
  if PRICING(v, R_v) < 0 then
    Q ← \{R ⊆ Q | R ∩ R_v = ∅\}
  return \{ROUTE(v, R_v) | v ∈ V & PRICING(v, R_v) < 0\}

in waves, first generating columns with one customer before progressively increasing the number of considered requests. Procedure GENERATE_COLUMNS (lines 6 – 12) generates columns by increasing number of requests. Procedure GENERATE_SIZE_COLUMNS (lines 13 – 18) generates columns of size k, where k is the number of requests in the column. It first computes Q, a set in which each element is a k-sized set of possible requests. It then considers the various vehicles ranked in decreasing order of their dual values σ_v. Line 15 computes the sets of requests with the smallest pricing objective value. If the pricing objective is negative (line 16), all set of requests which contains a request covered by R_v are removed from Q to ensure that M-RTRS generates a set of non-overlapping columns at each iteration (line 17). Finally, line 18 returns the routes for each vehicle with negative reduced costs.
2.6 The Real-Time Problem

M-RTRS divides the time horizon into epochs of length $\ell$, i.e., $[0, \ell), [\ell, 2\ell), [2\ell, 3\ell), \ldots$ and epoch $\tau$ corresponds to the time interval $[\tau \ell, (\tau + 1) \ell)$. During period $\tau$, M-RTRS batches the incoming requests into a set $P_\tau$, which is considered in the next epoch. It also optimizes the static problem using the requests accumulated in $P_{\tau-1}$ and those requests not yet committed to in the epochs $\tau - 1$ and before. The optimization is performed over the interval $[(\tau + 1) \ell, \infty)$.

It remains to specify how to compute the inputs to the optimization problem, i.e., the departing stops and times for each vehicle and the various set of requests to serve. To determine the starting stop for a vehicle $v$, the optimization in epoch $\tau$ uses the solution $\phi_{\tau-1}$ to the static problem in epoch $\tau - 1$ and considers the first stop $s_v$ in $\phi_{\tau-1}$ in the interval $[(\tau + 1) \ell, \infty)$ if it exists. This stop becomes the starting stop $u_v^0$ of the vehicle and its earliest time is given by the earliest departure time of vehicle $v$ in $\phi_{\tau-1}$. If vehicle $v$ is idle at stop $s_v$ in $\phi_{\tau-1}$ and not scheduled on $[(\tau + 1) \ell, \infty)$, then the departing stop is $s_v$ and the earliest departing time is $(\tau + 1) \ell$. Consider now the sets $P$, $D$, and $I_v$ ($v \in V$) for period $\tau$. For a vehicle $v$, all the requests before its departing stop $s_v$ are said to be committed and are not reconsidered. The set $I_v$ are the dropoffs of the requests that have been picked up before $s_v$ but not yet dropped off. The set $P$ corresponds to the requests that have not been picked up by any vehicle $v$ before $s_v$, as well as the requests batched in $P_{\tau-1}$. The set $D$ simply contains the dropoffs associated with $P$.

Finally, since the static problem may not schedule all the requests, it is important to update the penalty of unserved requests to ensure that they will not be delayed too long. The penalty for an unserved request $c$ in period $\tau$ is given by $p_c = \delta 2^{(\tau \ell - e_c)/(100)}$ and it increases exponentially over time as shown in Figure 2.4. The $\delta$ parameter incentivizes the schedule of the request in its first available period. Figure 2.4 displays the function for $\delta = 420$ seconds and $\ell = 30$ seconds: It ensures that the penalty doubles every ten periods.
Observe that the static model schedules all the requests which have not been committed to any vehicle. This gives a lot of flexibility to the real-time system at the cost of more complex pricing subproblems.

2.7 Experimental Results

2.7.1 Instance Description

M-RTRS was evaluated on the yellow trip data provided by the New York City Taxi and Limousine Commission (NYCTLC) [9]. This data provides pickup and dropoff locations, which were used to match trips to the closest virtual stops, starting times, which were used as the request time, and the number of passengers. This section reports results on a representative set of 24 instances, 1 hour per day for two weekdays per month from July 2015 through June 2016. To capture the true difficulty of the problem, rush hours (7–8am) were selected. The instances have an average of 21,326 customers and range from 6,678 customers to 28,484 customers. Individual requests with more customers than the capacity of the vehicles were split into several trips. An additional test was performed on the largest instance with 32,869 customers.
2.7.2 Virtual Stops

The evaluation assumes a dial-a-ride system using the concept of virtual stops proposed in the RITMO system [7] (Uber and Lyft are now considering similar concepts). Virtual stops are locations where vehicles can pick up and drop off customers without impeding traffic. They also ensure that customers are ready to pick up and make ride-sharing more efficient since they decrease the number of stops. To implement virtual stops, Manhattan was overlaid with a grid with cells of 200 squared meters and every cell had a virtual stop. The trip times were pre-computed by querying OpenStreetMap for travel times between each virtual stop [17]. All customers at a virtual stop are grouped and can be picked up together.

2.7.3 Algorithmic Setting

Both the final master problem and the restricted master problem are solved using Gurobi 8.1. Empty vehicles are initially evenly distributed over the virtual stops. The pricing problem uses parallel computing to implement line 15 of algorithm 1, exploring potential requests simultaneously. To meet real-time constraints, the implementation greedily extends the “optimal” routes of size $k$ to obtain routes of size $k + 1$. Unless otherwise specified, all experiments are performed with the following default parameters: 2,000 vehicles of capacity 5, $\alpha = 1.5$, $\beta = 240$ seconds, and $\delta = 420$ seconds. The impact of these parameters is also studied.

2.7.4 Wait Times

Figure 2.5 reports the distribution of the waiting for all customers across all instances. The results demonstrate the performance of M-RTRS: The average waiting time is about 2.58 minutes with a standard deviation of 1.31. On the instance with 32,869 customers, the average waiting time is 5.42 minutes.
2.7.5 Trip Deviation

Figure 2.6 depicts a histogram of trip deviations incurred because of ride-sharing. The results indicate that riders have an average trip deviation of 0.34 minutes with a standard deviation of 0.74. In percentage, this represents a deviation of about 12%. On the instance with 32,869 customers, the average trip deviation is 2.23 minutes, which shows the small overhead induced by ride-sharing.

2.7.6 The Impact of the Fleet Size

Figure 2.7 studies the impact of the fleet size on the waiting times and trip deviation. The plot reports the average waiting times for various numbers of riders, where capacity is $4, \alpha = 1, \beta = 840$ seconds, and $\delta = 420$ seconds to facilitate comparisons to Alonso-Mora et al. [6]. The results show that, even with 1,500 vehicles, the average waiting time remains below 6 minutes and the average deviation time below 40 seconds. Since M-RTRS is guaranteed to serve all the requests, these results demonstrate the potential of column generation and ride-sharing for large-scale real-time dial-a-ride platforms. Adopting M-RTRS has the potential to substantially reduce traffic in large cities, while still guaranteeing service within reasonable times. Recall that the approach in Alonso-Mora et al. [6] does not serve about 2% of the requests.
2.7.7 The Impact of Vehicle Capacity

Figure 2.8 studies the impact of the vehicle capacity (i.e., how many passengers a vehicle can carry) on the average waiting times and trip deviation. The parameters are set to 2,000 vehicles, $\alpha = 1$, $\beta = 840$ seconds, and $\delta = 420$ seconds to facilitate comparisons to Alonso-Mora et al. [6]. The results on waiting times show that moving to vehicles of capacity 8 further reduces the average waiting times, especially on the large instances. On the other hand, moving from a capacity 5 to 3 does not affect the results too much. The results on deviations are more difficult to interpret. Obviously moving to a capacity 8 further increases the deviation (although it remains below one minute). However, moving to vehicles of capacity 3 also increases the deviation, which is not intuitive. This may be a consequence of myopic decisions that cannot be corrected easily given the tight capacity.

2.7.8 The Impact of the Penalty

The penalty $p_i$ in the model is an exponential function of the current waiting time of customer $i$. Constant $\delta$ controls the initial penalty: If it is too small, the penalty for not scheduling a request for the first few periods is low, which causes an increase in wait times, as can be observed in Figure 2.9. Once $\delta$ is large enough, the average wait times converge to the same values.
2.7.9 Final Vehicle Assignments

As a result of re-optimization, the vehicle to which a rider is assigned can change. Figure 2.10 reports the amount of time until riders receive their final vehicle assignment (the vehicle which actually picks them up). Not surprisingly, this histogram closely follows the waiting time distribution. The majority of riders receive this assignment quickly. However, it takes some riders over 10 minutes to receive their final vehicle assignment, which shows that M-RTRS takes advantage of the ability to re-assign riders to vehicles which will result in better overall assignments.
2.7.10 The Impact of Column Generation

Figure 2.11 depicts the impact of column generation and reports the number of columns in the final MIP as all possible columns of sizes 1 and 2 to be conservative. The results show that the algorithm only explores a small percentage of all potential columns, demonstrating the benefits of a column-generation approach.

2.7.11 The Impact of Pruning

Figure 2.12 shows the impact of Theorem 1, which provides a way to prune the number of requests considered at each step of the algorithm. The figures report the total optimization time for all time periods of each instance. Each optimization must be performed in less than 30 seconds, but the graph reports the total optimization time over the entire hour. As the results indicate, the pruning benefits become substantial as the instance sizes grow. The results show that the pruning significantly reduces the computational time. They also show that M-RTRS should be able to handle even larger instances since, after exploiting Theorem 1, M-RTRS uses only about a sixth of the available time. This creates opportunities to exploit stochastic information.
2.7.12 The Impact of Ride Sharing

Figure 2.13 reports the average number of people in each vehicle at all times for each instance. The results show a significant amount of ride sharing, although single trips and idle time remain a significant portion of the rides, especially when the fleet is oversized. Lastly, Figure 2.8 shows that wait times are reduced by a factor of 4 when moving from single-rider trips to ride-sharing for large instances while the trip deviation only increases to at most 2 minutes for vehicles of capacity 8, thus demonstrating the value of ride sharing.

2.7.13 Comparison with Prior Work

The results of Alonso-Mora et al. [6] “show that 2,000 vehicles (15% of the taxi fleet) of capacity 10 or 3,000 of capacity 4 can serve 98% of the demand within a mean waiting time of 2.8 min and mean trip delay of 3.5 min.” M-RTRS relaxes the hard time-windows present in Alonso-Mora et al. [6] and improves on these results, yielding an average wait time of 2.2 minutes with only 2,000 vehicles, while guaranteeing service for all riders.

2.8 Conclusion

This paper considered the real-time dispatching of large-scale ride-sharing services over a rolling horizon. It presented M-RTRS, a real-time optimization framework that divides
the time horizon into epochs and uses a column-generation algorithm that minimizes wait times while guaranteeing services for every rider and a small trip deviation compared to a direct trip. This contrasts to earlier work which rejected customers when the predicted waiting time was considered too long (e.g., 7 minutes). This assumption reduced the search space at the cost of rejecting a significant number of requests.

The column-generation algorithm of M-RTRS is derived from a three-index formulation Cordeau and Laporte [8] which is adapted for use in real-time dial-a-ride applications. In addition, to ensure that all riders are served in reasonable times, the paper proposed an optimization model that balances the minimization of waiting times with penalties for riders that are not scheduled yet. These penalties are increased after each epoch to make it increasingly harder not to serve waiting riders. The paper also presented a key property of the formulation that makes it possible to reduce the search space significantly.

M-RTRS was evaluated on historic taxi trips from the New York City Taxi and Limousine Commission [9], which contains large-scale instances with more than 30,000 requests an hour. The results indicated that M-RTRS enables a real-time dial-a-ride service to provide service guarantees (every rider is served in reasonable time) while improving average waiting times and average trip deviations compared to prior work. The results also showed that larger occupancy vehicles bring benefits and that the fleet size can be further reduced while preserving very reasonable waiting times.

Substantial work remains to be done to understand the strengths and limitations of the approach. The current implementation is myopic and heavily driven by the dual costs to generate the columns. Different pricing implementation, including the use of constraint programming to replace our dedicated search algorithm, and the inclusion of stochastic information are natural directions for future research.
Acknowledgments

This research was partly supported by Didi Chuxing Technology Co. and Department of Energy Research Grant 7F-30154. We would like to thank the reviewers for their detailed comments and suggestions which dramatically improved the paper, and the program chairs for a rebuttal period that was long enough to run many experiments.
CHAPTER 3
REAL-TIME DISPATCHING OF LARGE-SCALE RIDE-SHARING SYSTEMS:
INTEGRATING OPTIMIZATION, MACHINE LEARNING, AND MODEL
PREDICTIVE CONTROL

3.1 Introduction

TNCs like Uber and Lyft have fundamentally transformed mobility in many cities, providing on-demand door-to-door transportation through mobile applications. They have also increased traffic in many cities: a recent study by Erhardt et al. [18] showed that, between 2010 and 2016, weekday vehicle hours of traffic delay have increased by 62% in San Francisco. In contrast, it was estimated that the delay increase would have been 22% without TNCs. To address this issue, several cities have begun limiting the number of TNC vehicles on the road. Another way to tackle the underlying congestion and pollution issues is to build mobility systems that utilize ride-sharing systematically. A study by Alonso-Mora et al. [6] showed that systematic ride-sharing may significantly reduce the number of vehicles needed to serve requests. Their results indicate that 98% of the historic demand for taxi services in NYC could be served with a much smaller taxi fleet, while maintaining short wait times. This paper continues this line of research and focuses on how to build a real-time dispatching and routing architecture that serves the needs of large-scale ride-sharing systems. It is envisioned that, in the future, these ride-sharing systems will be deployed using autonomous vehicles, guarantee service to all customers, and leverage advanced AI systems. In the transition period, they can be supported by human drivers provided that these drivers follow the instructions of the ride-sharing system.

The value of stochastic information in real-time vehicle routing has been demonstrated previously by Bent and Van Hentenryck [15]. However, incorporating stochastic informa-
Figure 3.1: The A-RTRS Architecture for Real-Time Dispatching.

tion in large-scale ride-sharing systems, where requests may arrive every tenth of a second during peak times, is a challenge. It is thus not surprising that state-of-the-art approaches are purely myopic [6, 4]. These systems batch requests and optimize frequently to account for real-time demand. Other approaches to real-time dispatching (e.g., [19]) use deep reinforcement learning, but they ignore ride-sharing and do not leverage advanced routing algorithms, focusing only on customer assignment. To incorporate stochastic information, this paper proposes a novel end-to-end framework (A-RTRS) for real-time ride-sharing systems that tightly integrates state-of-the-art optimization techniques, machine learning, and model predictive control.

The A-RTRS architecture is illustrated in Figure 3.1. Time is divided into epochs and, during epoch $t$, A-RTRS optimizes the routing of the requests that were batched in epoch $t - 1$ as well as unserved requests from earlier epochs. Moreover, at a lower frequency and prior to the routing optimization, A-RTRS relocates idle vehicles using a Model Predictive Control (MPC) step. The MPC step does not operate on individual requests for scalability reasons. Rather it works with longer time periods and at a coarser zone level (e.g., taxi zones in New York City), and relies on a machine-learning model to predict the number
of requests between each pair of zones over time. The main contribution of A-RTRS is to demonstrate that, in large-scale real-time ride-sharing systems, hybridizing state-of-the-art optimization algorithms for fine-grained routing decisions with model predictive control for idle vehicle relocation at a coarser space and time granularity provides significant operational benefits. Indeed, results on historic taxi trips from the New York City Taxi and Limousine Commission [9], indicate that this tight integration decreases average waiting times by about 30% over all test cases and reaches close to a 32% reduction in average waiting times for high-demand zones in the most challenging instances.

This paper is organized as follows. Section 3.2 presents the problem. Section 3.3 presents related work. Section 3.4 gives an overview of A-RTRS. Section 3.5 describes the dial-a-ride optimization. Sections 3.6 and 3.7 present the core conceptual contributions of the paper: the demand forecasting model and the vehicle relocation scheme. Section 3.8 reports the experimental results and section 3.9 concludes the paper.

### 3.2 Problem Statement

Operating a real-time ride-sharing system requires solving large-scale dial-a-ride problems, where each request corresponds to a trip for a number of riders from an origin to a destination that must take place after a specified pickup time. Constraints limit the time a rider can spend inside a vehicle (ride duration constraints) and the number of riders in a vehicle at any one time (vehicle capacity constraints). The goal is to serve all requests and minimize the average waiting time, while satisfying the ride duration and capacity constraints. Special attention is also devoted to ensure that no request is left unserved indefinitely. The systems studied in this paper either use a fleet of autonomous vehicles or their own pool of drivers who follow routing instructions exactly. The system can thus relocate the vehicles at will in order to anticipate demand. It is assumed that significant historical data is available and can be used to forecast demand at the zone level.
3.3 Related Work

A comprehensive review of popular dial-a-ride formulations which serve as the foundation of A-RTRS can be found in [8]. A-RTRS uses a rolling horizon, alternating request batching and optimization, as traditionally used in taxi pooling [12, 13]. Alonso-Mora et al. [6] were first to demonstrate the value of ride-sharing in NYC: they showed that 98% of the historic demand could be served with a smaller taxi fleet and short wait times. Their anytime algorithm uses cliques to generate vehicle routes and hard-time windows to discard requests which cannot be served efficiently. A linear program is employed to move idle vehicles towards discarded requests in order to better serve those areas in the future. Lowalekar et al. [20] improve over Alonso-Mora et al. [6] by partitioning the region into zones and assigning vehicles to zone paths. M-RTRS is the first algorithm designed to serve all requests with small waiting times: they use column generation to serve all requests with smaller number of vehicles and shorter waiting times. Their dial-a-ride algorithm is used as the dispatching engine of A-RTRS. To the author’s knowledge, no algorithm other than the one presented in chapter 2 provides guarantees to serve all requests: They can decide to ignore arbitrary requests. Note that these three algorithms are myopic: they do not exploit information about future requests. Iglesias et al. [21] proposed a model predictive control approach for serving individual requests at the zone level, combining a machine-learning model (based on deep learning) and a mixed-integer program for request assignments and vehicle relocation. They did not consider ride-sharing and their dispatching algorithm is performed at a coarse granularity. This paper leverages and generalizes their model predictive control approach. Ma et al. [22] integrated dispatching optimization and model predictive control for scheduling requests in a multimodal transit systems: they do not batch requests, use a single period for vehicle relocation, and assume Poisson arrivals for each zone. The dispatching of each request uses local search and insertion heuristics. The benefits of demand prediction and vehicle relocation has been demonstrated by Bent and Van
Hentenryck [14, 15] for various types of vehicle routing problems (using online stochastic optimization) and by Yu and Shen [23] for on-demand ride-pooling, using approximate dynamic programming. Holler et al. [19] used deep learning and bipartite matching for dispatching and vehicle relocation: their approach does not support ride sharing. Shah et al. [24] enhance Alonso-Mora et al. [6] with an approximation of the future reward learned using a deep neural network. They provide improvements over Alonso-Mora et al. [6] when the ride duration is twice as long as the shortest path. However, the approach does not provide service guarantees and does not minimize waiting times. It also rejects requests even when vehicles are available, which can be problematic to justify in practice. In contrast, this paper serves all requests with an average waiting time of 2.5 minutes with 2,000 vehicles during peak times and a more realistic ride-duration constraint (50% increase). The socio-economic benefits of ride-sharing systems is explored by Bistaffa et al. [25]. To the authors’ knowledge, this paper is the first integration of advanced optimization techniques, machine learning, and model predictive control for the real-time vehicle dispatching and relocation of large-scale ride-sharing systems.

3.4 Overall Organization

This section gives an overview of the A-RTRS architecture. As depicted in Figure 3.1, A-RTRS divides time into epochs of length $\ell^A$, i.e., $[0, \ell^A)$, $[\ell^A, 2\ell^A)$, $[2\ell^A, 3\ell^A)$, $\ldots$. During epoch $\tau$, A-RTRS batches incoming requests and performs an optimization that assigns prior requests to vehicles and routes them. The requests considered in this optimization are those batched in epoch $\tau - 1$, as well as unserved requests from earlier epochs. Periodically, A-RTRS performs a relocation optimization, which exploits a forecasting model to direct idle vehicles towards expected demand.
3.4.1 The Optimization Problem

The optimization problem receives as inputs a set of requests, each of which is characterized by its origin and destination, its earliest pickup time, and its number of riders. The optimization has at its disposal a number of vehicles. Each vehicle is characterized by its departure location, its earliest departure time, its capacity, and its set of riders. Each rider is characterized by her dropoff location and the time she has already spent in the vehicle.

The starting location and departure time of a vehicle are given by the current state of the mobility system. If a vehicle is idle in the existing schedule, its starting location is its current position and its departure time is the beginning of the epoch (i.e., $\tau^{\ell^4}$). If a vehicle is serving customers, its starting location is the first location it visits after the start of the epoch and the departure time is specified accordingly. The riders associated with a vehicle are those who have already been picked up and need to be dropped off. Hence, for every epoch, the optimization problem considers all the requests whose riders have not been picked up yet, while also scheduling the dropoffs of existing riders. Note that the optimization problems associated with two successive epochs may schedule a request differently as long as the request’s riders have not been picked up. This gives a lot of flexibility to the real-time system at the cost of more complex optimization problems.

Given the computational complexity of the dial-a-ride problem that must be solved in real time, the optimization may not be able to serve all requests for some epochs. Hence, following M-RTRS, A-RTRS associates a penalty with each request to ensure that the request is served in reasonable time. The penalty is increased after each epoch in which the request is not served. The optimization model minimizes a weighted sum of the average waiting time and the penalties associated with unserved requests.

3.4.2 Vehicle Relocation

Every $\omega$ epochs, A-RTRS performs a relocation of vehicles at the zonal level (e.g., taxi zones, census tracks, or traffic analysis zones). The goal is to determine the desired number
of idle vehicles to move from zone $i$ to zone $j$ over the next period $\left(\tau \ell^A, \tau \ell^A + \ell^R\right)$, where $\ell^R$ is the length of the relocation period and is significantly larger than the epoch length $\ell^A$. As a result, the relocation optimization operates at a much coarser granularity both in space and time.

This combination of micro- and macro-decisions for routing and relocation is one of the salient features of A-RTRS and is driven by the reality of the large-scale real-time ride-sharing systems, where the number of requests in each epoch makes it difficult computationally to exploit forecasting information during the routing decisions.

3.4.3 Forecasting the Demand

To inform the vehicle relocation, A-RTRS is assumed to have at its disposal historical data for the number of requests from zone $i$ to zone $j$ for every time period of length $\ell^R$. This historical data is used to train a forecasting model of the demand.

3.5 The Dial-A-Ride Optimization

During each epoch, A-RTRS solves a generalized dial-a-ride optimization specified in section 3.4. To perform this task, it borrows the algorithm presented in chapter 2, which is briefly reviewed in this section. The dial-a-ride optimization is based on a column generation that operates at the route level. A vehicle route specifies a sequence of pickups and dropoffs which satisfies the ride duration constraints and the vehicle capacity. The column generation interleaves the solving of (the linear relaxation of) a Restricted Master Problem (RMP), which selects routes, and pricing subproblems which generate new routes for each vehicle. The process terminates when no new routes can improve the solution of the RMP or the time limit for the column generation is met. The last stage of the dial-a-ride optimization is a mixed-integer program that solves the RMP exactly for the generated routes. The pricing subproblems are complex due to their objective of minimizing waiting times. As a result, traditional dynamic programming formulations are not effective and Riley et
\[ \min \sum_{r \in R} c_r y_r + \sum_{i \in P} p_i z_i \quad (3.1a) \]

subject to \( \left( \sum_{r \in R} y_r a_i^r \right) + z_i = 1 \quad \forall i \in P \quad (3.1b) \)

\[ \sum_{r \in R_v} y_r = 1 \quad \forall v \in V \quad (3.1c) \]

\[ z_i \in \mathbb{N} \quad \forall i \in P \quad (3.1d) \]

\[ y_r \in \{0, 1\} \quad \forall r \in R \quad (3.1e) \]

Figure 3.2: The Restricted Master Problem Formulation.

...al. [4] use an anytime exact algorithm that generates routes of increasing lengths.

The RMP is depicted in Figure 3.2. In the formulation, \( R \) denotes the set of routes, \( R_v \) denotes the subset of possible routes for vehicle \( v \), \( c_r \) represents the wait times incurred by all customers served by route \( r \), \( p_i \) is the penalty of not scheduling request \( i \) for this epoch, and \( a_i^r = 1 \) iff request \( i \) is served by route \( r \). The RMP uses the following decision variables: \( y_r \in [0, 1] \) is 1 iff route \( r \) is selected and \( z_i \in [0, 1] \) is 1 iff request \( i \) is not served by any of the selected routes. The objective function minimizes the waiting times of the served customers and the penalties for the unserved customers. Constraints (3.1b) ensure that \( z_i \) is set to 1 if request \( i \) is not served by any of the selected routes and constraints (3.1c) ensure that only one route is selected per vehicle.

Since the dial-a-ride optimization may not schedule all the requests, it is important to update the penalty of unserved requests to ensure that they will not be delayed too long. For the penalty for an unserved request \( c \) in epoch \( \tau \), Riley et al. [4] use \( p_c = \rho \frac{2(\tau^A - e_c)}{(10^A)} \), where \( e_c \) is the earliest possible pickup time for request \( c \). The \( \rho \) parameter was tuned to incentivize the algorithm to schedule each incoming request in its first available epoch.
3.6 Demand Forecasting

Forecasting the demand from zone \( i \) to zone \( j \) over time may be challenging in some settings, since this demand may be sparse for some zones and historical data may be limited. To address this difficulty, A-RTRS proceeds in two steps: it first predicts the number of requests in a zone \( z \) in time period \( t \) and then approximates the zone-to-zone demand.

3.6.1 Preprocessing

Let \( d_{zt} \) be the demand for zone \( z \) during period \( t \). In the case study, the time series \( \{d_{zt}\}_t \) is strongly non-stationary (the mean and variance vary over time). As a result, the forecasting model first stationarizes the time series by differencing it over a week period. More precisely, the forecasting model defines \( \delta_{zt} = d_{zt} - d_{z(t-n_r \times 7)} \), where \( n_r \) is the number of periods in a day, and predicts the differenced demand \( \delta_{zt} \) instead of \( d_{zt} \).

3.6.2 Vector Autoregression

To forecast the time series \( \{\delta_{zt}\}_t \), A-RTRS uses Vector Autoregression (VAR), a multivariate generalization of Autoregression (AR). In VAR, the expected value of a multivariate time series at a particular period is assumed to be a linear function of the value of the time series at previous time steps.

The prediction for the differenced demand in zone \( z \) in period \( t \) uses not only \( z \)'s demand in prior periods but also the differenced demands of \( z \)'s adjacent zones. Let \( N(z) \) denote the zones adjacent to \( z \) and \( d = |N(z)| + 1 \). Let vector \( \Delta_{zt} \in \mathbb{R}^d \) denote the weekly-differenced demands of \( z \) and its adjacent zones in period \( t \). Each element in \( \Delta_{zt} \) is an element in \( \{\delta_{zt}\}_{z \in N(z) \cup \{z\}} \). \( \delta_{zt} \) can then be modeled as

\[
\delta_{zt} = \phi_{zt-1} \Delta_{zt-1} + \cdots + \phi_{zt-k} \Delta_{zt-k} + \eta
\]

(3.2)

where \( \phi_{zt} \) is a row vector in \( \mathbb{R}^d \) and \( \eta \) is a white noise with zero mean. The coefficients \( \phi_{zt} \)
are estimated using least square regression and the order $k$ is selected based on the Akaike information criterion (AIC).

Once the parameters have been estimated, the prediction for the differenced demand of $z$ in period $t$ is given by

$$
\bar{\delta}_{zt} = \bar{\phi}_{zt-1}\Delta_{zt-1} + \cdots + \bar{\phi}_{zt-k}\Delta_{zt-k}
$$

The demand prediction for zone $z$ at time $t$ is then given by

$$
\bar{\lambda}_{zt} = d_{z(t-n_r \times 7)} + \bar{\delta}_{zt}.
$$

### 3.6.3 Destination Assignment

Given the number of requests for zone $z$ in period $t$, the trip destinations for these requests are assigned using historical distributions. A-RTRS uses an historical distribution for each hour during the weekdays and the weekend days. For example, when predicting the demand in zone $z$ during the 7–8am period on a Wednesday, if 70 percent of the trips originating from $z$ during this period on weekdays have their destination in zone $b$ in historical data, then the number of trips going from $z$ to $b$ will be $0.7\bar{\lambda}_{zt}$ rounded to the nearest non-negative integer. This returns the final demand prediction $\bar{\lambda}_{ijt}$ for the requests from zone $i$ to zone $j$ at $t$.

### 3.7 Idle Vehicle Relocation

The idle vehicle relocation process is run every $\omega$ epochs and considers periods of length $\ell^R$, i.e., $[0, \ell^R), [\ell^R, 2\ell^R), \ldots$ It has at its disposal the zone to zone demand forecasts for each period. It proceeds in two steps: (1) It first uses a Model Predictive Control (MPC) approach to find the desired number of vehicles in each zone; (2) It then selects the vehicle relocation assignments to minimize the relocation cost.
3.7.1 Zone Rebalancing

The MPC approach in this section is borrowed from Iglesias et al. [21] and generalized to real-time ride-sharing systems, where multiple riders can share a vehicle. Its goal is to determine the number of idle vehicles to move from zone \( i \) to zone \( j \) during the next period \((\tau \ell^A, \tau \ell^A + \ell^R)\) in order to minimize the average waiting time in the dial-a-ride optimizations. To achieve this goal, the MPC approach solves an assignment optimization at the zone level over multiple time periods. Hence, in contrast to the dial-a-ride optimization, it works at a coarser granularity and looks ahead in the future using the demand forecasting module.

The MPC approach uses a MIP model (MPC-MIP) over \( T \) periods, each of length \( \ell^R \). Let \( \mathcal{T} = \{0, \ldots, T - 1\} \). For each period \( t \in \mathcal{T} \), MPC-MIP takes as input \( \bar{\lambda}_{ijt} \), the forecasted demand originating in zone \( i \) with a destination in zone \( j \) at time \( t \), as well as a variety of information about the current state of the system. In particular, \( w_{ij} \) is the current sharing ratio for requests from zone \( i \) to zone \( j \) in the system (e.g., \( w_{ij} = 1 \) means that riders are alone in the vehicle, while \( w_{ij} = 1.5 \) means an average of 1.5 passengers per vehicle); \( A_{it} \) is the set of vehicles that will become idle in zone \( i \) during period \( t \) (estimated from routes of current vehicles) and \( tt_{ij} \) is the average number of periods it takes to move from a stop in zone \( i \) to a stop in zone \( j \).

The MPC-MIP decision variables are: the number \( x^e_{ijt} \) of empty vehicles to move from zone \( i \) to zone \( j \) starting at period \( t \); the number \( x^p_{ijt} \) of vehicles with passengers moving from \( i \) to \( j \) starting at period \( t \); and the number \( u_{ijt} \) of requests originating in zone \( i \) and ending in zone \( j \) which are not served at period \( t \). The objective (3.3a) minimizes the number of unserved requests while also enforcing a small penalty on moving vehicles. This penalty ensures that vehicles prefer to stay in their current zone if that current zone is expected to need them in the future. Constraints (3.3b) and (3.3c) are the flow balance constraints for requests. Constraints (3.3d) are the flow conservation constraints for vehicles.
3.7.2 Vehicle Relocation

MPC-MIP returns the number \( \bar{x}_{ij0}^{r} \) of vehicles that should move from zone \( i \) to zone \( j \) in period \( t \). Only the relocations in period \( \bar{x}_{ij0}^{r} \) are relevant for A-RTRS at this point in time. However, A-RTRS must now identify \( \bar{x}_{ij0}^{r} \) specific vehicles to relocate. This is performed by another MIP model (VR-MIP), which receives the following inputs: \( \bar{x}_{ij0}^{r} \), the sets \( A_{i0} \) of idle vehicles in zone \( i \), the time \( c_{vj} \) to move vehicle \( v \) to its closest stop in zone \( j \). VR-MIP decides whether a vehicle is relocated to a zone: Variable \( y_{vj} \) is 1 if vehicle \( v \) is chosen to relocate to zone \( j \). The VR-MIP objective (3.4a) minimizes the sum of the traveling times, Constraints (3.4b) ensure the correct number of relocations from zone \( i \) to zone \( j \), and Constraints (3.4c) ensure that a vehicle does not relocate to more than one zone.
3.8 Experimental Results

3.8.1 Case Study

A-RTRS is evaluated on the yellow trip data obtained via the NYCTLC [9]. The NYCTLC dataset provides the number of passengers for each trip and the start time of each trip, which is used as both the request time and the lower bound on the earliest pickup time. This section reports results on 24 instances, two hours each day for two days per month from July 2015 through June 2016. Rush hours (7–9am) were selected to obtain instances that are computationally challenging. The instances have an average of 48,100.5 customers and range from 19,276 to 59,820 customers. Individual requests with more customers than the vehicle capacity are split into several trips. Following Riley et al. [4], in order to ease ride sharing, avoid curb management issues, and reduce the number of stops, Manhattan is represented by a grid with cells of 200 squared meters. Each such cell represents a pickup/dropoff location. The travel time matrix for the network of locations was then precomputed by querying OpenStreetMap [17]. For every trip, the locations of the origin and destination were obtained by selecting the closest locations to their pickup and dropoff points in the NYCTLC dataset.

3.8.2 Runtime Configurations

A-RTRS is compared to its myopic version M-RTRS which has no idle vehicle relocation and is essentially the approach proposed by chapter 2. It is also compared to OA-RTRS, a version of A-RTRS using perfect information on future requests instead of the machine-learning predictions. Unless otherwise specified, all experiments were performed with the following default parameters for A-RTRS (and M-RTRS when relevant): 2000 vehicles of capacity 4, a maximum deviation from the shortest path determined by $\max\{\alpha t_c, \beta + t_c\}$, where $t_c$ is the shortest possible path from customer $c$’s origin to their destination, $w_{ij} = 1.2$, $\rho = 420$, $\alpha = 1.5$, $\beta = 240$ seconds, $\ell^A = 30$ seconds, $\ell^R = 300$ seconds, and $\omega = 10$. 
Table 3.1: Average Waiting Times by Instance Sizes.

<table>
<thead>
<tr>
<th>Number of Requests</th>
<th>M-RTRS</th>
<th>A-RTRS</th>
<th>OA-RTRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 40,000</td>
<td>2.33</td>
<td>2.37</td>
<td>2.15</td>
</tr>
<tr>
<td>40,000 – 50,000</td>
<td>3.83</td>
<td>2.40</td>
<td>2.41</td>
</tr>
<tr>
<td>50,000 &lt;</td>
<td>3.78</td>
<td>2.56</td>
<td>2.56</td>
</tr>
</tbody>
</table>

epochs. Empty vehicles are initially distributed evenly over the locations. The demand was forecasted at the hour level and scaled down uniformly due to the sparse demand in some of the zones. All MIP models are solved using Gurobi 8.1 [26].

3.8.3 Reduction in Waiting Times

Figures 3.5 and 3.6 report the distributions of the waiting times incurred by all customers across all instances with a logarithmic y-axis. The results demonstrate that A-RTRS reduces waiting times across the vast majority of trips. It reduces the average waiting times from 3.64 to 2.51 minutes (a 30% improvement) while also decreasing the standard deviation from 1.48 to 1.16.

Figures 3.7, 3.8, and 3.1 go into more details and report results on each instance and by instance sizes. The results show that the benefits of relocation are more significant for instances with a large number of requests. A-RTRS strongly dominates M-RTRS for instances with large demands, but is not as effective on instances with relatively low demand. This last result is due to the accuracy of the machine-learning algorithm as highlighted in Figure 3.8. With perfect information, OA-RTRS improves over A-RTRS over low-demand instances, but A-RTRS and OA-RTRS behave similarly on high-demand instances.

3.8.4 Zonal Information

Figures 3.9 and 3.10 describe where the reductions in waiting times occur. Together, they show that relocation benefits most the zones with significant demand, where the improve-
Figure 3.5: Histograms of Waiting Times for A-RTRS and M-RTRS (Logarithmic Y-Scale).

Figure 3.6: Histograms of Waiting Times for A-RTRS and OA-RTRS (Logarithmic Y-Scale).

...ments can reach almost 55%. Figure 3.11 is reassuring from a fairness standpoint: Zones with low demand keep low average waiting times under relocation.
3.8.5 Passenger Information

Figures 3.12 and 3.13 show that relocation slightly decreases ride sharing which is beneficial for passengers. There is less of a need to ride share to satisfy the demand and minimize
Figure 3.9: Percentage Improvement in Waiting Times by Origin.

Figure 3.10: Total Number of Requests By Zone (over all instances).

Figure 3.11: Average Waiting Times by Zone.
Figure 3.12: Average Number of Passengers per Vehicle.

Figure 3.13: Average Number of Passengers per Vehicle.

waiting times.
3.8.6 Vehicle Information

Figure 3.14 depicts the histograms of idle times per vehicles. It shows that M-RTRS has more vehicles with no idle time and more vehicles with high idle times. A-RTRS is more balanced.

3.8.7 Relocation

Figure 3.15 shows that most vehicles spend less than five minutes in relocation and the vast majority of vehicles spend less than 17% of their operating hours relocating. Figure 3.16 shows that OA-RTRS relocates more on the instances with lower demands, explaining why it improves over A-RTRS on these instances. Having a better demand predictor is thus an important research direction.

3.9 Conclusion

This paper proposed A-RTRS, an end-to-end framework for real-time optimization of large-scale ride-sharing systems. A-RTRS combines demand forecasting, state-of-the-art opti-
mization, and model predictive control to dispatch, route, and relocate vehicles in real-time, minimizing average waiting times. The mobility system provides service guarantees (i.e., it serves all requests), enforces a ride-duration constraint (i.e., no passenger travels more than 50% over their shortest path), minimizes waiting times, while achieving reasonable
waiting times through penalties increasing over time. Experiments using historic taxi trips in New York City indicate that this integration decreases average waiting times by about 30% over all test cases and reaches close to 55% on the largest instances for high-demand zones compared to a base line without relocation. On the NYC case study, A-RTRS serves all requests in reasonable time and with an average waiting of 2.51 minutes with a standard deviation of 1.16, using a fleet of 2,000 vehicles of capacity 4. The results also demonstrate that, while zones with large demand see the most benefits, zones with low demand maintain low waiting times and that the vast majority of vehicles spend less than 17% of their operating time relocating. In summary, A-RTRS demonstrates that, in large-scale real-time ride-sharing systems, hybridizing state-of-the-art optimization algorithms for fine-grained routing decisions with model predictive control for idle vehicle relocation at a coarser space and time granularity provides significant operational benefits. Future research will be devoted in improving the machine-learning algorithm, since more accurate predictions will enable a better performance on instances with relatively fewer requests.
CHAPTER 4
MARTA REACH: THE TECHNOLOGY BEHIND ON-DEMAND PUBLIC TRANSPORTATION

4.1 Introduction

When designing public transit service, transit agencies face a trade-off between quality of service and total area covered. The “ridership model” seeks to maximize public transportation adoption in limited area by providing exceptional service through frequent bus and metro routes. In contrast, the “coverage model” seeks to maximize the possible origin and destinations served by the public transit network at a cost to route frequency. Once a balance between these competing objectives is chosen, transit agencies have an increasing number of options for providing service, including: traditional metro, traditional bus, bus rapid transit, and demand responsive transit. When coverage is desired, DRT can be a cheaper option than traditional bus lines. For this reason, many transit agencies, including MARTA, are exploring DRT.

MARTA Reach is a DRT pilot created through a collaboration between MARTA and the Georgia Institute of Technology. Four areas of the Atlanta metropolitan area were selected: Belvedere, Gillem, North Fulton, and West Atlanta. Each service area is optimized separately using its own set of vehicles. In each area, many virtual stops were hand placed, because MARTA needed to verify each location for rider and driver safety. In prior chapters we assumed that a vehicle could idle at any virtual stop while it waited for a new assignment of requests to serve. However, in practice, certain virtual stops are impossible to idle at due to traffic, safety, or land ownership considerations. For instance, a virtual stop may be placed in a shopping plaza parking lot, but the owners of that plaza may not allow DRT vehicles to idle there for extended periods. Therefore, in each service area a subset of
the virtual stops were selected as vehicle idle locations. The vehicle dispatching algorithm needs to move any vehicle idling at a non-idle stop to these idle stops.

Idle vehicle relocation could be accomplished by moving a vehicle to the closest valid idle location, however, in practice this results in vehicles idling near popular destinations instead of near popular origins. Another option is to use the MPC presented in chapter 3, modifying the Vehicle Relocation MIP shown in 3.4 to only relocate to idle vehicles to idle stops. This presents an issue: predicting future demand is challenging because there is no historic ridership data for a new pilot and any zones created in these areas would have low ridership relative to the New York City instances presented in chapter 3. Therefore, a new algorithm, which relocates idle vehicles to idle stops, using only the last few hours of requests as data, was designed.

The rest of this chapter is organized as follows: section 4.2 gives an overview of the optimization framework, section 4.3 presents the algorithm, section 4.4 presents experiments on New York City for the sake of comparison with the work in previous chapters, and section 4.5 provides concluding remarks.

4.2 Overview of the Approach

This section gives an overview of the RR-RTRS architecture. The optimization for the dial-a-ride problem is exactly as described in chapter 2. For an overview of the assignment problem and the optimization framework, see subsection 3.4.1. The only difference from the previously described framework is that instead of only performing relocation every few epochs, we now perform relocation every single epoch, to prevent vehicles from idling at invalid idle locations.

After the passenger assignment decisions have been generated, at the end of each epoch, RR-RTRS performs a relocation of idle vehicles. Any vehicle that does have any current passengers or passenger assignments, given the current epoch’s generated passenger assignments, is considered idle. Vehicle relocation instructions are generated via a two step
process. First the desired number of vehicles at each idle stop is determined and second, the idle stop to which to relocate each vehicle is determined. These decisions are sent to each vehicle at the end of the epoch, at the same time as passenger assignment decisions are sent to non-idle vehicles.

4.3 Idle Vehicles and Relocation

This section presents the vehicle relocation operation. The objective is to relocate idle vehicles towards areas of high demand, while also ensuring that vehicles only idle at stops which are in the set of valid idle stops $S_{idle}$.

An idle vehicle, which does not have any current passengers nor future passengers, is denoted as $v \in V_{idle}$. There is a subset of virtual stops where a vehicle is allowed to idle, denoted $S_{idle} \subseteq S$ where $S$ is the set of all virtual stops. The set of idle vehicles that are not currently at an idle stop is $\hat{V}_{idle}$. The total number of idle vehicles not currently at an idle stop is given by $|\hat{V}_{idle}|$. Given the set of trip requests made in the last hour, $(p_r, d_r, p^\text{idle}_r) \in R$ where $p_r \in S$ is the pickup location, $d_r$ is the dropoff location, and $p^\text{idle}_r$ is the closest idle stop to the pickup location $p_r$. The number of requests where $p^\text{idle}_r$ equals an arbitrary valid idle stop is given by equation 4.1.

$$c_s = \sum_{r \in R} (p^\text{idle}_r == s) \quad \forall s \in S_{idle}$$

Let the array $C$ store the values of $c_s$ for every idle stop $s \in S_{idle}$.

4.3.1 Determining the Desired Number of Vehicles per Stop

The model in Figure 4.1 determines the desired number of vehicles, $v_s$, at each idle stop $s \in S_{idle}$. The objective 4.2a minimizes the sum of the ratio of requests, $c_s$, to vehicles, $z_s$, a proxy for $v_s$, at each idle stop. Constraint 4.2b ensures that the sum of the desired number of vehicles at each stop is equal to the total number of idle vehicles available. The
Algorithm 2: RATIO REBALANCING

1 forall $i \in \{0, ..., |\hat{V}_{\text{idle}}|\}$
2 \[ R \leftarrow \text{COMPUTE RATIOS}(\mathcal{V}, \mathcal{C}) \]
3 \[ x \leftarrow \arg \max_{\mathcal{V}} R \]
4 \[ \mathcal{V} \leftarrow \{\mathcal{V}_{0}, ..., \mathcal{V}_{x-1}, \mathcal{V}_{x} + 1, ... \mathcal{V}_{|\mathcal{V}_{idle}|-1}\} \]
5 return $\mathcal{V}$

Function \text{COMPUTE RATIOS} $(\mathcal{V}, \mathcal{C})$:

6 \[ R \leftarrow 0^{1, |\mathcal{S}_{idle}|} \]
7 forall $s \in \mathcal{S}_{idle}$
8 \[ x \leftarrow \text{if } v_{s} = 0 \text{ then } 0.5 \text{ else } v_{s} \]
9 \[ R \leftarrow \{R_{0}, ..., R_{s-1}, \frac{c_{s}}{x_{s}}, R_{s+1}, ..., R_{|\mathcal{V}_{idle}|}\} \]
10 return $R$

Domain 4.2d ensures that $v_{s}$ is a natural number, while the constraint 4.2c ensures that when $v_{s} = 0$, the division in the objective remains valid. This constraint also serves to double the penalty for assigning zero vehicles to an idle stop, compared to assigning one.

\[
\begin{align*}
\min \quad & \sum_{s \in S_{idle}} \frac{c_{s}}{z_{s}} \\
\text{subject to} \quad & \sum_{s \in S_{idle}} v_{s} = |\hat{V}_{_idle}| \\
& z_{s} = \max\{0.5, v_{s}\} \quad \forall s \in S_{idle} \\
& v_{s} \in \mathbb{N} \quad \forall s \in S_{idle} \quad (4.2d)
\end{align*}
\]

Figure 4.1: The Ratio Rebalancing Optimization (VR-Ratio).

This model can be solved to optimality with a greedy algorithm, as shown in algorithm 2.

4.3.2 Vehicle Relocation

The ratio rebalancing algorithm returns the number $v_{s}$ of vehicles desired at stop $s$. However, we must now identify $v_{s}$ specific vehicles to relocate to stop $s$. This is performed by a MIP model (RVR-MIP), which receives the following inputs: $v_{s}$, the set $V_{idle}^{s}$ of idle
\[
\begin{align*}
\min & \sum_{v \in \hat{V}_{idle}} \sum_{s \in S_{idle}} c_{vs} y_{vs} \quad (4.3a) \\
\text{subject to } & \sum_{v \in \hat{V}_{idle}} y_{vs} = v_s \quad \forall s \in S_{idle} \quad (4.3b) \\
& \sum_{s} y_{vs} \leq 1 \quad \forall v \in \hat{V}_{idle} \quad (4.3c) \\
& y_{vs} \in \{0, 1\} \quad \forall v \in \hat{V}_{idle}, s \in S_{idle} \quad (4.3d)
\end{align*}
\]

Figure 4.2: The Vehicle Relocation Optimization (RVR-MIP).

vehicles desired at each stop \(s\), and the time \(c_{vs}\) to move vehicle \(v\) to each idle stop \(s \in S_{idle}\).

RVR-MIP decides to which stop, \(s \in S_{idle}\), each vehicle \(v \in \hat{V}_{idle}\) should be relocated. Variable \(y_{vs}\) is 1 if vehicle \(v\) is chosen to relocate to stop \(s\). The RVR-MIP objective (4.3a) minimizes the sum of the traveling times, Constraints (4.3b) ensure the correct number of vehicles end up at stop \(s\). and Constraints (4.3c) ensure that a vehicle does not relocate to more than one stop.

4.3.3 Tie-breaking

Algorithm 2 has an issue, there may be ties when taking the argmax on line 3. Therefore, a tie-breaking procedure is necessary which algorithm 3 facilitates. However, breaking ties at this level could result in poor relocation decisions. Suppose there is a tie between idle stop \(A\) and idle stop \(B\), and that each already has two vehicles allocated. Suppose also that, currently there are 3 vehicles located in proximity to stop \(A\), and no vehicles near stop \(B\). In this scenario, to minimize vehicle relocation time, and thus maximize vehicle availability, the tie-breaking procedure should allocate the next vehicle to stop \(A\). This example illustrates two key concepts: the first, that tie-breaking only matters when \(|X| > S_{idle} - \sum_{s \in S_{idle}} R_s\) where \(S_{idle} - \sum_{s \in S_{idle}} R_s\) is the remaining idle vehicles the algorithm has left to assign; and the second, that tie-breaking in this case should occur with knowledge of individual vehicle locations. For this reason,
Algorithm 3 identifies tie-breaking decisions and leaves it up to the vehicle relocation optimization presented in Figure 4.3 to make the decisions. On line 3 \( x \) is changed to the set \( X \) of all idle stops where the ratio is maximized. If the number of remaining vehicles is greater than the number of tied idle stops \( |X| \) the algorithm picks one such \( x \in X \) and proceeds normally, as they will all be selected to receive an additional vehicle in subsequent iterations. If the number of remaining vehicles is less than the number of tied idle stops, then the tied idle stops are recorded in the set \( S^{tie}_{idle} \) and the algorithm terminates.

**Algorithm 3: Ratio Rebalancing with Tie-breaking**

1. for all \( i \in \{0, ..., \hat{V}_{idle}\} \)
2. \( R \leftarrow \text{COMPUTE RATIOS} (V, C) \)
3. \( X \leftarrow \text{args max}_s R \) \quad // \( X \) is a set of all \( s \) where \( R_s \) is maximal
4. if \( |X| > S_{idle} - \sum_{s \in S_{idle}} R_s \) then return \((X, V)\)
5. \( x \leftarrow X_0 \)
6. \( V \leftarrow \{V_0, ..., V_{x-1}, V_x + 1, V_{x+1}, ..., V_{|V|-1}\} \)
7. return \((\{\}, V)\)

**Function** \( \text{COMPUTE RATIOS} (V, C) : \)

8. \( R \leftarrow 0, 1, |S_{idle}| \)
9. for all \( s \in S_{idle} \)
10. \( x \leftarrow \text{if } v_s = 0 \text{ then } 0.5 \text{ else } v_s \)
11. \( R \leftarrow \{R_0, ..., R_{s-1}, \frac{v_s}{x_s}, R_{s+1}, ..., R_{|V_{idle}|}\} \)
12. return \( R \)

The ratio rebalancing algorithm with tie-breaking returns the number \( v_s \) of vehicles desired at stop \( s \), along with the set of stops \( X \) that desire an additional vehicle. Therefore, RVR-MIP is modified such that for all stops \( s \in X \), either \( v_s \) or \( v_s + 1 \) vehicles could be assigned, as shown in constraints (4.4c) and constraints (4.4d). This flexibility allows tie-breaking to happen in a way that minimizes total travel time.
\begin{align}
\text{min} & \quad \sum_{v \in V_{\text{idle}}} \sum_{s \in S_{\text{idle}}} c_{vs} y_{vs} & \quad (4.4a) \\
\text{subject to} & \quad \sum_{v \in V_{\text{idle}}} y_{vs} = v_s & \quad \forall s \in S_{\text{idle}} \setminus X & \quad (4.4b) \\
& \quad \sum_{v \in V_{\text{idle}}} y_{vs} \geq v_s & \quad \forall s \in X & \quad (4.4c) \\
& \quad \sum_{v \in V_{\text{idle}}} y_{vs} \leq v_s + 1 & \quad \forall s \in X & \quad (4.4d) \\
& \quad \sum_{s} y_{vs} = 1 & \quad \forall v \in V_{\text{idle}} & \quad (4.4e) \\
& \quad y_{vs} \in \{0, 1\} & \quad \forall v \in V_{\text{idle}}, s \in S_{\text{idle}} & \quad (4.4f)
\end{align}

Figure 4.3: The Vehicle Relocation Optimization with tie-breaking (TBVR-MIP)

### 4.4 Experimental Results

To facilitate comparison to previous chapters, RR-RTRS is evaluated on the yellow trip data obtained via the NYCTLC [9]. For an overview of the experimental design, see subsection 3.8.1. In practice, selecting idle stop locations is a matter of human judgement, taking into consideration many factors such as traffic, land use, and safety. However, the selection of idle locations can have a large impact on system performance. Idle stops at which more requests originate are more beneficial than idle stops with less. To simulate the possible variance in the usefulness of idle stops selected in practice, three different sets of idle stops were tested. To create each set of idle stops, one stop per taxi zone was selected. The sets, $S_{\text{idle}}^{\text{min}}$, $S_{\text{idle}}^{\text{median}}$, and $S_{\text{idle}}^{\text{max}}$, were generated by selecting the stop in each zone that had the least, median, and most amount of requests originating from them over all 24 instances. When using $S_{\text{idle}}^{\text{min}}$, $S_{\text{idle}}^{\text{median}}$, and $S_{\text{idle}}^{\text{max}}$ we denote the algorithm as RR-RTRS (min), RR-RTRS (median), and RR-RTRS (max) respectively. All parameters for RR-RTRS, A-RTRS, and M-RTRS were kept the same as in chapter 3. For an overview of these constants, see subsection 3.8.2. All MIP models are solved using Gurobi 9.5 [26]. The upgrade of Gurobi, combined with tweaks to a few implementation details, and newer
processors, result in slightly better performance for M-RTRS and A-RTRS than was reported in the previous two chapters.

4.4.1 Reduction in Waiting Times

Figures 4.4 and 4.5 report the distributions of the waiting times incurred by all customers across all instances with a logarithmic y-axis. With a pessimistic selection of idle locations, the results demonstrate that A-RTRS reduces waiting times across the vast majority of trips – reducing the average waiting times from 3.38 to 2.74 minutes (a 19% improvement) while also decreasing the standard deviation from 1.37 to 1.19. A-RTRS further reduces the wait time to 2.44 minutes with a standard deviation of 1.09, outperforming all tested selections of idle stop locations and demonstrating the value of predicting future demand.

Figures 4.6 and 4.7 report the distributions of the wait times for RR-RTRS when using \( S_{\text{median}} \) and \( S_{\text{max}} \), comparing them to the previously reported \( S_{\text{min}} \) case. Using \( S_{\text{median}} \) and \( S_{\text{max}} \) RR-RTRS reduces average waiting times to 2.73 minutes and 2.56 minutes respectively.
Figure 4.5: Histograms of Waiting Times for A-RTRS and RR-RTRS (Logarithmic Y-Scale).

Figure 4.6: Histograms of Waiting Times for RR-RTRS (min) and RR-RTRS (median) (Logarithmic Y-Scale).

Figures 4.8 and 4.9 go into more detail and report results on each instance and by instance size in terms of number of passengers. The results show the benefits of relocation
Figure 4.7: Histograms of Waiting Times for RR-RTRS (min) and RR-RTRS (max) (Logarithmic Y-Scale).

Figure 4.8: Waiting Times for M-RTRS, A-RTRS, and RR-RTRS by instance size.

are more significant for instances with a large number of requests.
4.4.2 Vehicle Information

Figure 4.10 shows idle time per vehicles. It shows that RR-RTRS has vehicles idling the longest, because, as stated, above, it adopts a conservative relocation scheme, only relocating vehicles which are not currently at idle locations. M-RTRS has vehicles idle less often, but offers worse performance, which indicates that the places vehicles are idling, controlled exclusively by the dispatching algorithm presented in chapter 2, are not conducive to serving requests quickly. A-RTRS has vehicles idle the least, in part due to the aggressive relocation strategy which ensures vehicles are close to demand.

4.4.3 Relocation

Figure 4.11 shows that most vehicles spend less than five minutes in relocation and the vast majority of vehicles spend less than 16% of their operating hours relocating. Figure 4.12 shows that RR-RTRS relocates a relatively constant amount, no matter the instance size. This is because it only considers vehicles not at valid idle locations, and therefore has less
Figure 4.10: Idling Times per Vehicle.

Figure 4.11: Relocation Time per Vehicle.

vehicles with which it can assign a rebalancing decision.
4.5 Conclusion

This paper proposed RR-RTRS, a framework for real-time optimization of large-scale ride-sharing systems. RR-RTRS combines state-of-the-art optimization and a more conservative relocation scheme which only relocates vehicles which are not at valid idle locations. The mobility system provides service guarantees (i.e., it serves all requests), enforces a ride-duration constraint (i.e., no passenger travels more than 50% over their shortest path), minimizes waiting times, while achieving reasonable waiting times through penalties increasing over time. Experiments using historic taxi trips in New York City indicate that this using pessimistic idle stops, this relocation scheme decreases average waiting times by about 19% when compared to not using any relocation. On the NYC case study, RR-RTRS serves all requests in reasonable time and with an average waiting of 2.74 minutes with a standard deviation of 1.19, using a fleet of 2,000 vehicles of capacity 4.
Appendices
REFERENCES


VITA

Connor Riley was born on July 6, 1994, to parents Emily and Matthew Riley in Connecticut. He earned a Bachelor of Science in Engineering for Computer Science and Engineering at the University of Connecticut and studied Operations Research at both University of Michigan and the Georgia Institute of Technology, where he earned his Doctor of Philosophy in Operations Research. While at the Georgia Institute of Technology, he designed and implemented the backend software for MARTA Reach, an on-demand ride-sharing pilot for the city of Atlanta. Connor currently resides in New York City, with his cat Nala.