STOCHASTIC MODELING AND CONTROL OF AUTONOMOUS MOBILITY-ON-DEMAND SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING AND THE COMMITTEE ON GRADUATE STUDIES OF STANFORD UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Ramón Darío Iglesias
August 2019
© 2019 by Ramon Dario Iglesias. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/mm997fz9077
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ram Rajagopal, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Rishee Jain

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

The last decade saw the rapid development of two major mobility paradigms: Mobility-on-Demand (MoD) systems (e.g. ridesharing, carsharing) and self-driving vehicles. While individually impactful, together they present a major paradigm shift in modern mobility. Autonomous Mobility-on-Demand (AMoD) systems, wherein a fleet of self-driving vehicles serve on-demand travel requests, present a unique opportunity to alleviate many of our transportation woes. Specifically, by combining fully-compliant vehicles with central coordination, AMoD systems can achieve system-level optimal strategies via, e.g., coordinated routing and preemptive dispatch. This thesis presents methods to model, analyze and control AMoD systems. In particular, special emphasis is given to develop stochastic algorithms that can cope with the uncertainty inherent to travel demand.

In the first part, we present a steady-state modeling framework built on queueing networks and network flow theory. By casting the system as a multi-class BCMP network, the framework provides analysis tools that allow the characterization of performance metrics for a given routing policy, in terms, e.g., of vehicle availabilities, and first and second order moments of vehicle throughput. Moreover, we present a scalable method for the synthesis of routing policies, with performance guarantees in the limit of large fleet sizes. The framework provides a large set of modeling options, and specifically address cases where the operational concerns of congestion and battery charge level are considered. We validate our theoretical results on a case study of New York City.

In the second part, we leverage the insights provided by the steady-state models to present real-time control algorithms. Specifically, we cast the real-time control problem within a stochastic model predictive control framework. The control loop consists of a forecasting generative model and a stochastic optimization subproblem. At each time step, the generative model first forecasts a finite number of travel demand for a finite horizon and then we solve the stochastic subproblem via Sample Average Approximation. We show via simulation that this approach is more robust to uncertain demand and vastly outperforms state-of-the-art fleet-level control algorithms. Finally, we validate the presented frameworks by deploying a fleet control application in a carsharing system in Japan. The application uses the aforementioned algorithms to provide, in real-time, tasks to the carsharing employees regarding actions to be taken to better meet customer demand. Results show significant improvement over human based decision making.
To Nour.
Acknowledgments

First, I will like to thank my advisor, Professor Marco Pavone, who through the years has provided the right amount of guidance, pressure, support, and encouragement. His valuable lessons were not only in the realms of control theory and robotics, but also in the realms of strategy and communication, among many others. For this, and more, I am very grateful.

I thank Professor Ram Rajagopal and Professor Rishee Jain for serving in the reading committee of this thesis and providing valuable feedback on its content. I would like to thank also Professor Rebecca Diamond and Professor Mac Schwager for serving on the defense committee. The ensuing discussion provided one of the richest discussions I’ve witnessed on this topic.

Modern work is never achieved in a vacuum, and I would like to thank my numerous collaborators who directly or indirectly contributed to this thesis. Special thanks to my co-authors Federico Rossi and Matthew Tsao. The numerous vivid brainstorming sessions and long all-nighters were among the most valuable experiences of my PhD and among the most cherished. I would like to also thank my co-authors and collaborators Rick Zhang, Mahnoosh Alizadeh, Tommy Heng, Ross Ryder, Stephen Zoepf, Bryan Casey, Kiril Solovey, and Mauro Salazar. Additionally, I am thankful to collaborators from industry, in particular Stephen Hughes, Masanori Yamato, and Akira Matsui, as well as the Toyota Motor Company and the Toyota Research Institute.

I am grateful to the members of the Autonomous System Lab, past and present, who provided a cheerful and inspiring work environment. Special thanks to Gema, Nadja and the rest of the team at Coupa Cafe in Y2E2, who provided not only quality coffee but also comic relief when most needed.

I would like to thank my friends: Jorge, Jose and Andrea for making my residence a home; Rikel and Sven for the endless, uncensored discussions; and the broader Game of Burgers family for what seems to be a lifetime of laughter.

I would like to thank my family: my parents, Clara and Ramon, for teaching me to get back on the horse after every fall; my siblings, Clarissa and Migue, for bringing me back to earth; and my grandfather, Alberto, for embodying the meaning of family and duty.

Finally, I will like to thank Nour. She has been present from the very beginning of this journey, and has experienced it as much as I did. Whatever sacrifices I made, she endured. Whatever counsel I needed, she gave. I can not help but feel deeply grateful for the years of love, support and joy.
I acknowledge the support of the National Science Foundation under CAREER Award CMMI-1454737 and the Toyota Research Institute (TRI). This thesis solely reflects the opinions and conclusions of its authors and not NSF, TRI, or any other entity.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract</strong></td>
<td>v</td>
</tr>
<tr>
<td><strong>Acknowledgments</strong></td>
<td>vii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 The Future of Personal Mobility</td>
<td>1</td>
</tr>
<tr>
<td>1.2 AMoD: Objectives and concerns</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Contribution</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Structure</td>
<td>4</td>
</tr>
<tr>
<td><strong>I Queueing-theoretical Models</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>2 BCMP Network Approach to Modeling AMoD Systems</strong></td>
<td>7</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Background Material</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Closed, multi-class BCMP networks</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Asymptotic behavior of closed BCMP networks</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Model Description and Problem Formulation</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1 Autonomous Mobility-on-Demand model</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2 Casting an AMoD system into a BCMP network</td>
<td>13</td>
</tr>
<tr>
<td>2.3.3 Problem formulation</td>
<td>16</td>
</tr>
<tr>
<td>2.3.4 Limitations</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Asymptotically Optimal Algorithms for AMoD Routing</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 Folding of traffic equations</td>
<td>18</td>
</tr>
<tr>
<td>2.4.2 Asymptotically optimal solution</td>
<td>19</td>
</tr>
<tr>
<td>2.4.3 Linear programming formulation and multi-commodity flow equivalence</td>
<td>21</td>
</tr>
<tr>
<td>2.5 Battery Charge Constraints</td>
<td>23</td>
</tr>
<tr>
<td>2.5.1 Casting charge constraints as classes</td>
<td>23</td>
</tr>
</tbody>
</table>
## II Real-time Control

### 3 Model Predictive Control of AMoD Systems

#### 3.1 Introduction

#### 3.2 Model Description and Problem Formulation

- 3.2.1 Optimal Rebalancing Strategy

#### 3.3 Model Predictive Control

- 3.3.1 Algorithm
- 3.3.2 State observation
- 3.3.3 Forecasting
- 3.3.4 Controller
- 3.3.5 Discussion

#### 3.4 Numerical Experiments

- 3.4.1 Dataset
- 3.4.2 Simulation environment
- 3.4.3 Forecasting
- 3.4.4 Detailed results for a single day
- 3.4.5 Comparison for different forecasting horizons
- 3.4.6 Computational complexity

#### 3.5 Conclusions

### 4 Stochastic Model Predictive Control of AMoD Systems

#### 4.1 Introduction

#### 4.2 Model and Problem Formulation

- 4.2.1 Model
- 4.2.2 Problem Formulation
- 4.2.3 Discussion

#### 4.3 Sample Average Approximation Techniques

- 4.3.1 Sample Average Approximation for AMoD control
- 4.3.2 Oracle inequality performance guarantees for SAA
- 4.3.3 Computational Complexity
## 4.4 Totally Unimodular Linear Relaxations

**4.4.1 Bipartite Matching for Servicing Waiting Customers** ........................................ 68
**4.4.2 Network Flow Optimization for Rebalancing Vehicles** .................................... 68

## 4.5 Stochastic optimization for Model Predictive Control of AMoD systems ................. 70

## 4.6 Numerical Experiments ......................................................................................... 70

**4.6.1 Scenarios** ........................................................................................................ 71
**4.6.2 Experimental Design** ..................................................................................... 72
**4.6.3 Forecasting** .................................................................................................... 72
**4.6.4 Results** .......................................................................................................... 73

## 4.7 Conclusions .......................................................................................................... 74

## 5 Real world implementation: Toyota Ha:mo ............................................................... 77

## 5.1 Introduction .......................................................................................................... 77

## 5.2 System Analysis .................................................................................................... 78

**5.2.1 Demand Analysis** .......................................................................................... 78
**5.2.2 Stockout Analysis** ......................................................................................... 81
**5.2.3 Existing Rebalancing Efforts** ......................................................................... 87
**5.2.4 Discussion** ..................................................................................................... 89

## 5.3 System Control .................................................................................................... 89

**5.3.1 Problem formulation** ...................................................................................... 90
**5.3.2 Model predictive control** ............................................................................... 92
**5.3.3 Pilot description** ............................................................................................ 93
**5.3.4 Discussion** ..................................................................................................... 97

## 5.4 Conclusion ........................................................................................................... 99

## 6 Conclusions ............................................................................................................. 101

## 6.1 Summary ............................................................................................................... 101

## 6.2 Future Directions ................................................................................................ 102

## Bibliography ............................................................................................................. 105
List of Tables

3.1  Wait times for each scenario. ........................................... 51
3.2  Optimization Running Time. ............................................ 53
4.1  Results summary for the DiDi scenario. Wait times are in seconds. ............. 74
4.2  Results summary for the NYC scenario. Wait times are in seconds. ............... 74
List of Figures

2.1 BCMP network model of an AMoD system. Diamonds represent infinite-server road links, squares represent the single-server vehicle stations, and dotted circles represent road intersections (playing the role of “connectors” among the queues). 14

2.2 Graphical depiction of a double queue. Vehicles directed to station \( i \) at charge level \( b \) enter the rebalancing queue \( i_R \) with probability \( \beta_{i,b} \); they enter the passenger-serving queue \( i_Q \), with probability \( 1 - \beta_{i,b} \). Note that the probability of joining \( i_R \) (or, equivalently, \( i_Q \)) depends on the SOC. Vehicles entering \( i_R \) switch to a class \( r \), corresponding to a rebalancing request, with probability \( \lambda(r)/\sum_{r' \in O_{i,b,R}} \lambda(r') \). Conversely, vehicles entering \( i_Q \) switch to class \( q \), corresponding to a customer request, with probability \( \lambda(q)/\sum_{q' \in O_i} \lambda(q') \). 25

2.3 Manhattan scenario. Left: modeled road network. Center: Station locations. Right: Resulting vehicular flow (darker flows show higher vehicular presence). 33

2.4 (a) Station availabilities as a function of fleet size for the baseline case. (b) Expected number of vehicles by usage as a function of fleet size for the baseline case. (c) Utilization as a function of fleet size for the most utilized road. The colored band denotes \( \pm 1 \) standard deviation from the mean. (d) Increase in expected travel time for each O-D pair when considering the BPR delay model. 35

2.5 Passenger and rebalancing request throughput distribution for (a) the station with the highest rebalancing rate, and (b) the station with highest number of passenger requests. 36

3.1 Results from the MPC-Perfect (gray), MPC-LSTM (blue), Reactive [Pavone et al., 2012] (orange), and TV-Reactive (green) scenarios at each time step. From top to bottom: Number of waiting customers; number of vehicles carrying passengers; number of vehicles rebalancing; and, number of rebalancing tasks issued. 52

3.2 Mean wait times for different combinations of \( T_{\text{forward}} \) and \( T_{\text{back}} \) 53
4.1 The top plot shows the number of waiting customers as a function of time for both our controller (MPC-LSTM-SAA) in blue and the controller from Chapter 3 in green. Under the operation of our controller, there are significantly fewer waiting customers throughout the day, which is reflected by the waiting times. Unexpectedly, as a price of this improved service, our controller issues more rebalancing requests, since it is planning for many outcomes, as shown by the middle plot. Looking at the middle and bottom plots together, we see that our controller does additional rebalancing precisely when there is significant variance in the demand.

4.2 The top plot shows the number of waiting customers as a function of time for several controllers. As expected, compared to the reactive controller, the predictive controllers have more customers wait at the beginning of the simulation in order to better prepare for the customers appearing later. Leveraging stochastic optimization, MPC-LSTM-SAA outperforms MPC-LSTM-LP and Reactive in terms of mean waiting time. As a tradeoff, the bottom plot shows that the reactive controller issues the least amount of rebalancing tasks, while MPC-LSTM-SAA issues the most.

5.1 The Ha:mo system consists of 62 stations and 126 single-capacity vehicles.

5.2 94% of trips occurred during weekdays, 74% occurred during rush hours (depicted in orange and red in the right figure).

5.3 Top: Average number of departing trips per hour by station for each segment of the day. Bottom: Average number of arriving trips per hour by station for each segment of the day.

5.4 Demand netflow by segment of the day.

5.5 Vehicle vs parking average availability by station.

5.6 Availability by segment of the day. Top: Vehicle availability. Bottom: Parking availability.

5.7 Coverage throughout the day.

5.8 Relationship between departures and vehicle availability.

5.9 Relationship between arrivals and parking availability.

5.10 Fraction of Ha:mo trips for different walking distance buckets. The red line denotes the bottom 1% percentile.

5.11 Walkability graph and clusters.

5.12 Relationship between departures and vehicle availability using walkability clusters.

5.13 Relationship between departures and vehicle availability using walkability clusters.

5.14 Current workflow for Ha:mo system in Toyota City.

5.15 Average number of rebalancing trips per hour.

5.16 Typical rebalancing strategies.

5.17 Main rebalancing effort per segment.
5.19 Rebalancing implementation strategies for the different phases. Solid lines denote communication via the web, dashed lines denote ad-hoc communication, and dotted lines denote physical interaction. Status quo (top): The operator monitors the system using the information gathered from the vehicles and the customers’ reservations. As needed, he or she asks the rebalancers to move vehicles to match the desired number of vehicles per station. Phase 2 (Middle): The operator is still responsible for deciding how the vehicles should be rebalanced; however, the controller provides suggested actions based on information gathered from the customers, vehicles, and rebalancers. Phase 3 (Bottom): The instructions are issued directly to the rebalancers automatically. The operator is still responsible for monitoring the system.

5.24 Left: Parking coverage during the pilot. The red vertical lines denote when the different rebalancing tasks ended. Right: Parking coverage change over the same time period for all weekdays year-to-date. The red vertical line denotes the change observed during the pilot.
Chapter 1

Introduction

This thesis presents a systematic approach to modeling, analyzing and controlling Autonomous Mobility-on-Demand (AMoD) systems within the presence of uncertainty. AMoD systems consist of fleets of self-driving vehicles serving on-demand travel requests within urban areas. In contrast to traditional Mobility-on-Demand systems, such as bike-, ride- and carsharing, the autonomous nature of the vehicles enables system-level coordination. In turn, this allows operators to tackle a variety of operational objectives and concerns that were previously difficult or impossible to address, such as vehicle imbalance or coordinated vehicle charging.

1.1 The Future of Personal Mobility

Personal mobility in the form of privately owned automobiles contributes to increasing levels of traffic congestion, pollution, and under-utilization of vehicles (on average 5% in the US [Neil, 2015]) – clearly unsustainable trends for the future. The pressing need to reverse these trends has spurred the creation of cost-competitive, on-demand personal mobility solutions such as car-sharing (e.g. Car2Go, ZipCar) and ride-sharing (e.g. Uber, Lyft). The convenience and flexibility of these Mobility-on-Demand (MoD) services has helped them become ubiquitous. However, without proper fleet management, car-sharing and, to some extent, ride-sharing systems lead to vehicle imbalances; vehicles aggregate in some areas while becoming depleted in others, due to the asymmetry between trip origins and destinations [Zhang and Pavone, 2016]. Additionally, the growing popularity of ridesharing has made them a major source of traffic congestion in urban areas, for example, [Erhardt et al., 2019] estimates that, between 2010 and 2016, weekday vehicle hours of delay increased by 62% compared to 22% in a counterfactual 2016 scenario without ridesharing.

Self-driving vehicles offer the distinctive advantage of being able to rebalance themselves, in addition to the convenience, cost savings, and possibly safety of not requiring a driver. Indeed, it has
been shown that AMoD systems have the potential to significantly reduce passenger cost-per-mile-traveled, while keeping the advantages and convenience of personal mobility [Spieser et al., 2014]. Moreover, they offer the ability of coordinating their individual tasks, such as travel routes, to achieve system-level objectives. For example, [Rossi et al., 2018] shows that it is possible to avoid increasing congestion due to empty vehicle travel by coordinating routing strategies at the fleet-level.

The main focus of this thesis is to present algorithmic frameworks that build on top of prior work to enable the analysis and real-time control of AMoD systems while accounting for system stochasticity.

1.2 AMoD: Objectives and concerns

Throughout this thesis, we will consider a hypothetical AMoD system operating within a delimited geographical area and with various operational limitations. In particular, we will assume that the number of vehicles in the system is fixed, and that travel demand is a random process which we might or might not be able to characterize exactly. Moreover, travel requests have fixed origin and destinations (i.e. passengers can not update their requests) and are on-demand (passengers can not schedule their requests ahead of time).

We assume that system operator wants to maximize the number of travel requests served, a proxy for revenue in the case of profit-seeking enterprises or for quality-of-service in the case of social-welfare maximizing utilities.

In addition to this objective, the operator might face a variety of operational challenges. First, the operator must deal with the problem of vehicle imbalance, where due to travel demand asymmetry vehicles tend to accumulate in certain regions while deplete in others. To counter this phenomenon, the operator must frequently rebalance the vehicles to maintain acceptable availability across the service area. Second, for sufficiently large AMoD systems, traffic congestion might be a major concern for the operator. In such cases, the operator must coordinate the travel routes of the vehicles to avoid congestion. Third, in the case of electric fleets, the limited range might require coordinated charging to avoid impacting the availability and trip duration faced by the passengers. Moreover, these operational concerns are aggravated by uncertainty present on each of the underlying processes of the system: whether the travel time between an origin and a destination, the amount of battery spent, or the number of travel requests at a given time interval, each of these processes carries its own inherent stochasticity.

1.3 Contribution

The goal of this thesis is to present scalable modeling and control frameworks that capture the aforementioned operational objectives and concerns along with their associated uncertainty. In the
1.3. **CONTRIBUTION**

same vein, the main contributions of this thesis are two-fold: i) a queueing-theoretical framework which provides a large set of modeling options (e.g., the inclusion of road capacities and charging constraints), and ii) a stochastic model predictive control framework which enables data-driven real-time control of large fleets while capturing a variety of operational concerns.

Specifically, in Chapter 2, we show how an AMoD system can be cast within the framework of closed, multi-class BCMP queueing networks. The framework captures stochastic passenger arrivals, vehicle routing on a road network, congestion effects, and battery charging-discharging for electric vehicles. Importantly, such a framework allows one to use a number of queueing theoretical tools to analyze performance metrics for a given routing policy in terms, e.g., of vehicle availabilities and second-order moments of vehicle throughput. Additionally, we propose a scalable method for the synthesis of routing and charging policies: namely, we show that, for large fleet sizes, the stochastic optimal routing and charging strategy can be found by solving a linear program. Finally, we explore the applicability of our theoretical results on a case study of Manhattan. The case study showcases how stakeholders can leverage the proposed framework to derive structural insights into the operations of the AMoD system as well as answer system design questions, such as fleet size, based on operational performance. This chapter was originally published in [Iglesias et al., 2019].

In Chapters 3 and 4, we present the model predictive control framework. First, in Chapter 3, we propose an efficient approach to find the optimal dispatching policy for the case when the trip demand is known ahead of time. This provides an upper bound on the performance of the system. The approach is able to simultaneously optimize the dispatching policy and the number of required vehicles: thus, it can be used for fleet sizing. Second, we propose an MPC algorithm for operating the system in real-time by leveraging short-term forecasts of customer demand. The complexity of the algorithm does not depend on the number of vehicles or on the number of customers in the transportation system: thus, the algorithm can be used to effectively control large-scale AMoD systems. Third, we validate these approaches using a dataset of DiDi Chuxing, the major ridesharing company in China: our results show that the proposed MPC algorithm outperforms a state-of-the-art algorithm with a 91.3% reduction in mean customer wait time. The results of this chapter were originally published in [Iglesias et al., 2018].

In Chapter 4, we extend the MPC algorithm to include uncertainty of the travel demand forecasts. Then, we provide high probability bounds on the suboptimality of the proposed algorithm when competing against an oracle controller which knows the true distribution of customer demand. Finally, we demonstrate through experiments that the proposed algorithm outperforms the aforementioned deterministic counterparts when the demand distribution has significant variance. In particular, on the same DiDi Chuxing dataset, our controller yields a 62.7% reduction in customer waiting time compared to the work presented in Chapter 3. The contents of this chapter originally appeared in [Tsao et al., 2018].

Finally, in Chapter 5, we test the model predictive control framework on Toyota Ha:m0, a
carsharing system in Toyota City, Japan. First, we adjust the algorithm presented in Chapter 3 to capture the operational constraints of a human-based rebalancing strategy (namely, rebalancing the rebalancers themselves) as well as the parking constraints inherent to station-based carsharing systems. Then, we designed and develop a software system including web and mobile interfaces necessary to coordinate the actions of the human rebalancers. We show through a limited pilot increased coverage and availability of the system.

1.4 Structure

The structure of this thesis is as follows. In Part I, consisting of Chapter 2, we focus on the queueing-theoretical framework. In particular, Section 2.2 first presents a background on queueing theory that is leveraged throughout the chapter. Sections 2.3, 2.4 and 2.5 characterize how to cast an AMoD system as a BCMP queueing network and how to synthesize rebalancing, routing and charging policies, while Section 2.6 showcases the merits of the proposed framework via a case study of New York City.

Part II, consisting of Chapters 3, 4 and 5, is an exposition of the MPC framework. In Chapter 3, we present in Section 3.2 a time-varying network flow model under the assumption of known travel requests which then, in Section 3.3, we leverage to develop the core of the MPC algorithm. Section 3.4 presents a simulation-based case study using data from Didi Chuxing with promising results using the MPC algorithm. In Chapter 4, Sections 4.2 through 4.5 extend the MPC algorithm to account for uncertainty of travel demand forecasts via Sample Average Approximation. Section 4.6 then shows via simulation the impact of properly accounting for forecasting uncertainty by revisiting the Didi Chuxing case study. Finally, Chapter 5 presents the work required to implement the MPC algorithm in a real-world system. Specifically, Section 5.2 first introduces and then analyses the current state of the Toyota Ha:mo system, while Section 5.3 presents the control system including the core MPC algorithm and the user interfaces as well as the results of the pilot.

The last chapter, Chapter 6, summarizes the work presented in this thesis and explores new promising avenues of research.
Part I

Queueing-theoretical Models
Chapter 2

A BCMP Network Approach to Modeling and Controlling Autonomous Mobility-on-Demand Systems

In this chapter we present a queueing network approach to the problem of routing and rebalancing a fleet of self-driving vehicles providing on-demand mobility (i.e. an AMoD system) within a capacitated road network. We first cast an AMoD system into a closed, multi-class BCMP queueing network model capable of capturing the passenger arrival process, traffic, the state-of-charge of electric vehicles, and the availability of vehicles at the stations. Second, we propose a scalable method for the synthesis of routing and charging policies, with performance guarantees in the limit of large fleet sizes. Third, explore the applicability of our theoretical results on a case study of Manhattan. Collectively, this chapter provides a unifying framework for the analysis and control of AMoD systems, which provides a large set of modeling options (e.g., the inclusion of road capacities and charging constraints), and subsumes earlier Jackson and network flow models.

2.1 Introduction

A number of works have recently investigated the potential of AMoD systems, with a specific focus on the synthesis and analysis of coordination algorithms.

This chapter aims to devise a general, unifying analytical framework for the analysis and control of AMoD systems, which subsumes many of the analytical models recently presented in the
literature, chiefly, [Pavone et al., 2012], [Zhang and Pavone, 2016], and [Rossi et al., 2018]. Specifically, this chapter extends the Jackson network approach in [Zhang and Pavone, 2016] by adopting a Baskett-Chandy-Muntz-Palacios (BCMP) queueing-theoretical framework [Baskett et al., 1975, Kobayashi and Gerla, 1983]. The generality offered by the BCMP framework allows one to take into account several real-world constraints, in particular (i) state-of-charge of autonomous electric vehicles and (ii) road capacities (that is, congestion). In contrast to previous work, the proposed BCMP model allows one to characterize such effects analytically along with performance guarantees. Moreover, the proposed BCMP model recovers the traffic congestion results in [Rossi et al., 2018], with the additional benefits of taking into account the stochasticity of transportation networks and providing estimates for performance metrics. Thus, the results in this chapter provide novel tools for the analysis and control of AMoD systems in the presence of stochasticity and system-wide constraints such as traffic congestion and vehicle charging.

Literature Review: The issue of vehicle rebalancing has been addressed in a variety of ways in the literature. For example, in the context of bike-sharing, [Chemla et al., 2013] proposes rearranging the stock of bicycles between stations using trucks. The works in [Nourinejad et al., 2015], [BoyacÄś et al., 2015], and [Acquaviva et al., 2014] investigate using paid drivers to move vehicles between car-sharing stations where cars are parked, while [Banerjee et al., 2015] studies the merits of dynamic pricing for incentivizing drivers to move to underserved areas.

Within the context of AMoD systems, where vehicles can rebalance themselves, previous work can be categorized into two main classes: heuristic methods and analytical methods. Heuristic routing strategies are extensively investigated in [Fagnant and Kockelman, 2014], [Fagnant et al., 2016], and [Levin et al., 2016] by leveraging a traffic simulator and, in [Zhang et al., 2016a], by leveraging a model predictive control framework. Analytical models of AMoD systems are proposed in [Pavone et al., 2012], [Zhang and Pavone, 2016], and [Rossi et al., 2018], by using fluidic, Jackson queueing network, and capacitated flow frameworks, respectively. Analytical methods have the advantage of providing structural insights (e.g., [Rossi et al., 2018]), and provide guidelines for the synthesis of control policies. The problem of controlling AMoD systems is similar to the System Optimal Dynamic Traffic Assignment (SO-DTA) problem (see, e.g., [Chiu et al., 2011, Patriksson, 2015]) where the objective is to find optimal routes for all vehicles within congested or capacitated networks such that the total cost is minimized. The main differences between the AMoD control problem and the SO-DTA problem is that SO-DTA only optimizes customer routes, and not rebalancing routes.

Previous work addressing AMoD charging and congestion constraints rely either on deterministic models or are simulation-based studies. Integration of electric vehicles in AMoD systems has been studied in a model-predictive control setting in [Zhang et al., 2016a] and in an agent-based simulation framework in [Chen et al., 2016]. Both studies characterize the effects of charging speed on the level of service via simulations. As for congestion, the impact of AMoD systems on traffic has been a hot topic of debate. For example, [Levin et al., 2016] notes that empty-traveling rebalancing vehicles
may increase congestion and total in-vehicle travel time for customers, but [Rossi et al., 2018] shows that, with congestion-aware routing and rebalancing, the increase in congestion can be avoided. However, their proposed model does not account for the stochasticity of travel demand.

Statement of Contributions: The contribution of this chapter is threefold. First, we show how an AMoD system can be cast within the framework of closed, multi-class BCMP queueing networks. The framework captures stochastic passenger arrivals, vehicle routing on a road network, congestion effects, and battery charging-discharging for electric vehicles. Importantly, such a framework allows one to use a number of queueing theoretical tools to analyze performance metrics for a given routing policy in terms, e.g., of vehicle availabilities and second-order moments of vehicle throughput. Second, we propose a scalable method for the synthesis of routing and charging policies: namely, we show that, for large fleet sizes, the stochastic optimal routing and charging strategy can be found by solving a linear program. Finally, we explore the applicability of our theoretical results on a case study of Manhattan.

Organization: The rest of the chapter is organized as follows. In Section 2.2, we discuss basic properties of BCMP networks. In Section 2.3, we describe the model of the AMoD system in the presence of road congestion constraints, cast it into a BCMP network, and formally present the routing and rebalancing problem. In Section 2.4.2, we focus on the derivation of solution algorithms that are shown to achieve optimal performance in the limit of large fleet sizes. In Section 2.5, we extend the BCMP model and solution algorithms to capture the state-of-charge of the vehicles. We validate our approach in Section 2.6 by performing a case study of Manhattan. Finally, in Section 2.7, we state our concluding remarks and discuss potential avenues for future research.

2.2 Background Material

In this section we review some basic definitions and properties of BCMP networks, on which we will rely extensively later in the chapter.

2.2.1 Closed, multi-class BCMP networks

Let $\mathcal{Z}$ be a network consisting of $N$ independent queues (or nodes). A set of agents move within the network according to a stochastic process, i.e., after receiving service at queue $i$ they proceed to queue $j$ with a given probability. No agent enters or leaves the network from the outside, so the number of agents is fixed and equal to $m$. Such a network is referred to as a closed queueing network. Agents belong to one of $K \in \mathbb{N}_{>0}$ classes, and they can switch between classes upon leaving a node.

Let $x_{i,k}$ denote the number of agents of class $k \in \{1, \ldots, K\}$ at node $i \in \{1, \ldots, N\}$. The state of node $i$, denoted by $\mathbf{x}_i$, is given by $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,K}) \in \mathbb{N}^K$. The state space of the network is
CHAPTER 2. BCMP NETWORK APPROACH TO MODELING AMOD SYSTEMS

[Gelenbe and Pujolle, 1998]:

\[ \Omega_m := \{ (\mathbf{x}_1, \ldots, \mathbf{x}_N) : \mathbf{x}_i \in \mathbb{N}^K, \quad \sum_{i=1}^{N} \|\mathbf{x}_i\|_1 = m \}, \]

where \( \| \cdot \|_1 \) denotes the standard 1-norm (i.e., \( \|\mathbf{x}_i\|_1 = \sum_{k} |x_{i,k}| \)). The relative frequency of visits (also known as relative throughput) to node \( i \) by agents of class \( k \), denoted as \( \pi_{i,k} \), is given by the traffic equations [Gelenbe and Pujolle, 1998]:

\[ \pi_{i,k} = \sum_{k'}^{K} \sum_{j=1}^{N} \pi_{j,k'} p_{j,k';i,k}, \quad \text{for all } i \in \{1, \ldots, N\}, \tag{2.1} \]

where \( p_{j,k';i,k} \) is the probability that upon leaving node \( j \), an agent of class \( k' \) goes to node \( i \) and becomes an agent of class \( k \). Equation (2.1) does not have a unique solution (a typical feature of closed networks), and \( \pi = \{ \pi_{i,k} \}_{i,k} \) only determines frequencies up to a constant factor (hence the name “relative” frequency). It is customary to express frequencies in terms of a chosen reference node, e.g., so that \( \pi_{1,1} = 1 \).

Queues are allowed to be one of four types: First Come, First Served (FCFS), Processor Sharing, Infinite Server, and Last Come, First Served. FCFS nodes have exponentially distributed service times, while the other three queue types may follow any Cox distribution [Gelenbe and Pujolle, 1998]. Such a queueing network model is referred to as a closed, multi-class BCMP queueing network [Gelenbe and Pujolle, 1998].

Let \( \mathcal{N} \) represent the set of nodes in the network. For the remainder of the chapter, we will focus on networks that have only two types of nodes: FCFS queues with a single server (for short, SS queues), forming a set \( \mathcal{S} \subset \mathcal{N} \), and infinite server queues (for short, IS queues), forming a set \( \mathcal{I} \subset \mathcal{N} \). Furthermore, we consider class-independent and load-independent nodes (i.e. nodes whose service rate is independent of the agent classes or number of agents in the queue), whereby at each node \( i \in \{1, \ldots, N\} \) the service rate is given by:

\[ \mu_i(x_i) = c_i(x_i) \mu^0_i, \]

where \( x_i := \|\mathbf{x}_i\|_1 \) is the number of agents at node \( i \), \( \mu^0_i \) is the (class-independent) base service rate, and \( c_i(x_i) \) is the (load-independent) capacity function

\[ c_i(x_i) = \begin{cases} x_i & \text{if } x_i \leq c^0_i, \\ c^0_i & \text{if } x_i > c^0_i, \end{cases} \]

which depends on the number of servers \( c^0_i \) at the queue. In the case considered in this chapter, \( c^0_i = 1 \) for all \( i \in \mathcal{S} \) and \( c^0_i = \infty \) for all \( i \in \mathcal{I} \).


2.2. BACKGROUND MATERIAL

Under the assumption of class-independent service rates, the multi-class network $Z$ can be “compressed” into a single-class network $Z^*$ with state-space $\Omega_m^* := \{(x_1, ..., x_N) : x_i \in \mathbb{N}, \sum_{i=1}^{N} x_i = m\}$ [Kant and Srinivasan, 1992]. Performance metrics for the original, multi-class network $Z$ can be found by first analyzing the compressed network $Z^*$, and then applying suitable scalings for each class. Specifically, let $\pi_i = \sum_{k=1}^{K} \pi_{i,k}$ and $\gamma_i = \sum_{k=1}^{K} \frac{\pi_{i,k}}{\mu_i}$, be the total relative throughput and relative utilization at a node $i$, respectively. Then, the stationary distribution of the compressed, single-class network $Z^*$ is given by

$$P(x_1, ..., x_N) = \frac{1}{G(m)} \prod_{i=1}^{N} \frac{x_i^{\gamma_i}}{\prod_{a=1}^{x_i} c_i(a)} ,$$

where $G(m) = \sum_{x_i \in \Omega^*_m} \prod_{i=1}^{N} \frac{x_i^{\gamma_i}}{\prod_{a=1}^{x_i} c_i(a)}$ is a normalizing constant. Remarkably, the stationary distribution has a product form, a key feature of BCMP networks.

Three performance metrics that are of interest at each node are throughput, expected queue length, and availability. First, the throughput at a node (i.e., the number of agents processed by a node per unit of time) is given by

$$\Lambda_i(m) = \pi_i \frac{G(m-1)}{G(m)} .$$

Second, let $P_i(x_i; m)$ be the probability of finding $x_i$ agents at node $i$; then the expected queue length at node $i$ is given by $L_i(m) = \sum_{x_i=1}^{m} x_i P_i(x_i; m)$.

In the case of IS nodes (i.e., nodes in $\mathcal{I}$), the expected queue length can be more easily derived via Little’s Law as [David, 2012]

$$L_i(m) = \Lambda_i(m)/\mu_i, \quad \text{for all } i \in \mathcal{I}. \quad (2.3)$$

The throughputs and the expected queue lengths for the original, multi-class network $Z^*$ can be found via scaling [Kant and Srinivasan, 1992], specifically, $\Lambda_{i,k}(m) = (\pi_{i,k}/\pi_i)\Lambda_i(m)$ and $L_{i,k}(m) = (\pi_{i,k}/\pi_i)L_i(m)$.

Finally, the availability of single-server, FCFS nodes (i.e., nodes in $\mathcal{S}$) is defined as the probability that the node has at least one agent, and is given by [David, 2012]

$$A_i(m) = \gamma_i \frac{G(m-1)}{G(m)}, \quad \text{for all } i \in \mathcal{S}.$$  

It is worth noting that evaluating the three performance metrics above requires computation of the normalization constant $G(m)$, which is computationally expensive. However, several techniques are available to avoid the direct computation of $G(m)$. In particular, in this chapter we use the Mean Value Analysis method [Gelenbe and Pujolle, 1998].
2.2.2 Asymptotic behavior of closed BCMP networks

In this section we describe the asymptotic behavior of closed BCMP networks as the number of agents $m$ goes to infinity. The results described in this section are taken from [David, 2012], and are detailed for a single-class network. However, as stated in the previous section, results found for a single-class network can easily be ported to the multi-class equivalent in the case of class-independent service rates.

Let $\rho_i := \gamma_i/c_i^o$ be the utilization factor of node $i \in \mathcal{N}$, where $c_i^o$ is the number of servers at node $i$. Assume that the relative throughputs $\{\pi_i\}_i$ are normalized so that $\max_{i \in \mathcal{S}} \rho_i = 1$; furthermore, assume that nodes are ordered by their utilization factors so that $1 = \rho_1 \geq \rho_2 \geq \ldots \geq \rho_N$, and define the set of bottleneck nodes as $\mathcal{G} := \{i \in \mathcal{S} : \rho_i = 1\}$.

It can be shown [David, 2012, p. 14] that, as the number of agents $m$ in the system approaches infinity, the availability at all bottleneck nodes converges to 1 while the availability at non-bottleneck nodes is strictly less than one, that is

$$\lim_{m \to \infty} A_i(m) = \begin{cases} 1 & \forall i \in \mathcal{G}, \\ < 1 & \forall i \notin \mathcal{G}. \end{cases} \tag{2.4}$$

Additionally, the queue lengths at the non-bottleneck nodes have a limiting distribution given by

$$\lim_{m \to \infty} P_i(x_i; m) = \begin{cases} (1 - \rho_i) \rho_i^{x_i} & i \in \mathcal{S}, i \notin \mathcal{G}, \\ e^{-\gamma_i} \frac{\rho_i^{x_i}}{x_i!} & i \in \mathcal{I}. \end{cases} \tag{2.5}$$

Together, (2.4) and (2.5) have strong implications for the operation of queueing networks with a large number of agents, and in particular for the operation of AMoD systems. Intuitively, (2.4) shows that as we increase the number of agents in the network, they will be increasingly queued at bottleneck nodes, driving availability in those queues to one. Alternatively, non-bottleneck nodes will converge to an availability strictly less than one, implying that there is always a non-zero probability of having an empty queue. In other words, agents will aggregate at the bottlenecks and become depleted elsewhere. Additionally, (2.5) shows that, as the number of agents goes to infinity, non-bottleneck nodes tend to behave like queues in an equivalent open BCMP network with the bottleneck nodes removed, i.e., individual performance metrics can be calculated in isolation.

2.3 Model Description and Problem Formulation

In this section, we introduce a BCMP network model for AMoD systems, and formalize the problem of routing and rebalancing such systems under stochastic conditions. Casting an AMoD system as a queueing network allows us to characterize and compute key performance metrics including...
the distribution of the number of vehicles on each road link (a key metric to characterize traffic congestion) and the probability of servicing a passenger request. To emphasize the relationship with the theory presented in the previous section, we reuse the same notation whenever concepts are equivalent.

### 2.3.1 Autonomous Mobility-on-Demand model

Consider a set of stations $S$ distributed within an urban area connected by a network of road links $I$, and $m$ autonomous vehicles providing one-way transportation between these stations for incoming customers. Customers arrive to a station $s \in S$ with a target destination $t \in S$ according to a time-invariant Poisson process with rate $\lambda \in \mathbb{R}_{>0}$. The arrival process for all origin-destination pairs is summarized by the set of tuples $Q = \{(s^{(q)}, t^{(q)}, \lambda^{(q)})\}_{q}$. If on customer arrival there is an available vehicle, the vehicle drives the customer towards its destination. Alternatively, if there are no vehicles, the customer leaves the system (i.e., chooses an alternative transportation system). Thus, we adopt a passenger loss model. Such model is appropriate for systems where high quality-of-service is desired; from a technical standpoint, this modeling assumption decouples the passenger queueing process from the vehicle queueing process.

A vehicle driving a passenger through the road network follows a routing policy $\alpha^{(q)}$ (defined in the next section) from origin to destination, where $q$ indicates the origin-destination-rate tuple. Once it reaches its destination, the vehicle joins the station FCFS queue and waits for an incoming trip request.

A known problem of such systems is that vehicles will inevitably accumulate at one or more of the stations and reduce the number of vehicles servicing the rest of the system [David, 2012] if no corrective action is taken. To control this problem, we introduce a set of “virtual rebalancing demands” or “virtual passengers” whose objective is to balance the system, i.e., to move empty vehicles to stations experiencing higher passenger loss. Similar to passenger demands, rebalancing demands are defined by a set of origin, destination and arrival rate tuples $R = \{(s^{(r)}, t^{(r)}, \lambda^{(r)})\}_{r}$, and a corresponding routing policy $\alpha^{(r)}$. Therefore, the objective is to find a set of routing policies $\alpha^{(q)}, \alpha^{(r)}$, for all $q \in Q, r \in R$, and rebalancing rates $\lambda^{(r)}$, for all $r \in R$, that balances the system while minimizing the number of vehicles on the road, and thus reducing the impact of the AMoD system on overall traffic congestion.

### 2.3.2 Casting an AMoD system into a BCMP network

We are now in a position to frame an AMoD system in terms of a BCMP network model. Initially, we will present the framework in the absence of charging constraints, and in a later section we will extend the model to include them. First, the passenger loss assumption allows the model to

---

1Stations are not necessarily physical locations: they can also be interpreted as a set of geographical regions.
be characterized as a queueing network with respect only to the *vehicles*. In other words, at each station node, vehicles form a queue while waiting for customers and are “serviced” when a customer arrives. Thus, we will henceforth use the term “vehicles” to refer to the queueing agents. From this perspective, the stations $S$ are equivalent to SS queues, and the road links $I$ are modeled as IS queues. The set of all queues is given by $\mathcal{N} = \{S \cup I\}$, in analogy with the Background Material.

Second, we specify the BCMP network model. We abstract the underlying road network and the stations as a directed graph where the edges represent either the road links or the stations, and the vertices represent the road intersections. The BCMP network model can then be derived from such a directed graph as follows. Let $\text{Parent}(i)$ and $\text{Child}(i)$ be the origin and destination vertices of edge $i$ in the directed graph. Then, a road that goes from intersection $j$ to intersection $l$ is represented in the BCMP network model by an IS queue $i \in I$ such that $\text{Parent}(i) = j$ and $\text{Child}(i) = l$. Note that the road may not have lanes in the opposite direction, in which case a queue $i'$ with $\text{Parent}(i') = l$ and $\text{Child}(i') = j$ would not exist. For example, in Figure 2.1, queue 14 starts at vertex 1 and ends at vertex 5. However, there is no queue that connect the vertices in the opposite direction. In turn, we assume that stations are adjacent to road intersections, and therefore stations are represented in the BCMP network model as SS queues $i \in S$ with the same parent and child vertex, i.e., a self-loop. An intersection may have access to either one station (e.g., vertex 2 in Figure 2.1) or zero stations (e.g., vertex 5 in Figure 2.1). Finally, intersections in the BCMP network model are simply conceptual entities playing the role of “connectors” among the queues, see Figure 2.1.

![Figure 2.1: BCMP network model of an AMoD system. Diamonds represent infinite-server road links, squares represent the single-server vehicle stations, and dotted circles represent road intersections (playing the role of “connectors” among the queues).](image)

Third, we introduce classes to represent the process of choosing destinations. We map the set of tuples $Q$ and $R$ to a set of classes $K$ such that $K = \{Q \cup R\}$. Moreover, let $\mathcal{O}_i$ be the subset of classes whose origin $s^{(k)}$ is the station $i$, i.e., $\mathcal{O}_i := \{k \in K : s^{(k)} = i\}$, and $\mathcal{D}_i$ be the subset of classes whose destination $t^{(k)}$ is the station $i$, i.e., $\mathcal{D}_i := \{k \in K : t^{(k)} = i\}$. Thus, the probability that a vehicle at station $i$ will leave for station $j$ with a (real or virtual) passenger is the ratio between the respective (real or virtual) arrival rate $\lambda^{(k)}$, with $s^{(k)} = i$, $t^{(k)} = j$, and the sum of all arrival rates at station $i$. Formally, the probability that a vehicle of class $k$ switches to class $k'$ upon arrival to
2.3. MODEL DESCRIPTION AND PROBLEM FORMULATION

its destination $t^{(k)}$ is

$$p_{t^{(k)}}^{(k')} = \left( \tilde{\lambda}^{(k')}/\tilde{\lambda}_{t^{(k)}} \right),$$

where $\tilde{\lambda}_i = \sum_{k \in \mathcal{O}_i} \lambda^{(k)}$ is the sum of all arrival rates at station $i$. In other words, $\tilde{\lambda}_i$ represents the rate of arrival of passenger and rebalancing requests to station $i$, while $p_{t^{(k)}}^{(k')}$ encodes the likelihood of whether the request is a real passenger or rebalancing task and the desired target destination.

Note that at all times a vehicle belongs to some class $k \in \mathcal{K}$, regardless of whether it is waiting at a station or traveling along the network.

The traversal of a vehicle from its source $s^{(k)}$ to its destination $t^{(k)}$ is guided by a routing policy $\alpha^{(k)}$. This routing policy consists of a matrix of transition probabilities. Let $\mathcal{W}_i = \{ j \in \mathcal{N} : \text{Parent}(j) = i \}$ be the set of queues that begin at vertex $i$, and $\mathcal{U}_i = \{ j \in \mathcal{N} : \text{Child}(j) = i \}$ be the set of queues that end at vertex $i$. A vehicle of class $k$ leaves the station $s^{(k)}$ via one of the adjacent roads $j \in \mathcal{W}_i$ with probability $\alpha_{i,j}^{(k)}$. It continues traversing the road network via these adjacency relationships following the routing probabilities $\alpha_{i,j}^{(k)}$ until it is adjacent to its goal $t^{(k)}$. At this point, the vehicle proceeds to the destination and changes its class to $k' \in \mathcal{O}_{t^{(k)}}$ with probability $p_{i,t^{(k)}}^{(k')}$. This behavior is encapsulated by the routing matrix

$$p_{i,k;j,k'} = \begin{cases} 
\alpha_{i,j}^{(k)} & \text{if } k = k', j \in \mathcal{W}_{\text{Child}(i)}, t^{(k)} \notin \mathcal{W}_{\text{Child}(i)}, \\
p_{j}^{(k')} & \text{if } j = i^{(k)}, t^{(k)} \in \mathcal{W}_{\text{Child}(i)}, k' \in \mathcal{O}_j, \\
0 & \text{otherwise},
\end{cases}$$

such that $\sum_{j,k'} p_{i,k;j,k'} = 1$. Thus, the relative throughput $\pi_{i,k}$, total relative throughput $\pi_i$, and utilization $\gamma_i$ have the same definition as in the Background Material.

As stated before, the queueing process at each station is modeled as a SS queue where the service rate of the vehicles $\mu_i(a)$ is equal to the sum of real and virtual passenger arrival rates, i.e.,

$$\mu_i(a) = \bar{\lambda}_i$$

for any station $i$ and queue length $a$. Additionally, by modeling road links as IS queues, we assume that their service rates follow a Cox distribution with mean $\mu_i(a) = c_i(a)/T_i$, where $T_i$ is the expected time required to traverse link $i$ in absence of congestion, and $c_i(a)$ is the capacity factor when there are $a$ vehicles in the queue. In this chapter, we only consider the case of load-independent travel times, therefore $c_i(a) = a$ for all $a$, i.e., the service rate is the same regardless of the number of vehicles on the road. We do not make further assumptions on the distribution of the service times. The assumption of load-independent travel times is representative of uncongested traffic [Bureau of Public Roads, 1964]; in the next section we discuss how to incorporate probabilistic constraints for congestion on road links.
2.3.3 Problem formulation

As stated in Equation (2.4), vehicles tend to accumulate in bottleneck stations driving their availability to 1 as the fleet size increases, while the rest of the stations have availability strictly smaller than 1. In other words, for unbalanced systems, availability at most stations is capped regardless of fleet size. Therefore, it is desirable to make all stations “bottleneck” stations, i.e., set the constraint \( \gamma_i = \gamma_j \) for all \( i, j \in S \), so as to (i) enforce a natural notion of “service fairness,” and (ii) prevent needless accumulation of empty vehicles at the stations.

However, it is desirable to minimize the impact that the rebalancing vehicles have on the road network. We achieve this by minimizing the expected number of vehicles on the road serving customer and rebalancing demands. Using Equation (2.3), the expected number of vehicles on a given road link \( i \) is given by \( \Lambda_i(m)T_i \).

Lastly, we wish to avoid congestion on the individual road links. Traditionally, the relation between vehicle flow and congestion is parametrized by two basic quantities: the free-flow travel time \( T_i \), i.e., the time it takes to traverse a link in absence of other traffic; and the nominal capacity \( C_i \), i.e., the measure of traffic flow beyond which travel time increases very rapidly [Patriksson, 2015]. Assuming that travel time remains approximately constant when traffic is below the nominal capacity (an assumption typical of many state-of-the-art traffic models [Patriksson, 2015]), our approach is to keep the expected traffic \( \Lambda_i(m)T_i \) below the nominal capacity \( C_i \) and thus avoid congestion effects. Note that by constraining in expectation there is a non-zero probability of exceeding the capacity; however, we will show that, asymptotically, it is also possible to constrain the probability of exceeding the congestion constraint.

Accordingly, the routing problem we wish to study in this chapter (henceforth referred to as the Optimal Stochastic Capacitated AMoD Routing and Rebalancing problem, or OSCARR) can now be formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in I} \Lambda_i(m)T_i, \\
\text{subject to} & \quad \gamma_i = \gamma_j, \quad i, j \in S, \quad \text{(2.7a)} \\
& \quad \Lambda_i(m)T_i \leq C_i, \quad i \in I, \quad \text{(2.7b)} \\
& \quad \pi_{j,k} = \sum_{k' \in K} \sum_{j \in N} \pi_{j,k}p_{j,k};t^{(k)},k', \quad k \in K, \quad \text{(2.7c)} \\
& \quad \pi_{i,k} = \sum_{k \in K} \sum_{j \in N} \pi_{j,k}p_{j,k};i,k, \quad i \in \{S \cup I\}, \quad \text{(2.7d)} \\
& \quad \sum_{j \in \text{WChild}(i)} \alpha_{ij}^{(k)} = 1, \quad \alpha_{ij}^{(k)} \geq 0, \quad i, j \in \{S \cup I\}, \quad \text{(2.7e)} \\
& \quad \lambda^{(r)} \geq 0, \quad r \in R. \quad \text{(2.7f)}
\end{align*}
\]
Constraint (2.7a) enforces equal availability at all stations, while constraint (2.7b) ensures that all road links are (on average) uncongested. Constraints (2.7c)–(2.7f) enforce consistency in the model. Specifically, (2.7c) ensures that all traffic leaving the source $s^{(k)}$ of class $k$ arrives at its destination $t^{(k)}$, (2.7d) enforces the traffic equations (2.1), (2.7e) ensures that $\alpha_{ij}^{(k)}$ is a valid probability measure, and (2.7f) guarantees nonnegative rebalancing rates.

### 2.3.4 Limitations

At this point, we would like to reiterate some assumptions and limitations built into the model. First, the proposed model is time-invariant. That is, we assume that customer and rebalancing rates remain constant for the segment of time under analysis, and that the network is able to reach its equilibrium distribution. An option for including the variation of customer demand over time is to discretize a period of time into smaller segments, each with its own arrival parameters and resulting rebalancing rates. These customer arrival rates, in turn, could be conditioned on external factors such as weather. Second, the passenger loss model assumes impatient customers and is well suited for cases where a high level of service is required. This allows us to simplify the model by focusing only on the vehicle process; however, it disregards the fact that customers may have different waiting thresholds and, consequently, the queueing process of waiting customers. Third, we focus on keeping traffic within the nominal road capacities in expectation, allowing us to assume load-independent travel times and to model exogenous traffic as a reduction in road capacity. Finally, we make no assumptions on the distribution of travel times on the road links: the analysis proposed in this chapter captures arbitrary distributions of travel times and only depends on the mean travel time.

### 2.4 Asymptotically Optimal Algorithms for AMoD Routing

In this section we show that, as the fleet size goes to infinity, the solution to OSCARR can be found by solving a linear program. This insight allows the efficient computation of asymptotically optimal routing and rebalancing policies and the characterization of the corresponding performance parameters.

First, we show that (i) the relative throughput at the stations can be expressed in terms of the relative throughputs at the other stations, and (ii) the balanced network constraint can be expressed in terms of the arrival rates. Then, we express the problem from a flow conservation perspective. Finally, we show that the problem allows an asymptotically optimal solution with bounds on the probability of exceeding road capacities. The solution we find is equivalent to the one presented in [Rossi et al., 2018]: thus, we show that the network flow model in [Rossi et al., 2018] also captures the asymptotic behavior of a stochastic AMoD routing and rebalancing problem.
2.4.1 Folding of traffic equations

The next two lemmas show that the traffic equations (2.1) at the SS queues can be expressed in terms of other SS queues, and that the balanced network constraint can be expressed in terms of real and virtual passenger arrivals.

Lemma 2.4.1 (Folding of traffic equations). Consider a feasible solution to OSCARR. Then, the total relative throughputs of the single server stations can be expressed in terms of the relative throughputs of the other single server stations, that is

\[ \pi_i = \sum_{k \in K} \tilde{p}_{s(k)} \pi_{s(k)}, \quad i \in S. \]  

(2.8)

Proof. Using the routing matrix specified in Equation (2.6) we can rewrite the class throughputs (2.1) as

\[ \pi_{i,k} = \sum_{k' \in K, j \in N} \pi_{j,k'} p_{j,k';i,k} = \sum_{k' \in D_i} \sum_{j \in N_{in}(j)} \pi_{j,k'} \tilde{p}_{s(k')} \pi_{s(k')}, \]  

(2.9)

The second equality exploits the fact that only queues feeding into \( i \) and vehicles whose class destination is \( i \) are routed to \( i \). The third and fourth equalities follow from the fact that the probability of switching into class \( k \) at queue \( i \) is the same regardless of the original class \( k' \). This allows us to rewrite the total relative throughput as

\[ \pi_i = \sum_{k \in K} \tilde{p}_{s(k)} \sum_{k' \in D_i} \sum_{j \in N_{in}(j)} \pi_{j,k'} = \sum_{k' \in D_i} \sum_{j \in N_{in}(j)} \pi_{j,k'}, \]  

(2.10)

since \( \sum_{k=1}^{K} \tilde{p}_{s(k)} = 1 \). As a consequence of (2.10) and (2.9), the class relative throughputs can be related to the total relative throughputs

\[ \pi_{i,k} = \tilde{p}_{s(k)} \pi_i. \]  

(2.11)

Now, assume the relative throughputs belong to a feasible solution to OSCARR. We proceed to reduce (2.7c) by using the routing matrix

\[ \pi_{s(k),k} = \sum_{k' \in K} \sum_{j \in N} \pi_{j,k'} p_{j,k';(k),k'} = \sum_{k' \in K} \tilde{p}_{s(k')} \sum_{j \in N_{in}(t(k))} \pi_{j,k'} = \sum_{j \in N_{in}(t(k))} \pi_{j,k'}. \]  

(2.12)
By inserting this into (2.10) and applying (2.11) we obtain
\[ \pi_i = \sum_{k \in D_i} \pi_s(k), \quad \pi_i = \sum_{k \in D_i} \tilde{p}_{s(k)} \pi_s(k). \] (2.13)

**Lemma 2.4.2** (Balanced system in terms of arrival rates). Consider a feasible solution to OS-CARR, then the constraint \( \gamma_i = \gamma_j \) for all \( i, j \) is equivalent to
\[ \tilde{\lambda}_i = \sum_{k \in D_i} \lambda^{(k)}. \] (2.14)

**Proof.** The proof of this lemma is very similar to Theorem 4.3 in [Zhang and Pavone, 2016]. Consider the case where (2.14) holds. We can write (2.8) in terms of the relative utilization rate:
\[ \left( \sum_{k \in D_i} \lambda^{(k)} \right) \gamma_i = \sum_{k \in D_i} \gamma_s(k) \lambda^{(k)}. \] (2.15)

Now, by grouping customer and rebalancing classes by origin-destination pairs, we define \( \varphi \) as
\[ \varphi_{ij} = \lambda^{(q)} + \lambda^{(r)}, \] (2.16)
such that \( s^{(q)} = s^{(r)} = j \) and \( t^{(q)} = t^{(r)} = i \). Additionally, let \( \zeta_{ij} = \varphi_{ij} / \sum_j \varphi_{ij} \). We note that there are no classes for which \( s^{(k)} = t^{(k)} \), so we set \( \varphi_{ii} = \zeta_{ii} = 0 \). Under this definition, the variables \{\( \zeta_{ij} \)\}_{ij} represent an irreducible Markov chain. Thus, Equation (2.15) can be rewritten as \( \gamma_i = \sum_j \gamma_j \zeta_{ij} \) or more compactly as \( Z \gamma = \gamma \), where the rows of \( Z \) are \([\zeta_{i1}, \zeta_{i2}, ..., \zeta_{iS}]\), with \( S = |S|, i = 1, ..., S \), and \( \gamma = (\gamma_1, ..., \gamma_s) \). Since \( Z \) is an irreducible, row stochastic Markov chain, by the Perron-Frobenius theorem the unique solution is given by \( \gamma = (1, ..., 1)^T \). Thus, \( \gamma_i = \gamma_j \) for all \( i \).

On the other hand, we consider again Equation (2.15). If the network \( Z \) is a solution to problem (2.7), then for all \( i, j \), we have \( \gamma_i = \gamma_j = \gamma \), and (2.15) becomes
\[ \gamma \tilde{\lambda}_i = \gamma \sum_{k \in D_i} \lambda^{(k)}, \quad \tilde{\lambda}_i = \sum_{k \in D_i} \lambda^{(k)}. \] (2.17)

### 2.4.2 Asymptotically optimal solution

As discussed in the background material, relative throughputs are computed up to a constant multiplicative factor. Thus, without loss of generality, we can set the additional constraint \( \pi_{s(1)} = \tilde{\lambda}_1 \),

---

**References:**

[Zhang and Pavone, 2016](#)
which, along with (2.17), implies that

\[ \pi_i = \bar{\lambda}_i, \quad \pi_{s(k),k} = \lambda^{(k)}, \quad \text{and} \quad \gamma_i = 1, \quad \text{for all} \quad i \in S. \]

(2.18)

As discussed earlier, the availabilities of stations with the highest relative utilization tend to one as the fleet size goes to infinity. Since the stations are modeled as SS queues, \( \rho_{i} = \gamma_{i} \) for all \( i \in S \).

Therefore, if the system is balanced, \( \gamma_{i} = \gamma_{S}^{\max} = \gamma = 1 \) for all \( i \in S \). That is, the set of bottleneck stations \( G \) includes all stations in \( S \) and \( \lim_{m \to \infty} G(m-1)/G(m) = 1 \) by Equation (2.4).

As \( m \to \infty \) and \( G(m-1)/G(m) \to 1 \), the throughput at every station \( \Lambda_i(m) \) becomes a linear function of the relative frequency of visits to that station, according to Equation (2.2). Thus, the objective function and the constraints in (2.7) are reduced to linear functions. We define the resulting problem (i.e., Problem (2.7) with \( G(m-1)/G(m) = 1 \)) as the Asymptotically Optimal Stochastic Capacitated AMoD Routing and Rebalancing problem, or A-OSCARR. The following lemma shows that the optimal solution to OSCARR approaches the optimal solution to A-OSCARR as \( m \) increases.

**Lemma 2.4.3** (Asymptotic behavior of OSCARR). Let \( \{\pi_{i,k}^{*}(m)\}_{i,k} \) be a set of relative throughputs corresponding to an optimal solution to OSCARR with a given set of customer demands \( \{\lambda^{(q)}\}_{q} \) and a fleet size \( m \). Also, let \( \{\hat{\pi}_{i,k}\}_{i,k} \) be a set of relative throughputs corresponding to an optimal solution to A-OSCARR for the same set of customer demands. Then,

\[
\lim_{m \to \infty} \frac{G(m-1)}{G(m)} \sum_{i \in I} T_i \sum_{k \in K} \pi_{i,k}^{*}(m) = \sum_{i \in I} T_i \sum_{k \in K} \hat{\pi}_{i,k}.
\]

(2.19)

**Proof.** We arrive to the proof by contradiction. Recall that \( \pi_i = \sum_{k \in K} \pi_{i,k} \). Assume Equation (2.19) did not hold. By definition,

\[
\frac{G(m-1)}{G(m)} \sum_{i \in I} T_i \pi_i^{*}(m) \leq \frac{G(m-1)}{G(m)} \sum_{i \in I} T_i \pi_i,
\]

(2.20)

and

\[
\sum_{i \in I} T_i \hat{\pi}_i(m) \leq \sum_{i \in I} T_i \pi_i,
\]

(2.21)

for all \( m \) and \( \{\pi_{i,k}\}_{i,k} \). Applying the limit to (2.20) and using (2.4), we obtain

\[
\sum_{i \in I} T_i \lim_{m \to \infty} \pi_i^{*}(m) \leq \sum_{i \in I} T_i \pi_i.
\]

However, according to our assumption,

either \( \sum_{i \in I} T_i \lim_{m \to \infty} \pi_i^{*}(m) > \sum_{i \in I} T_i \hat{\pi}_i \), or \( \sum_{i \in I} T_i \lim_{m \to \infty} \pi_i^{*}(m) < \sum_{i \in I} T_i \hat{\pi}_i \).
2.4. ASYMPTOTICALLY OPTIMAL ALGORITHMS FOR AMOD ROUTING

But the former violates Equation (2.20), and the latter Equation (2.21).

As discussed in the Problem Formulation, constraint (2.7b) only enforces an upper bound on the expected number of vehicles traversing a link. However, in the asymptotic regime, it is possible to enforce an analytical upper bound on the probability of exceeding the nominal capacity of any given road link. As seen in Equation (2.5), as the fleet size increases, the distribution of the number of vehicles on a road link \(i\) converges to a Poisson distribution with mean \(T_i \pi_i\). The cumulative density function of a Poisson distribution is given by 
\[
\Pr(X < \bar{x}) = Q(\lfloor \bar{x} + 1 \rfloor, \tilde{C})
\]
where \(\tilde{C}\) is the mean of the distribution and \(Q\) is the regularized upper incomplete gamma function. Let \(\epsilon\) be the maximum tolerable probability of exceeding the nominal capacity. Set \(\hat{C}_i = Q^{-1}(1 - \epsilon; [C_i + 1])\), i.e., 
\[
Q([C_i + 1], \hat{C}_i) = 1 - \epsilon.
\]
Then the constraint \(\Lambda_i(m) T_i \leq \hat{C}_i\) is equivalent to 
\[
\lim_{m \to \infty} P_i(x_i < C_i; m) \geq 1 - \epsilon.
\]

2.4.3 Linear programming formulation and multi-commodity flow equivalence

In the previous subsection, we show that A-OSCARR collapses into linear functions. In this subsection, we further show that A-OSCARR can be framed as an instance of the well-known multi-commodity flow problem and that A-OSCARR is equivalent to the Congestion-Free Routing and Rebalancing problem presented in [Rossi et al., 2018]: thus, (i) A-OSCARR can be solved efficiently by ad-hoc algorithms for multi-commodity flow (e.g. [Goldberg et al., 1998]) and (ii) the theoretical results presented in [Rossi et al., 2018] (namely, the finding that rebalancing trips do not increase congestion) extend, in expectation, to stochastic systems.

First, we show that the problem can be solved exclusively for the relative throughputs on the road links, and then we show that the resulting equations are equivalent to a minimum cost multi-commodity flow problem.

The relative throughput going from an intersection \(i\) into adjacent roads is 
\[
\sum_{j \in \mathcal{W}_i} \pi_{j,k},
\]
where \(\mathcal{W}_i = \{\mathcal{W}_i \cap \mathcal{I}\}\) is the set of road links that begin in node \(i\). Similarly, the relative throughput entering the intersection \(i\) from the road network is 
\[
\sum_{j \in \mathcal{U}_i} \pi_{j,k},
\]
where \(\mathcal{U}_i = \{\mathcal{U}_i \cap \mathcal{I}\}\) is the set of road links terminating in \(i\). Additionally, define \(d_i^{(k)}\) as the difference between the relative throughput leaving the intersection and the relative throughput entering the intersection. From (2.7d), (2.7c), and (2.18), it can be shown that, for a customer class \(q\) at an intersection \(i\), \(d_i^{(q)}\) should be equal to the arrival rate if \(i\) is adjacent to the source station, the negative arrival rate if it is adjacent to the
target station, and 0 otherwise. Formally, \[
\sum_{j \in W_i'} \pi_{j,q} - \sum_{j \in U_i'} \pi_{j,q} = d^{(q)}_i, \quad \text{where} \quad d^{(q)}_i = \begin{cases} 
\lambda^{(q)} & \text{if } i = s^{(q)}, \\
-\lambda^{(q)} & \text{if } i = t^{(q)}, \\
0 & \text{otherwise.} 
\end{cases}
\]

While the rebalancing arrival rates \(\lambda^{(r)}\) are not fixed, we do know from Equation (2.7c) and from the definition of \(d^{(q)}_i\) that \(d^{(r)}_{s^{(r)}} = -d^{(r)}_{t^{(r)}}\). Thus,
\[
\sum_{j \in W_{s^{(r)}}'} \pi_{j,r} - \sum_{j \in U_{s^{(r)}}'} \pi_{j,r} = - \sum_{j \in W_{t^{(r)}}'} \pi_{j,r} + \sum_{j \in U_{t^{(r)}}'} \pi_{j,r}.
\]

Finally, we can rewrite Lemma 2.4.2 as
\[
\sum_{q \in Q} d^{(q)}_i + \sum_{r \in R} \sum_{j \in W_i'} \pi_{j,r} - \sum_{j \in U_i'} \pi_{j,r} = 0.
\]

Thus, in the asymptotic regime, Problem (2.7) can be restated as

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in I} \sum_{k \in K} T_i \pi_{i,k}, \\
\text{subject to} & \quad \sum_{q \in Q} d^{(q)}_i + \sum_{r \in R} \sum_{j \in W_i'} \pi_{j,r} - \sum_{j \in U_i'} \pi_{j,r} = 0 & \forall i \in S, \quad (2.22a) \\
& \quad T_i \sum_{k \in K} \pi_{j,k} \leq \hat{C}_i & \forall i \in I, \quad (2.22b) \\
& \quad \sum_{j \in W_i'} \pi_{j,q} - \sum_{j \in U_i'} \pi_{j,q} = d^{(q)}_i & \forall i \in S, \quad (2.22c) \\
& \quad \sum_{j \in W_{s^{(r)}}'} \pi_{j,r} - \sum_{j \in U_{s^{(r)}}'} \pi_{j,r} = \sum_{j \in W_{t^{(r)}}'} \pi_{j,r} - \sum_{j \in U_{t^{(r)}}'} \pi_{j,r} & \forall r \in R, \quad (2.22d) \\
& \quad \sum_{j \in W_i'} \pi_{j,r} - \sum_{j \in U_i'} \pi_{j,r} = 0 & \forall i \in S \setminus \{s^{(r)}, t^{(r)}\}, \quad (2.22e) \\
& \quad \sum_{j \in W_{s^{(r)}}'} \pi_{j,r} - \sum_{j \in U_{s^{(r)}}'} \pi_{j,r} \geq 0 & \forall r \in R, \quad (2.22f) \\
& \quad \pi_{i,k} \geq 0, & \forall i \in I, k \in K. \quad (2.22g)
\end{align*}
\]

Here, constraints (2.22a) and (2.22b) are direct equivalents to (2.7a) and (2.7b), respectively. By keeping traffic continuity and equating throughputs at source and target stations, (2.22c) enforces (2.7c) and (2.7d) for the customer classes. For the rebalancing classes, (2.22d) is equivalent to (2.7c)
and \(2.22e\) to \(2.7d\). Non-negativity of rebalancing rates \(2.7f\) is kept by \(2.22f\).

Thus, A-OSCARR can be solved efficiently as a linear program. Note that this formulation is very similar to the multi-commodity flow formulation presented in [Rossi et al., 2018]. The formulation in this chapter prescribes specific routing policies for distinct rebalancing origin-destination pairs, while [Rossi et al., 2018] only computes a single “rebalancing flow.” These two formulations, however, are equivalent, as by using a flow decomposition algorithm [Ford and Fulkerson, 1962], one can “expand” the single rebalancing flow considered in [Rossi et al., 2018] into a set of rebalancing flows, one for each origin-destination pair. Therefore, it is possible to extend the theoretical results presented in [Rossi et al., 2018] to the stochastic setting. Most notably, it is possible to find rebalancing trips that in expectation do not cause more congestion than what would be caused from the same travel demand being satisfied by private vehicles.

### 2.5 Battery Charge Constraints

Plug-in electric vehicles (EVs) are especially suitable to AMoD systems. On the one hand, the type of short-range trips typical of Mobility-on-Demand (and, in the future, AMoD) systems is well-suited to the current generation of range-limited electric vehicles; on the other hand, intelligent policies for rebalancing and charging of EVs can ensure that vehicles with an adequate charge level are available to passengers, greatly reducing “range anxiety”—one of the main barriers to EV adoption [Evarts, 2013].

The BCMP framework (and in particular the notion of queueing classes) can be leveraged to model the battery charge level of electric vehicles and efficiently compute coupled rebalancing and charging policies. Accordingly, in this section, we extend the proposed BCMP model to include the constraints imposed by operating a fleet of charge-constrained autonomous EVs.

#### 2.5.1 Casting charge constraints as classes

We discretize the state-of-charge (SOC) of vehicles using a finite set of quantized charge levels. Specifically, we define \(B = \{1, 2, \ldots, B\}\) as an ordered set of \(B\) charge levels, such that \(B\) denotes a full battery and \(1\) denotes an empty battery. A vehicle entering road \(j \in \mathcal{I}\) at charge level \(b \in B\) spends \(e_j\) energy in its traversal; thus, the vehicle’s charge level at the end of the road is \(b - e_j\). We assume that vehicles spend no energy while idle at the stations, i.e., \(e_i = 0\) if \(i \in \mathcal{S}\). Note that this model assumes that charge depletion on a road segment is independent of speed/congestion: this approximation is reasonable in our model as (i) we force the transportation network to be (mostly) congestion free and (ii) if a road link is congestion free, it is reasonable to assume that each vehicle travels at the same free flow speed for that link. Vehicles can recharge their batteries at a set of plug-in chargers \(\mathcal{F}\): a vehicle can increase the level of charge of its batteries at charger \(f \in \mathcal{F}\) by up to \(e_f\) levels in time \(T_f\). Analogously to road links, chargers are modeled as IS queues with service
rate $1/T_f$.

To track the SOC of individual vehicles, we include the charge level as part of their class. Classes are henceforth denoted by the tuple $(k, b)$: a vehicle belonging to class $(k, b)$ is servicing request $k$ at charge level $b$. Analogously, the relative throughput of vehicles serving request $k$ at charge level $b$ on road link $i$ is $\pi_{i,k,b}$. We refer to the total relative throughput on a road link $i$ as

$$\pi_i = \sum_{k \in K} \sum_{b \in B} \pi_{i,k,b}.$$  

Routing and charging depend on the vehicles’ SOC: for example, energy-depleted vehicles need to charge their batteries before providing service to passengers. Accordingly, rebalancing requests are characterized by a set of tuples that include the initial and final SOC, namely

$$\mathcal{R} = \{s^{(r)}, t^{(r)}, \lambda^{(r)}, \text{soc}_s^{(r)}, \text{soc}_t^{(r)}\}$$

where $\text{soc}_s^{(r)}$ is the initial SOC and $\text{soc}_t^{(r)}$ is the final SOC.

For a given station $j$, the distribution of customer destinations is encoded by the distribution of the transition probabilities $\{p_j^{(k')}\}_{j,k'}$ in Equation (2.6), for any class $k' \in Q$ corresponding to passenger requests. In the closed queueing network model adopted in this chapter, customers are assigned to the first available vehicle in the FCFS queue, irrespective of its charge level. Therefore, for classes $k' \in Q$, the distribution $\{p_j^{(k')}\}_{j,k' \in Q}$ must be independent of the charge level $b$ of incoming vehicles: if this was not the case, the arrival rate of passengers in a given class would depend on the SOC of vehicles queueing at the station.

On the other hand, the system operator should enforce different rebalancing and charging strategies for non-passenger-carrying vehicles depending on the vehicles’ SOC. For example, it may be preferable to charge vehicles that are running out of battery, whereas vehicles with a high charge level may be devoted to rebalancing purposes. To enable this, the distribution $\{p_j^{(k')}\}_{j,k' \in \mathcal{R}}$ for the rebalancing classes $k' \in \mathcal{R}$ is modeled as dependent on the charge level. However, if we were to include the notion of SOC-dependent class assignment directly into the BCMP model from the previous section, we would introduce a spurious correlation between the passenger arrivals and the SOC: for example, in the case where vehicles with low charge were to be primarily assigned to rebalancing classes, a station having only low charge vehicles would behave as if there were no passenger arrivals. To address this issue, our strategy is to introduce the notion of stations with double queues.

Specifically, each station $i \in S$ is represented by two vehicle queues, one awaiting passenger requests, indexed with $i_Q$, and one awaiting rebalancing requests, indexed with $i_R$. Let $\beta_{i,b}$ be the probability that a vehicle arriving at station $i$ is assigned to the rebalancing queue $i_R$. Specifically, upon arrival to its destination station $i = t^{(k)}$, a vehicle of class $(k, b)$ proceeds to the rebalancing queue $i_R$ with probability $\beta_{i,b}$ and switches to class $(r, b)$ with probability $\tilde{p}_{iR,b}^{(r)} = \lambda^{(r)}/\sum_{r' \in O_{i,b,R}} \lambda^{(r')}$, where $O_{i,b,R}$ is the set of rebalancing requests $r'$ with $s^{(r')} = i$ and $\text{soc}_s^{(r')} = b$. Conversely, the
vehicle proceeds to the passenger queue $i_Q$ with probability $1 - \beta_{i,b}$ and switches to class $(q,b)$ with probability $\bar{p}_{iQ}^{(q)} = \lambda^{(q)}/\sum_{q' \in O_i} \lambda^{(q')}$. Figure 2.2 shows a graphical depiction of the station model with *double queues*.

Figure 2.2: Graphical depiction of a double queue. Vehicles directed to station $i$ at charge level $b$ enter the rebalancing queue $i_R$ with probability $\beta_{i,b}$; they enter the passenger-serving queue $i_Q$, with probability $1 - \beta_{i,b}$. Note that the probability of joining $i_R$ (or, equivalently, $i_Q$) depends on the SOC. Vehicles entering $i_R$ switch to a class $r$, corresponding to a rebalancing request, with probability $\lambda^{(r)}/\sum_{r' \in O_{i,b,R}} \lambda^{(r')}$, Conversely, vehicles entering $i_Q$ switch to class $q$, corresponding to a customer request, with probability $\lambda^{(q)}/\sum_{q' \in O_i} \lambda^{(q')}$. The routing policy for both customer-carrying and rebalancing vehicles is allowed to depend on the state of charge: for instance, vehicles with a low SOC may traverse a charging station to recharge their batteries. Accordingly, we let $\alpha^{(k,b)}$ denote the routing policy for a vehicle of class $(k,b)$ (as before, a routing policy is simply a matrix of transition probabilities).

Specifically, let $B_i^{(k)}$ be the set of *acceptable* SOC at the target station for class $k$, defined as $B_i^{(k)} = \{ \text{soc}^{(k)}_i \}$ if $k \in \mathcal{R}$, and $B_i^{(k)} = \mathcal{B}$ otherwise. We then define the routing matrix as

$$p_{i,k,b,j,k',b'} = \begin{cases} 
\alpha^{(k,b)}_{i,j} & \text{if } k = k', \ j \in \mathcal{W}_{\text{Child}(i)}, \\
\beta_{i,b} p_{j,R,b}^{(k')} & \text{if } j = t^{(k)}, \ i^{(k)} \in \mathcal{W}_{\text{Child}(i)}, \\
 & \text{if } j' \in \mathcal{O}_{j,b',R}, \ b' = b - e_i \in B_i^{(k)}, \\
(1 - \beta_{i,b}) p_{j,Q}^{(k')} & \text{if } j = t^{(k)}, \ i^{(k)} \in \mathcal{W}_{\text{Child}(i)}, \\
 & \text{if } k' \in \mathcal{O}_j, b' = b - e_i \in B_i^{(k)}, \\
0 & \text{otherwise.}
\end{cases} \tag{2.23}$$

### 2.5.2 Problem Formulation

The goal of the problem is to minimize the amount of traffic both on the roads and at the charging stations. Accordingly, we consider the cost function $\sum_{i \in \mathcal{I}'} \Lambda_i(m) T_i$, where $\mathcal{I}'$ represents the union of
road link and plug-in charger queues, i.e., \( I' := \{ I \cup F \} \). In analogy to the OSCARR problem, the system is constrained to remain in balance, i.e., \( \gamma_i = \gamma_j \) for all \( i, j \in S \) (note that in this case \( i \) and \( j \) might denote either a rebalancing or a passenger queue). We refer to the resulting problem as the \textit{Optimal Stochastic Capacitated AMoD Routing, Rebalancing and Charging problem} (OSCARR-C):

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in I'} A_i(m) T_i, \\
\text{subject to} & \quad \gamma_i = \gamma_j, \quad i, j \in S, \quad \text{(2.24a)} \\
& \quad A_i(m) T_i \leq C_i, \quad i \in I', \quad \text{(2.24b)} \\
& \quad \pi_{s(q),q} = \sum_{k' \in K, j \in N} \sum_{b \in B} \sum_{b' \in B} \pi_{j,q,b} p_{j,q,b,\ell(s,q)} \ell(k',b'), \quad q \in Q, \quad \text{(2.24c)} \\
& \quad \pi_{s(r),r,\text{soc}(r)} = \sum_{k' \in K, j \in N} \sum_{b \in B} \sum_{b' \in B} \pi_{j,q,b} p_{j,q,b,\ell(s,r),k',b'} \ell(r, b' = \text{soc}(r), \quad r \in R, \quad \text{(2.24d)} \\
& \quad \pi_{i,k,b} = \sum_{k' \in K, j \in N} \sum_{b' \in B} \sum_{k'' \in K} \pi_{j,k,b} p_{j,k,b',\ell_i,\ell(k',b')}, \quad i \in \mathcal{N}, \quad k \in K, \quad b \in B, \quad \text{(2.24e)} \\
& \quad \sum_{j \in \text{WChild}(i)} \alpha_{ij}^{(k,b)} = 1, \quad \alpha_{ij}^{(k,b)} \geq 0, \quad i, j \in \mathcal{N}, \quad k \in K, \quad b \in B, \quad \text{(2.24f)} \\
& \quad \lambda^r \geq 0, \quad r \in R. \quad \text{(2.24g)}
\end{align*}
\]

Constraints \( \text{(2.24c)} \) and \( \text{(2.24d)} \) ensure that passengers’ and rebalancing traffic leaving a source reaches the corresponding destination, and, in the case of rebalancing, a desired charge level. In analogy to \( \text{(2.7d)} \), \( \text{(2.24e)} \) enforces traffic continuity at the road and charge level. Constraint \( \text{(2.24f)} \) ensures that \( \alpha_{ij}^{(k,b)} \) is a valid probability measure, and \( \text{(2.24g)} \) limits rebalancing requests to positive values.

The setups of OSCARR-C and of OSCARR are very similar. Specifically, the modeling assumptions required to derive the asymptotically optimal formulation of OSCARR are also valid for OSCARR-C; namely, a closed, multi-class network with IS and SS queues constrained to maintain equal relative availability across the SS queues. Therefore, in analogy with \( \text{(2.18)} \), the desired relative utilizations can be set to \( \gamma_i = 1 \) for all \( i \in S \) without loss of generality. As a result, the SS relative throughputs are constrained to equal their service rates, that is,

\[
\pi_i = \tilde{\lambda}_i, \quad \text{for all } i \in S. \quad \text{(2.25)}
\]

Additionally, Lemma \( \text{2.4.3} \) is still valid. Indeed, the queues \( I' \) in the objective function are all IS queues. Furthermore, since for a feasible solution to OSCARR-C Equation \( \text{(2.25)} \) holds, it follows that \( \gamma_i = \gamma_{\text{max}}^S = 1 \) for all \( i \in S \), and, thus, all the SS queues in \( S \) are bottleneck queues. Therefore, from \( \text{(2.4)} \), \( \lim_{m \to \infty} \frac{G(m-1)}{G(m)} = 1. \) Thus, the two assumptions for Lemma \( \text{2.4.3} \) are verified. As a
2.5. BATTERY CHARGE CONSTRAINTS

consequence, according to Lemma 3, the optimal solution \( \{ \hat{\pi}_i \} \) to the asymptotic approximation of OSCARR-C is also optimal for the full OSCARR-C as the fleet size goes to infinity. We define the problem of finding an asymptotically optimal solution to OSCARR-C as the Asymptotically Optimal Stochastic Capacitated AMoD Routing, Rebalancing and Charging problem, or A-OSCARR-C. We next discuss how to solve A-OSCARR-C.

The key strategy is to rewrite constraints (2.24a), (2.24c), (2.24d), (2.24e), and (2.24f) as traffic conservation constraints, both at intersections and at stations. To this purpose, we first find a traffic conservation constraint in terms of both the relative throughputs and the transition probabilities. We then derive traffic conservation constraints only in terms of relative throughputs, both at intersections and at stations – these are the constraints that will be used to set up A-OSCARR-C (2.46).

In the following paragraphs, we provide and rigorously derive a tractable formulation of A-OSCARR-C based on the aforementioned strategy.

Traffic conservation in terms of relative throughputs and transition probabilities: Denote the relative throughput of vehicles arriving at station \( i \) with charge level \( b \) as \( \pi_{i,b} \). In other words, according to the routing matrix (2.23),

\[
\pi_{i,b} = \beta_{i,b} \sum_{k \in K} \pi_{i,k,b}.
\]

Similarly, denote the relative throughput of vehicles arriving at station \( i \) with charge level \( b \) as \( \pi_{i,b} \). In other words, \( \pi_{i,b} = (1 - \beta_{i,b}) \sum_{k \in K} \pi_{i,k,b} \).

For bookkeeping purposes, we define the combined relative throughput as \( \hat{\pi}_{i,b} := \pi_{i,R,b} + \pi_{i,Q,b} \). From (2.25), the relative throughput for the passenger queues must equal the rate of arrival, that is:

\[
\sum_{q \in O_i} \lambda(q) = \sum_{q \in Q} \sum_{b \in B} (1 - \beta_{i,b}) \hat{\pi}_{i,b} \tilde{p}(q)_{i}
\]

\[
= \sum_{q \in Q} \tilde{p}(q)^{(i)} \sum_{b \in B} (1 - \beta_{i,b}) \hat{\pi}_{i,b}
\]

\[
= \sum_{b \in B} (1 - \beta_{i,b}) \hat{\pi}_{i,b}
\]

\[
= \sum_{b \in B} \hat{\pi}_{i,b} - \pi_{i,R,b}.
\]

(2.26)

We now turn our attention to the traffic equations (2.24e). Note that, as per (2.23), the only queues that contribute relative throughput into a queue \( i \in I' \) are the set of queues \( U_{\text{parent}(i)} \) that feed into the parent intersection of \( i \). Thus, for queues \( i \in I' \)

\[
\pi_{i,k,b} = \sum_{k' \in K} \sum_{j \in N'} \sum_{b' \in B} \pi_{j,k',b'} p_{j,k',b';i,k,b}
\]

\[
= \sum_{k' \in K} \sum_{j \in U_{\text{parent}(i)}} \sum_{b' \in B} \pi_{j,k',b'} p_{j,k',b';i,k,b}.
\]

(2.27)

Moreover, \( p_{j,k',b';i,k,b} \neq 0 \) only for queues \( j \) which feed into the parent intersection of \( i \) and for charge
levels $b'$ such that $b' = b + e_j$ (note that $e_j = 0$, if $j \in S$). Therefore, for queues $i \in \mathcal{I}'$

$$\pi_{i,k,b} = \sum_{k' \in \mathcal{K}} \sum_{j \in \mathcal{U}_{\text{Parent}(i)}} \sum_{b'' \in \mathcal{B}} \pi_{j,k',b'} \mathcal{P}_{j,k',b''} i,k,b$$

$$= \sum_{j \in \mathcal{U}_{\text{Parent}(i)}} \pi_{j,k,b+e_j} \alpha_{j,i}^{(k,b+e_j)}.$$ \hspace{1cm} (2.28)

Let $\mathcal{W}_{\text{Parent}(i)}' = \{\mathcal{W}_{\text{Parent}(i)} \cap \mathcal{I}'\}$, $\mathcal{U}_{\text{Parent}(i)}' = \{\mathcal{U}_{\text{Parent}(i)} \cap \mathcal{I}\}$ and $l = \text{Parent}(i)$. For a road or charger queue $i \in \mathcal{I}'$ such that $l = \text{Parent}(i)$, we can then simplify (2.24e) and obtain the traffic conservation equation

$$\pi_{i,k,b} = \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,k,b+e_j} \alpha_{j,i}^{(k,b)}$$

$$= \begin{cases} \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,k,b+e_j} \alpha_{j,i}^{(k,b+e_j)}, & \text{if } s(k) \notin \mathcal{U}_{\text{Parent}(i)}', \\ \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,k,b+e_j} \alpha_{j,i}^{(k,b+e_j)} + \pi_{s(k),b} \alpha_{j,i}^{(b)}, & \text{if } s(k) \in \mathcal{U}_{\text{Parent}(i)}', k \in \mathcal{Q}, \\ \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,k,b+e_j} \alpha_{j,i}^{(k,b+e_j)} + \pi_{s(k),b} \alpha_{j,i}^{(k,b)}, & \text{if } s(k) \in \mathcal{U}_{\text{Parent}(i)}', b = \text{soc}_s^{(k)}, k \in \mathcal{R}, \end{cases}$$ \hspace{1cm} (2.29)

where the second equality exploits the fact that only the source station of class $k$ sends vehicles of class $k$ into the road network.

**Traffic conservation at intersections in terms of relative throughputs only:** We are now in a position to derive a number of traffic conservation constraints at intersections, in terms of only relative throughputs.

**Case 1:** if an intersection $l$ is not adjacent to either the source or target of a passenger class $q \in \mathcal{Q}$ (that is, $l \notin \{\text{Child}(s(q)), \text{Parent}(t(q))\}$ for $q \in \mathcal{Q}$), then $\sum_{j' \in \mathcal{W}_{l}'} \alpha_{j'y}^{(q,b)} = 1$. Thus, we can sum (2.29) over all queues to obtain the following traffic conservation equations for class $q \in \mathcal{Q}$ in terms of the relative throughput:

$$\sum_{j \in \mathcal{W}_{l}'} \pi_{j,q,b} = \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,q,b+e_j}, \quad \text{for } l \notin \{\text{Child}(s(q)), \text{Parent}(t(q))\}, \quad q \in \mathcal{Q}.$$ \hspace{1cm} (2.30)

**Case 2:** for a rebalancing class $r \in \mathcal{R}$, if neither $\{l = \text{Child}(s(r)) \land b = \text{soc}_s^{(r)}\}$ nor $\{l = \text{Parent}(t(r)) \land b = \text{soc}_t^{(r)}\}$, then $\sum_{j' \in \mathcal{W}_{l}'} \alpha_{j'y}^{(r,b)} = 1$. Thus, we can obtain the following traffic conservation equations for class $r \in \mathcal{R}$ in terms of the relative throughput

$$\sum_{j \in \mathcal{W}_{l}'} \pi_{j,r,b} = \sum_{j \in \mathcal{U}_{\text{Parent}(i)}'} \pi_{j,r,b+e_j} \quad \text{if } \neg\{l = \text{Child}(s(r)) \land b = \text{soc}_s^{(r)}\}$$

$$\text{and } \neg\{l = \text{Parent}(t(r)) \land b = \text{soc}_t^{(r)}\}, \quad r \in \mathcal{R}.$$ \hspace{1cm} (2.31)
2.5. BATTERY CHARGE CONSTRAINTS

Case 3: If \( l \) is adjacent to the source station, then summing (2.24) over all queues, one obtains

\[
\sum_{j \in W_t^i} \pi_{j,k,b} = \sum_{j \in U_t^i} \pi_{j,k,b+e_j} + \pi_{i,k,b}, \quad i = s^{(k)}, \quad l = \text{Child}(s^{(k)}). \tag{2.32}
\]

If \( k \) is a passenger class \( q \in Q \), then \( \pi_{i,k,b} = (1 - \beta_{i,k}) \tilde{\pi}_{i,b} \tilde{p}_t^q \), and one can derive the traffic conservation constraint

\[
\sum_{j \in W_t^i} \pi_{j,q,b} - \sum_{j \in U_t^i} \pi_{j,q,b+e_j} = (1 - \beta_{i,b}) \tilde{\pi}_{i,b} \tilde{p}_t^q,
\]

\[
= (\tilde{\pi}_{i,b} - \pi_{i,b}) \tilde{p}_t^q,
\]

\[
= (\tilde{\pi}_{i,b} - \pi_{i,b}) \sum_{\xi \in \mathcal{O}_t} \lambda^{q},
\]

\[
= \sum_{\xi \in B} (\tilde{\pi}_{i,b} - \pi_{i,b}) \lambda^{q},
\]

\[
= \tilde{\rho}_{iQ,b} \lambda^{q}, \quad \text{if } i = s^{(q)}, q \in Q, \quad l = \text{Child}(s^{(q)}),
\]

where the fourth equality follows from (2.26). In the fifth equality, we denote \( \tilde{\rho}_{iQ,b} \) as the ratio of the relative throughput that goes through queue \( iQ \) at charge level \( b \), that is \( \tilde{\rho}_{iQ,b} = \pi_{iQ,b}/\pi_{iQ} \). We treat this ratio as a decision variable: intuitively, \( \tilde{\rho}_{iQ,b} \) controls the charge level distribution of the vehicles available for passenger use.

If, instead, \( k = r \in R \), \( l \) is adjacent to the source station, and \( b \) is the desired target charge level, then \( \pi_{i,r,b} = \lambda^{(r)} \). Therefore one obtains the traffic conservation constraint

\[
\sum_{j \in W_t^i} \pi_{j,k,b} = \sum_{j \in U_t^i} \pi_{j,k,b+e_j} + \lambda^{(r)}, \quad b = \text{soc}_{q}^{(r)}, \quad l = \text{Child}(s^{(r)}), \quad r \in R.
\tag{2.34}
\]

Case 4: If the intersection \( l \) is adjacent to the target station \( t^{(k)} \) of a class \( k \) (either in \( Q \) or \( R \)), then \( \sum_{j \in W_t^i} \alpha_{j,j'}^{(k,b)} \neq 1 \) in general. Let \( \zeta_{t^{(k)},k,b} \) be the relative throughput of class \( k \) that enters station \( t^{(k)} \) from adjacent queues at charge level \( b \). Then, the sum of (2.29) must satisfy

\[
\sum_{j \in W_t^i} \pi_{j,q,b} + \zeta_{t^{(k)},k,b} = \sum_{j \in U_t^i} \pi_{j,q,b+e_j}, \quad \text{if } l = \text{Parent}(t^{(k)}).
\tag{2.35}
\]

However, we know from (2.24c) that the total relative throughput of a passenger class \( q \in Q \) entering its target station \( t^{(q)} \) must equal the total relative throughput at its source station, i.e., \( \pi_{s^{(q)}},q = \sum_{b \in B} \zeta_{t^{(q)},k,b} \). Additionally, if (2.25) holds, it can be shown that \( \pi_{s^{(q)}},q = \lambda^{(q)} \). Thus, summing (2.35) over all charge levels for passenger class \( q \in Q \), we obtain the traffic conservation constraint

\[
\sum_{b \in B} \sum_{j \in U_t^i} \pi_{j,q,b+e_j} - \sum_{j \in W_t^i} \pi_{j,q,b} = \lambda^{(q)}, \quad \text{if } l = \text{Parent}(t^{(q)}), \quad q \in Q.
\tag{2.36}
\]
Alternatively, equation (2.24d) enforces that the total relative throughput of a rebalancing class \( r \in R \) entering its target station \( t^{(r)} \) with charge level \( soc_t^{(r)} \) be equal to the relative throughput leaving its source \( s^{(r)} \) with charge level \( soc_s^{(r)} \). Thus, \( \pi_{s^{(r)},r,soc_s^{(r)}} = \zeta_{t^{(r)},r,b} = \lambda^{(r)} \), and the sum of (2.29) must satisfy

\[
\sum_{U_l'} \pi_{j,r,b+e_j} - \sum_{j \in W_l'} \pi_{j,r,b} = \lambda^{(r)}, \quad \text{if } l = \text{Parent}(t^{(r)}), \ b = soc_t^{(r)}, \ r \in R. \tag{2.37}
\]

Note that both the arrival rates and the relative throughputs of the rebalancing classes are decision variables. The rebalancing rates are not explicitly represented in the optimization problem: rather, they are implicitly set by equating (2.34) and (2.37):

\[
\sum_{W_l'} \pi_{j,r,soc_s^{(r)}} - \sum_{j \in U_l'} \pi_{j,r,soc_s^{(r)}+e_j} = \\
\sum_{j \in U_l'} \pi_{j,r,soc_t^{(r)}+e_j} - \sum_{W_l'} \pi_{j,r,soc_t^{(r)}}, \tag{2.38}
\]

and by setting a positive constraint on the relative throughputs at the source station, i.e.

\[
\sum_{W_l'} \pi_{j,r,soc_s^{(r)}} - \sum_{j \in U_l'} \pi_{j,r,soc_s^{(r)}+e_j} > 0, \ r \in R. \tag{2.39}
\]

Traffic conservation at stations in terms of relative throughputs only: Finally, we characterize a traffic conservation constraint for the stations (namely, Equation (2.45)), by manipulating the traffic equations (2.24e). Specifically, recall that, for a given station \( i \in S \), \( \hat{\pi}_{i,b} = \pi_{i_R,b} + \pi_{i_Q,b} \). Due to Equation (2.25), \( \pi_{i_R,b} = \sum_{r \in \mathcal{D}_{i,b,R}} \lambda^{(r)} \) and \( \pi_{i_Q,b} = \sum_{q \in \mathcal{O}_i} \theta_{i_Q,b} \lambda^{(q)} \). Therefore,

\[
\hat{\pi}_{i,b} = \sum_{r \in \mathcal{D}_{i,b,R}} \lambda^{(r)} + \sum_{q \in \mathcal{O}_i} \theta_{i_Q,b} \lambda^{(q)} \tag{2.40}
\]

Conversely, \( \hat{\pi}_{i,b} \) must equal the sum of passenger and rebalancing traffic that enters station \( i \in S \) at charge level \( b \). In particular, the rebalancing traffic entering station \( i \) at charge level \( b \) is \( \sum_{r \in \mathcal{D}_{i,b,R}} \lambda^{(r)} \), where \( \mathcal{D}_{i,b,R} \) is the set of rebalancing classes whose destination is \( i \in S \) with target charge level \( b \). From (2.36), we see that for a single charge level the relative throughput of a passenger class \( q \in \mathcal{Q} \) entering \( i \in S \) is \( \sum_{U_l'} \pi_{j,q,b+e_j} - \sum_{j \in W_l'} \pi_{j,q,b} \), where \( l = \text{Parent}(i) \). Summing this over all passenger classes and adding the rebalancing traffic we obtain

\[
\hat{\pi}_{i,b} = \sum_{r \in \mathcal{D}_{i,b,R}} \lambda^{(r)} + \sum_{q \in \mathcal{D}_{i,b,R}} \sum_{U_l'} \pi_{j,q,b+e_j} - \sum_{j \in W_l'} \pi_{j,q,b}. \tag{2.41}
\]
Together, (2.40) and (2.41) imply that
\[
\sum_{r \in \mathcal{O}_{i,b,R}} \lambda(r) + \sum_{q \in \mathcal{O}_{i}} q_{i,Q} \varrho_{i,Q,b} \lambda(q) = \sum_{r \in \mathcal{D}_{i,b,R}} \lambda(r) + \sum_{q \in \mathcal{D}_{i,Q}} \sum_{U'_l} \pi_{j,q,b} - \sum_{q \in \mathcal{D}_{i,Q}} \sum_{U'_l} \pi_{j,q,b}.
\] (2.42)

Note that \( \sum_{U'_l} \pi_{j,q,b} - \sum_{j \in U'_l} \pi_{j,r,b} + \epsilon_j \) equals \( \lambda(r) \) at its source station, \( -\lambda(r) \) at the target station, and 0 otherwise. Therefore, we can express \( \lambda(r) \) in terms of the relative throughputs
\[
\sum_{W'_l} \pi_{j,r,b} - \sum_{j \in U'_l} \pi_{j,r,b} + \epsilon_j = \begin{cases} 
\lambda(r), & \text{if } l = \text{Child}(s(r)), \\
-\lambda(r), & \text{if } l = \text{Parent}(t(r)), \\
0, & \text{otherwise}.
\end{cases}
\] (2.43)

The difference between incoming and departing rebalancing relative throughput at a station now becomes
\[
\sum_{r \in \mathcal{R}} \sum_{W'_l} \pi_{j,r,b} - \sum_{j \in U'_l} \pi_{j,r,b} + \epsilon_j = \sum_{r \in \mathcal{O}_{i,b,R}} \lambda(r) - \sum_{r \in \mathcal{D}_{i,b,R}} \lambda(r).
\] (2.44)

Thus, by rewriting (2.42), we obtain the traffic conservation constraint at each station \( i \in S \)
\[
\sum_{q \in \mathcal{O}_{i,Q}} q_{i,Q} \lambda(q) + \sum_{r \in \mathcal{R}} \sum_{W'_l} \pi_{j,r,b} - \sum_{j \in U'_l} \pi_{j,r,b} + \epsilon_j - \sum_{q \in \mathcal{D}_{i,Q}} \sum_{U'_l} \pi_{j,q,b} - \sum_{j \in W'_l} \pi_{j,q,b} = 0, \quad l = \text{Parent}(i) = \text{Child}(i).
\] (2.45)

Collecting all the results above, A-OSCARR-C can now be framed in terms of the relative throughputs \( \{ \pi_{i,k,b} \}_{i,k,b} \) and the ratios \( \{ q_{i,Q} \}_{i,b} \):

\[
\text{minimize} \quad \sum_{i \in \mathcal{I}'} \sum_{k \in \mathcal{K}} \sum_{b \in \mathcal{B}} \pi_{i,k,b},
\]
subject to
\[
\sum_{i \in \mathcal{I}'} \sum_{k \in \mathcal{K}} \sum_{b \in \mathcal{B}} \pi_{i,k,b} \leq \hat{C}_i, \quad \forall i \in \mathcal{I}, \quad (2.30), \quad 2.31, \quad 2.33, \quad 2.36, \quad 2.38, \quad 2.39, \quad 2.45 \quad (2.46a)
\]
\[
\pi_{i,k,b} \geq 0, \quad \forall i \in \mathcal{I}, k \in \mathcal{K}, b \in \mathcal{B} \quad (2.46b)
\]
\[
\sum_{b \in \mathcal{B}} q_{i,Q} = 1, \quad \forall i \in \mathcal{S}, \quad (2.46c)
\]
\[
q_{i,Q,b} \geq 0 \quad i \in \mathcal{S}, b \in \mathcal{B}. \quad (2.46d)
\]

Constraints (2.30)-(2.39) enforce consistency in the model. Constraint (2.45) enforces conservation of traffic at each charging level and, consequently, equal availabilities at each station. (2.46a)
sets the bounds on the expected traffic at the road and charger queues, and (2.46b) enforces non-negative traffic values. Finally, constraints (2.46c) and (2.46d) make sure that the ratios \( \{ \varrho_{i,b} \} \) are a valid probability measure.

As in the non-battery case, A-OSCARR-C can be solved as a linear program. The state size is \( O(|\mathcal{I}'||B|(|\mathcal{R}| + |\mathcal{Q}|)) \). \(|\mathcal{Q}| \) grows quadratically with \(|\mathcal{S}|\), and \(|\mathcal{R}| \) grows quadratically with \(|\mathcal{S}||B|\); thus, for large road networks, the problem size can become unwieldy even for modern linear programming algorithms. For instance, for a road network with 350 nodes, 1000 road segments and 60 charge levels, the problem size is slightly above 9 billion variables; for comparison, state-of-the-art LP solvers can reliably handle problems with tens of millions of variables on modern machines [Mittelmann, 2016]. Remarkably, it is possible to reduce the problem size, with no loss of information, by addressing A-OSCARR-C as an augmented network flow problem, bundling customer traffic demands according to their source node, and collecting all rebalancing demands into a single class. Under this approach, the same problem instance would be reduced to just over 3 million variables. For a given set of optimal customer and rebalancing flows, individual routes can be recovered using a flow decomposition algorithm [Ford and Fulkerson, 1962], in analogy with A-OSCARR. We refer the reader to [Rossi et al., 2018] for a thorough discussion.

### 2.6 Numerical Experiments

To illustrate a real-life application of the models and methods presented in this chapter, we performed a case study of Manhattan, where system performance metrics were computed as a function of fleet size using Mean Value Analysis [Gelenbe and Pujolle, 1998]. The road network model used for this case study consists of a subset of Manhattan’s real road network (shown in Figure 2.3), with 1,005 road links and 357 intersections. To select station positions and compute the rates \( \lambda(q) \) (for each tuple \( q \in \mathcal{Q} \) modeling the arrival process) we used the taxi trips within Manhattan that took place between 7:00AM and 8:00AM on March 1, 2012 (22,416 trips) from the New York City Taxi and Limousine Commission dataset\(^2\). We clustered the pickup locations into 50 different groups with \( K \)-means clustering and placed a station at the road intersection closest to each cluster centroid.

We then fit an origin-destination model with exponential distributions to describe the customer trip demands between the stations. In order to observe congestion effects, road capacities were reduced (specifically, by 55%) to ensure that maximum road utilization is achieved on some of the road links; in the real world, an analogous reduction in road capacity would be caused by traffic exogenous to the taxi system.

2.6. NUMERICAL EXPERIMENTS

Figure 2.3: Manhattan scenario. Left: modeled road network. Center: Station locations. Right: Resulting vehicular flow (darker flows show higher vehicular presence).

2.6.1 Routing and rebalancing under congestion constraints

We considered two scenarios: (i) the “baseline” scenario where traffic constraints on each road link are based on expectation, i.e., the average number of vehicles on a road link is below its nominal capacity; and (ii) the “conservative” scenario where the constraints are based on the asymptotic probability of exceeding the nominal capacity (specifically, the asymptotic probability of exceeding the nominal capacity is constrained to be lower than 10%). Figure 2.3 shows the road network, the station locations, and the resulting traffic flow, and Figure 2.4 shows the results.

We see from Figure 2.4a that, as intended, the station availabilities are balanced and approach one as the fleet size increases. However, Figure 2.4b shows that there is a trade off between availability and vehicle utilization. For example, for a fleet size of 4,000 vehicles, on average, half of the vehicles are waiting at the stations. In contrast, a fleet of 2,400 vehicles results in availability of 91% and only 516 vehicles (in expectations) wait at the stations. Not shown in the figures, 34% of the trips are for rebalancing purposes; in contrast, only about 18% of the traveling vehicles are rebalancing. This shows that rebalancing trips are significantly shorter than passenger trips, which is in line with the goal of minimizing the number of empty vehicles on the road and thus road congestion.

Figures 2.4a and 2.4b show only the results for the baseline case; for the conservative scenario, the difference in availabilities is less than 0.1%, and the difference in the total expected number of...
vehicles on the road is less than 7, regardless of fleet size. However, road utilization is significantly different in the two scenarios we considered. In Figure 2.4c we see that, as the fleet size increases, the likelihood of exceeding the nominal capacity approaches 50%. In contrast, in the conservative scenario, the probability of exceeding the capacity is never more than 10%—by design—regardless of fleet size.

Finally, we verified the validity of the load-independent travel time assumption. Assuming asymptotic conditions (in which case the number of vehicles on each road follows a Poisson distribution), we computed for both scenarios the expected travel time between each origin-destination pair by using the Bureau of Public Roads (BPR) delay model [Bureau of Public Roads, 1964], and estimated the difference with respect to the load-independent travel time used in this chapter. The BPR delay model is a commonly used equation for relating traffic to travel time [Bureau of Public Roads, 1964]. Under this model, the travel time on a road link is given by

$$T'_i = T_i \left(1 + \delta \left(\frac{x_i}{C_i}\right)^\beta\right),$$  \hspace{1cm} (2.47)

where $T'_i$ is the real mean travel time, $T_i$ is the free flow travel time, $x_i$ is the number of vehicles on the road, $C_i$ is the nominal capacity of the road, and $\delta$ and $\beta$ are parameters usually set to 0.15 and 3, respectively. The results, depicted in Figure 2.4d, show that the maximum difference for the baseline and conservative scenarios are an increase of around 8% and 4%, respectively, and the difference tends to be smaller for higher trip times. Thus, for this specific case study, our assumption is reasonable.

2.6.2 Inclusion of charging constrains

To study the behavior of the Manhattan scenario under charging constraints, we made some additional assumptions. We assumed a vehicle battery size of 8 kWh and a full charge range of 50km, specifications that are similar to Toyota’s iRoad urban mobility vehicle.\footnote{http://www.toyota-global.com/innovation/personal_mobility/i-road/} We assumed that energy consumption depends exclusively on distance and computed the road energy costs, $e_j$, using the road lengths and battery range. Additionally, we assumed that every station is equipped with chargers capable of delivering 75kW of power per vehicle, comparable to the superchargers offered by Tesla\footnote{https://www.tesla.com/supercharger}, and did not enforce a limit on the number of vehicles that can charge simultaneously. Battery capacity was discretized into 60 discrete charge levels, a number that showed a good trade-off between accuracy in the energy cost at the roads and the problem size. Finally, in order to discourage recharging of customer-carrying vehicles while penalizing needless rebalancing of customer-empty
vehicles, we imposed a higher cost per unit of time for passenger-carrying trips. Currently, the average hourly salary in the United States is $26.19\textsuperscript{5} and the hourly cost of a rental car with the ZipCar car-sharing service starts at $7\textsuperscript{6}. Assuming that the cost per unit time of a vehicle is $7/hr and the value of time of a passenger is $26.19/hr, then the overall value of time of a passenger-carrying vehicle is approximately $33/hr, or five times higher than the cost of an empty, or rebalancing, vehicle. Therefore, we weigh passenger traffic as being five times costlier than rebalancing traffic.

Figures 2.5a and 2.5b show our results. As expected, the algorithm does not route any of the passenger classes through the chargers, that is, a customer should not expect to spend time charging or worry about the range of the vehicle. Conversely, vehicles entering a station at lower charge levels are overwhelmingly devoted to rebalancing tasks. For example, in Figure 2.5, we see the throughput distribution, as the fleet size approaches infinity, for the station with the highest number of rebalancing requests (2.5a) and for the station with the highest passenger arrival rate (2.5b). As intended, all the charging is done by rebalancing vehicles. This explains the rebalancing spike at lower charge levels: the optimizer assigns vehicles with lower energy levels exclusively to serve rebalancing tasks. In contrast, the algorithm utilizes vehicles at the highest charge levels exclusively for passenger requests. By satisfying all charging requirements with empty vehicles, the

\textsuperscript{5}https://www.bls.gov/news.release/empsit.t19.htm
\textsuperscript{6}http://www.zipcar.com/check-rates/sf
proposed approach is able to successfully mitigate the risk of range anxiety.

![Figure 2.5: Passenger and rebalancing request throughput distribution for (a) the station with the highest rebalancing rate, and (b) the station with highest number of passenger requests.](image)

The impact of the charging constraints on availability is notable. In order to maintain 91% availability, the fleet should comprise 3,300 vehicles, a 37.5% increase over the scenario without charging. A fleet this large has an average of 384 vehicles charging at any given time. The fleet may also have a significant effect on the electric power system. The range of charger utilization varies significantly by location. The power draw of the most used charger station is between 1.1 and 2.2 MW 90% of the time, while the least utilized charger is between 0 and 300 kW. While the smaller charging stations could be well served by current charging station standards, like Tesla Supercharger stations which offer 145kW per charger for two cars, the larger stations will likely require greater coordination with the local power authorities. In total, the expected power consumption in the system is around 30MW.

### 2.6.3 Discussion

The two previous experiments showcase the modeling power of the proposed framework. In particular, the framework enables future practitioners to couple key modeling features, such as congestion and charging, to stochastic performance metrics such as availability, and, thus, to synthesize control policies. However, it is important to highlight some limitations. First, the problem formulations OSCARR and OSCARR-C are not always feasible. Most notably, travel demand might exceed road capacity. In such cases, some available options are to evaluate whether to increase the threshold in some of the road links (and reduce the freeflow speed accordingly), or whether to relax the problem by including slack variables that penalize capacity violation. Second, the numerical experiments presented rely on steady state analysis of the AMoD system with a fixed freeflow in the road links. A further study should evaluate its merits by comparing against microscopic simulations such as MATsim [Balmer et al., 2009].
2.7 Conclusions

In this chapter we presented a novel queueing theoretical framework for modeling AMoD systems. We showed that, for the routing and rebalancing problem, the stochastic model we propose asymptotically recovers existing models based on network flow approximations. The model enables the analysis and control of the probabilistic distribution of the vehicles, as opposed to just expected values. In particular, this model allows one to set arbitrary bounds on the asymptotic probability of exceeding the capacity of individual road links. The model is very expressive and can capture both congestion and the charge level of electric vehicles servicing the customers. As such, it can be used to synthesize routing, rebalancing and charging control policies for AMoD fleets with electric vehicles and stochastic demand.

The flexibility of the model presented will be further exploited in future work. First, we would like to incorporate a more accurate congestion model, using load-dependent IS queues as roads, in order to study heavily congested scenarios. Second, we currently consider the system in isolation from other transportation modes, whereas, in reality, customer demand depends on the perceived quality of the different transportation alternatives. Future research will explore the effect of AMoD systems on customer behavior and how to optimally integrate fleets of self-driving vehicles with existing public transit. Third, we would like to further explore the couplings that might arise between the charging policies of an electric-powered AMoD fleet and the electric grid. Of particular interest is the potential participation of an electric-powered AMoD system in the ancillary services market of the power grid. Fourth, the current model assumes that each customer travels alone: future research will address the problem of ride-sharing, where multiple customers may share the same vehicle. Lastly, the control policy proposed in this chapter is open-loop and thus sensitive to modeling errors (e.g., incorrect estimation of customer demand). Future research will characterize the stability, persistent feasibility and performance of real-time, closed-loop model predictive control schemes based on a receding-horizon implementation of the routing policies presented in this chapter.
Part II

Real-time Control
Chapter 3

Data-Driven Model Predictive Control of Autonomous Mobility-on-Demand Systems

In this chapter, we move away from steady-state models that provide structural insights into the operations of AMoD systems and, instead, delve into the real-time challenges of operating such systems. Specifically, we present an end-to-end, data-driven framework to control AMoD systems. We first model the AMoD system using a time-expanded network, and present a formulation that computes the optimal rebalancing strategy (i.e., preemptive repositioning) and the minimum feasible fleet size for a given travel demand. Then, we adapt this formulation to devise a Model Predictive Control (MPC) algorithm that leverages short-term demand forecasts based on historical data to compute rebalancing strategies. Using simulations based on real customer data from DiDi Chuxing, we test the end-to-end performance of this controller with a state-of-the-art LSTM neural network to predict customer demand: we show that this approach scales very well for large systems (indeed, the computational complexity of the MPC algorithm does not depend on the number of customers and of vehicles in the system) and outperforms state-of-the-art rebalancing strategies by reducing the mean customer wait time by up to to 91.3%.

3.1 Introduction

A key operational challenge for AMoD systems, as for any transportation system with asymmetric demand, is the problem of *imbalance*: vehicles naturally concentrate in a subset of the areas serviced by the MoD system, limiting the availability in other regions [Fricker and Gast, 2012; David, 2012]. Devising efficient operating strategies for the imbalance problem is an active area of research for MoD
and AMoD systems. However, the majority of the existing body of work does not leverage the ability to forecast customer demand. Accordingly, most existing control strategies are reactive: thus, they do not deal well with rapidly time-varying demand due to, e.g., commuting cycles, events, or weather phenomena. The goal of this chapter is to present an end-to-end, data-driven framework to control AMoD systems with a focus on the imbalance problem: by leveraging information about predicted customer demand in the control synthesis problem, we design a predictive control strategy that anticipates imbalances in customer demand and rebalance vehicles accordingly, and we demonstrate the performance of such strategy with real-world data.

**Literature Review.** A considerable amount of research has been devoted to the design and analysis of optimal control of taxi-like fleets. In non-autonomous MoD systems such as Uber, and Lyft, ridesharing operators often use pricing incentives, commonly known as dynamic pricing, to nudge drivers towards areas where demand outstrips supply. In [Banerjee et al., 2016] the authors provide a framework for synthesizing pricing policies; [Banerjee et al., 2015] shows that dynamic pricing is not necessarily better than the optimal static policy, but is more robust with respect to system parameter uncertainties.

However, in AMoD systems, the fleet operator can directly control the routes and schedules of the autonomous vehicles. [Pavone et al., 2012] [Zhang and Pavone, 2016] [Zhang et al., 2016b] [Volkov et al., 2012] approach the problem of controlling a fleet by first formulating steady-state solutions using queuing theoretical [Zhang and Pavone, 2016], fluidic [Pavone et al., 2012], network flow [Zhang et al., 2016b], or Markov [Volkov et al., 2012] models, respectively, and then deriving heuristic control laws informed by the steady-state solution that can be applied in real-time. However, by relying on steady-state formulations, the aforementioned pricing or control heuristics are time-invariant, and, in particular, cannot accommodate time-varying forecasted demand. Time-varying, MPC controllers have been proposed in [Zhang et al., 2016a] [Miao et al., 2016], and the methods in [Miao et al., 2016] [Miller and How, 2017] explicitly consider forecasted demand. In particular, the Model Predictive Control (MPC) approach from [Zhang et al., 2016a] experimentally outperforms the time-invariant heuristics listed above, and could potentially leverage forecasted demand. However, by assigning decision variables to each vehicle in [Zhang et al., 2016a] [Miao et al., 2016] and enumerating all positioning possibilities in [Miller and How, 2017], the problem sizes in these methods grow significantly for large fleets, which limits their real-life applications to small or medium fleets. Moreover, the model in [Miao et al., 2016] requires that vehicles should be able to pickup and dropoff customers within a time step, limiting its applicability to systems where all customer travel times are similar.

**Statement of Contributions.** Our contribution in this chapter is threefold. First, we propose an efficient approach to find the optimal dispatching policy for the case when the trip demand is known ahead of time. This provides an upper bound on the performance of the system. The approach is able to simultaneously optimize the dispatching policy and the number of required vehicles: thus,
it can be used for fleet sizing. Second, we propose an MPC algorithm for operating the system in real-time by leveraging short-term forecasts of customer demand. The complexity of the algorithm does not depend on the number of vehicles or on the number of customers in the transportation system: thus, the algorithm can be used to effectively control large-scale AMoD systems. Third, we validate these approaches using a dataset of DiDi Chuxing, the major ridesharing company in China: our results show that the proposed MPC algorithm outperforms a state-of-the-art algorithm with a 91.3% reduction in mean customer wait time.

Organization. The rest of this chapter is organized as follows: We first present in Section 3.2 a time-varying model for AMoD systems. In Section 3.3 we leverage the time-varying model to propose a MPC algorithm that relies on predicted future customer demand to control an AMoD system. Finally, in Section 3.4 we validate the approach on a real-world scenario based on a dataset of DiDi Chuxing, and characterize the performance of the proposed controller as a function of the prediction quality.

3.2 Model Description and Problem Formulation

In this section, we propose a time-varying network-flow model for AMoD systems that assumes perfect information is available about future customer arrivals. The model is amenable to efficient optimization: thus, it can be used to optimize the scheduling and routing for an AMoD system a posteriori. The model is not causal, and therefore can not directly be used for real-time control of an AMoD system; however, it forms the core of the MPC controller presented in the next section.

We consider an urban environment discretized into a set $\mathcal{N}$ of distinct regions (also known as stations in the AMoD literature [Pavone et al., 2012; Zhang and Pavone, 2016]). Time is represented by discrete intervals of a given size $\Delta t$. For a period under consideration of length $T$ time intervals, denote $\mathcal{T} = \{1, ..., T\}$ as the ordered set of time intervals. An AMoD system provides transportation services. We denote the travel time experienced by self-driving vehicles traveling from region $i \in \mathcal{N}$ to region $j \in \mathcal{N}$ as $\tau_{ij}$: that is, a vehicle departing region $i$ at time $t$ arrives in region $j$ at time $t' = t + \tau_{ij}$. The travel time can change depending on the departure time (modeling time-varying traffic conditions); however, the travel time is assumed to be independent of the number of vehicles traveling between the regions (i.e., congestion is considered an exogenous phenomenon).

Self-driving vehicles service customers’ transportation requests. We denote the number of customers who wish to travel from region $i \in \mathcal{N}$ to another region $j \in \mathcal{N}$, departing at time $t$, as $\lambda_{ijt}$. Similarly, we denote the number of vehicles that transport customers from $i \in \mathcal{N}$ to $j \in \mathcal{N}$ departing at time $t$ as $x^p_{ijt}$.

In order to satisfy the customer demand at each given region and time interval, the number of available vehicles at each time and in each region must be no smaller than the number of customers who wish to depart the region. To this end, it is necessary to recurrently rebalance the empty
vehicles from stations with an excess number of vehicles to stations with an insufficient number of available vehicles. We represent the number of empty, rebalancing vehicles traveling from \( i \in \mathcal{N} \) to \( j \in \mathcal{N} \) at time \( t \in \mathcal{T} \) as \( x^r_{ijt} \). When \( i = j \), the vehicles are considered to be idling, and this is regarded as a special case of rebalancing.

Finally, let \( s_{it} \) represent the number of initial available vehicles at region \( i \in \mathcal{N} \) at time \( t \in \mathcal{T} \). The variable \( s_{it} \) is free: the optimizer is allowed to determine the number and location of vehicles that is required to service all customers. Initial available vehicles can only be added at the first time interval: \( s_{it} = 0 \) \( \forall t > 1 \). The overall number of vehicles in the AMoD system is \( m = \sum_{i \in \mathcal{N}, t \in \mathcal{T}} s_{it} \).

### 3.2.1 Optimal Rebalancing Strategy

Under the assumption of perfect knowledge of customer demand, and assuming that the starting positions of the vehicles are free, it is possible to find the optimal rebalancing strategy by solving the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{(ij,t)} c^r_{ij} x^r_{ijt}, \\
\text{subject to} & \quad x^p_{ijt} = \lambda_{ijt}, \quad \forall i,j \in \mathcal{N}, t \in \mathcal{T}, \\
& \quad \sum_{j \in \mathcal{N}} x^p_{ijt} + x^r_{ijt} - x^p_{jit} - x^r_{jit} = s_{it}, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}, \\
& \quad s_{it} = 0, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}, t > 1, \\
& \quad x^r_{ijt}, s_{it} \in \mathbb{N}, \quad \forall i,j \in \mathcal{N}, t \in \mathcal{T},
\end{align*}
\]

where \( c^r_{ij} \) is the cost of rebalancing a vehicle from \( i \in \mathcal{N} \) to \( j \in \mathcal{N} \) (proportional to the travel time and distance between the regions). Idling vehicles are considered a special case of rebalancing denoted as \( x^i_{it} \) and have a corresponding cost \( c^i_{it} \), which captures time-only dependent costs (e.g. average cost of ownership per time interval) or costs specific to idling (e.g. parking). The decision variables are grouped into the following sets: \( \mathcal{X}^p = \{ x^p_{ijt} \}_{i,j,t} \), \( \mathcal{X}^r = \{ x^r_{ijt} \}_{i,j,t} \), \( \mathcal{S} = \{ s_{it} \}_{i,t} \). The constraint (3.1b) ensures that all customer demands are serviced, and constraint (3.1c) enforces that, for every time interval and each region, the number of arriving vehicles equals the number of departing vehicles. The constraint (3.1d) ensures that starting vehicles can only be inserted at the first time interval, and (3.1e) constrains the decision variables to be nonnegative integers.

Problem (3.1) always admits a feasible solution. The next theorem formalizes this intuition.

**Theorem 3.2.1** (Feasibility of the offline optimal rebalancing problem). Problem (3.1) admits a feasible solution for any set of customer demands \( \{ \lambda_{ijt} \}_{i,j,t} \).
3.3. MODEL PREDICTIVE CONTROL

Proof sketch: The number and location of available vehicles is a decision variable in Problem (3.1). Thus, a feasible solution exists where a vehicle is assigned to the departure location of each customer and idles at the location until the customer’s departure time.

A few comments are in order. First, while Problem (3.1) is stated as an integer linear program (ILP), it can be shown that the problem is totally unimodular and thus can be solved efficiently as a linear program (LP) [Ahuja et al., 1993]. Thus, very large instances of the problem can be solved efficiently on commodity hardware. Second, note that while we model idling as a special case of rebalancing, we can give it a different treatment by relying on its cost parameter. For example, the downtown area of a city might impose a charge for the time spent within the district or parking might be expensive in certain areas, thus, incentivizing vehicles to move. In Section 3.4, we assume that the cost per time interval of idling is simply related to time average cost of ownership, and, therefore, it is lower than rebalancing (since rebalancing additionally incurs, for example, fuel expenses). Finally, the starting positions are decision variables: thus, Problem (3.1) simultaneously finds the optimal rebalancing strategy and the corresponding fleet size required to execute the strategy. Therefore, the solution to this problem can be used to inform fleet sizing decisions based on historical customer demand.

3.3 Model Predictive Control

In this section, we propose an MPC implementation of the optimal rebalancing problem presented in Section 3.2.1 that leverages predictions of future demand. We begin by outlying the overall algorithm and then we delve into the details of each subcomponent.

3.3.1 Algorithm

Let $T$ be the planning horizon under consideration and $T_{\text{forward}}$ the forecasting horizon for which we can obtain a predicted customer demand (note that $T_{\text{forward}} \leq T$). Also, denote $\Lambda_{t,t'} = \{\lambda_{ijt} : \forall i, j \in N, t \in [t, ..., t']\}$ as the set of real customer demands from $t$ to $t'$, and $\hat{\Lambda}_{t,t'} = \{\hat{\lambda}_{ijt} : \forall i, j \in N, t \in [t, ..., t']\}$ as the set of predicted customer demands from $t$ to $t'$. $\{\lambda_{ij0}\}_{ij0}$ is the set of outstanding customer demand (customers who have requested a vehicle, but have not yet been served), and the rest of the notation is defined as in the previous section.

The proposed algorithm, summarized in Algorithm 1, is as follows: at a given time $t_0$ we first observe the system state to capture the vehicle availabilities, $S$, and the outstanding customer demand, $\{\lambda_{ij0}\}_{ij0}$. We then proceed to predict future customer demand $\hat{\Lambda}_{t_0,t_0+T_{\text{forward}}}$ for the next $T_{\text{forward}}$ time steps. Using this information, we compute the optimal rebalancing strategy $\mathcal{X}^*$ by solving a mixed-integer linear program (described in Section 3.3.4). Finally, we assign the rebalancing tasks corresponding to the first time interval to available vehicles as they become available.
CHAPTER 3. MODEL PREDICTIVE CONTROL OF AMOD SYSTEMS

After a period $\Delta t$, i.e. at $t_0 + 1$, we recompute the rebalancing strategy using Algorithm 1. Thus, this process is repeated during the entire operation of the system.

**Algorithm 1: Model Predictive Control**

1. **procedure** MPC
2. $\mathcal{S} \leftarrow$ count idle vehicles and estimate trip arrivals
3. $\lambda_{ij0} \leftarrow$ count outstanding customers
4. $\hat{\Lambda}_{t_0, t_0 + T_{\text{forward}}} \leftarrow f(\theta_i)$
5. $\mathcal{X}^p, \mathcal{X}^r, \mathcal{W}, \mathcal{D} \leftarrow$ solve Problem (3.5)
6. Assign $\{x'_{ij1}\}_{ij1}$ to available vehicles

### 3.3.2 State observation

At the beginning of each iteration of Algorithm 1, the first step is to capture the current system state in terms of vehicle availabilities and outstanding passengers.

Unlike in Section 3.2.1 where the vehicles’ starting locations, $\mathcal{S}$, are decision variables, during the operation of the system, the time and location at which the vehicles become available are fully determined by the current system state. Vehicles are considered available either when they are idling or after they complete a trip. Thus, at the start of the optimization process, let $a_i$ be the current number of idling vehicles at region $i \in \mathcal{N}$. Additionally, let $v_{it}$ be the number of vehicles traveling to region $j \in \mathcal{N}$ expected to arrive at time interval $t$. Then, the vehicle starting locations for the planning horizon are

$$s_{it} = \begin{cases} a_i + v_{it}, & \text{if } t = 1, \\ v_{it}, & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{N}, t \in \mathcal{T}. \quad (3.2)$$

Moreover, during real-time operations, the system may have to plan not only for predicted future customers but also for outstanding customers that were not serviced at previous time steps. Thus, in this step we count the outstanding travel requests and represent them by $\{\lambda_{ij0}\}_{ij0}$.

### 3.3.3 Forecasting

The second part of Algorithm 1 consists of predicting future customer demand. Let $f$ be a forecasting model trained with historical data, $\theta_i$ a diverse set of features relevant to the model available at the time of prediction (e.g. current traffic conditions, weather, recent travel demand, etc.), and $T_{\text{forward}}$ the forecasting horizon. Then, we denote $\hat{\Lambda}_{t+1, t+T_{\text{forward}}} = f(\theta_i)$ as the expected demand for the forecasting horizon, where $\hat{\Lambda}_{t+1, t+T_{\text{forward}}} = \{\hat{\lambda}_{ijt'}\}_{ijt'}, \forall i, j \in \mathcal{N}, t' \in [t + 1, \ldots, t + T_{\text{forward}}]$. The design of techniques for forecasting customer demand is beyond the scope of this thesis: we refer the reader to [Zhao et al., 2016] for a recent review. In Section 3.4, we propose a forecasting model based on neural networks and validate its performance with real customer data.
3.3. MODEL PREDICTIVE CONTROL

3.3.4 Controller

The third step computes the rebalancing strategy for the planning horizon \([1, ..., T]\) using the observed state and the predicted demand. To achieve this, we adapt Problem (3.1) for real-time usage by introducing several modifications to the problem formulation.

First, from the proof sketch of Theorem 3.2.1, note that the ensured feasibility of (3.1) depends on being able to choose the starting vehicle positions \(S\). Thus, for a fixed \(S\) and given customer demands, (3.1) might be infeasible. To ensure persistent feasibility of the MPC controller, we relax constraint (3.1b) by allowing the optimizer not to service certain customers. The slack variables \(D = \{d_{ijt}\}_{ijt}\) denote the predicted demand of customers wanting to travel from \(i\) to \(j\) departing at time \(t\) that will remain unsatisfied.

Second, outstanding customers may be at stations currently without any available vehicles: thus, it may be infeasible to pick them up at \(t = 1\). To mitigate this, we let the pickup time for outstanding customers be an optimization variable (with an associated cost that is proportional to the customers’ waiting time). Formally, we define \(w_{ijt}\) as the decision variable denoting the number of outstanding customers at region \(i \in \mathcal{N}\) who wish to travel to region \(j\) and be picked up at time \(t \in \mathcal{T}\). To ensure that all outstanding customers are considered, we include the following constraint:

\[
\sum_{t \in \mathcal{T}} w_{ijt} = \lambda_{ij0}, \quad \forall i, j \in \mathcal{N},
\]

where \(\lambda_{ij0}\) is the number of outstanding customers wanting to go from \(i\) to \(j\). Accordingly, the counterpart of Equation (3.1b) in the MPC controller is:

\[
x_{ijt}^p + d_{ijt} = \hat{\lambda}_{ijt} + w_{ijt}, \quad \forall i, j \in \mathcal{N}, t \in \mathcal{T}.
\]

Dropping customer demand and making outstanding customers wait are both undesirable. For a given origin-destination pair \(i, j \in \mathcal{N}\), we define \(c^w_{ijt}\) and \(c^d_{ijt}\) as the cost associated with a wait time of \(t\) time steps for an outstanding passenger, and the cost for not servicing a predicted customer demand at time \(t\), respectively.
We are now in a position to state the overall MPC optimization problem:

\[
\min_{X_p, X_r, W, D} \sum_{ijt} c_{ijt}^r x_{ijt}^r + c_{ijt}^w w_{ijt} + c_{ijt}^d d_{ijt},
\]  
\(3.5a\)

subject to

\[
x_{ijt}^p + d_{ijt} - w_{ijt} = \hat{\lambda}_{ijt}, \quad \forall i, j \in N, t \in T,
\]  
\(3.5b\)

\[
\sum_{j \in N} x_{ijt}^p + x_{ijt}^r - x_{jit}^p - \tau_{ij} - x_{jit}^r - \tau_{ij} = s_{it},
\]  
\(3.5c\)

\[
\sum_{t \in T} w_{ijt} = \lambda_{i,j,0}, \quad \forall i, j \in N,
\]  
\(3.5d\)

\[
x_{ijt}^p, x_{ijt}^r, w_{ijt}, d_{ijt} \in \mathbb{N}, \quad \forall i, j \in N, t \in T.
\]  
\(3.5e\)

Here, (3.5b) and (3.5c) are the passenger and vehicle continuity constraints, (3.5d) ensures that all outstanding passengers are served, and (3.5e) limits the decision variables to nonnegative integers.

### 3.3.5 Discussion

Note that the rebalancing, waiting and dropping costs are optimization parameters that should be set by the operator to reflect real-life costs. However, it is important to highlight that the relative difference between the waiting costs and the dropping costs should be carefully chosen: the optimizer will choose to drop a customer if it is cheaper than making her or him wait. Additionally, the planning horizon should be carefully chosen: an excessively short planning horizon may make it impossible for the optimizer to allocate rebalancing vehicles to customers stranded in remote regions, since the travel time might be longer than the planning horizon itself. Finally, Problem (3.5) is not totally unimodular, due to the presence of constraint (3.5d); therefore, the problem must be solved as a mixed-integer linear program (MILP). However, unlike existing MILP approaches in literature (e.g. Zhang et al., 2016a; Miao et al., 2016) the problem size of (3.5) does not grow with the number of vehicles – a remarkable fact that makes this approach suitable for large fleet sizes. Indeed, in Section 3.4 we show that modern MILP algorithms are able to solve Problem (3.5) quickly for large-scale problems based on real-world data.

### 3.4 Numerical Experiments

In this section, we first present numerical experiments comparing the performance of the proposed algorithm against an existing, high-performing rebalancing heuristic (Pavone et al., 2012, Zhang et al., 2016a). Then, we test the sensitivity of the algorithm to the length of the forecast horizon \(T_{\text{forward}}\). The experiments were carried on simulations based on a real-world dataset from
the Chinese ridesharing company Didi Chuxing.

### 3.4.1 Dataset

The DiDi dataset contains all trips requested by users in the city of Hangzhou from January 1 to January 21, 2016 (approximately eight million trips). For each trip, the dataset records the time of the request, the departure and destination locations (discretized in districts, or regions), a customer ID, a driver ID, and the price paid. Start locations are discretized into 66 districts (denoted as core districts), each identified by a hash. Destination locations are similarly discretized in 793 districts (a superset of the start locations). For the purpose of our numerical experiments, we disregarded trips that did not start and end in the core districts (approximately one million trips).

The dataset reports no geographic information about the location of the individual districts. Furthermore, no information is provided about the duration (and therefore the end time) of completed trips. However, using RideGuru [RideGuru, 2017] we were able to estimate the travel time of each trip from the trip price; we used this estimate to reconstruct the average travel time between each pair of districts.

### 3.4.2 Simulation environment

We simulate the operations of an AMoD system servicing customer trips from the DiDi dataset. The city is modeled as a set of regions, corresponding to the districts in the DiDi dataset. Each pair of regions is connected by a road whose travel time equals the average estimated travel time computed from the DiDi dataset.

Customer requests are “played back”: for each fulfilled transportation request in the dataset, we introduce in the simulation a customer request with the same start time, start location, and arrival location. If a customer request appears in a region where a vehicle is available, the customer is assigned to the vehicle and departs immediately. Otherwise, the customer waits in a queue for the next available vehicle. Vehicles remain idle at a region until they are assigned to either a customer request or a rebalancing task. The travel time of vehicles assigned to customer requests corresponds to the travel time of the actual corresponding trip in the dataset. In contrast, the travel time of vehicles assigned to a rebalancing task is the average estimated travel time between the origin and the destination region. The system state evolves in discrete time: every time step in the simulation corresponds to 6 seconds.

Every $\Delta t = 5$ minutes, we execute the MPC algorithm. For each region, the algorithm produced a (possibly empty) list of routes that empty vehicles should follow. The routes are then assigned to idle vehicles within each region. At the beginning of each iteration of Algorithm, the unused rebalancing tasks are deleted, and the process is repeated.
3.4.3 Forecasting

Long Short-Term Memory (LSTM) neural networks [Hochreiter and Schmidhuber, 1997] are a popular and effective method for forecasting time-series, and have increasingly gained attention in transportation demand forecasting (e.g. [Laptev et al., 2017, Song et al., 2016, Kea et al., 2017]).

In this chapter, we built forecasting model based on an LSTM neural network following the encoder-decoder architecture [Cho et al., 2014]. The model’s input is a multivariate time series describing the demand for each origin and destination pair for the last $T_{\text{back}}$ time steps. The model forecasts demand for each origin and destination pair for the following $T_{\text{forward}}$ time steps for each region. That is, for a trained model $f$ we forecast demand as follows:

$$\hat{\Lambda}_{t_0+1, t_0+T_{\text{forward}}} = f(\Lambda_{t_0-T_{\text{back}}, t_0}).$$

(3.6)

3.4.4 Detailed results for a single day

We simulated the day with highest activity, January 21, 2016 with 330,000 trips. We compared the performance of four different controllers:

- **MPC-Perfect.** The controller described in Algorithm [1] with a planning horizon of $T = 50$ using the exact customer demand as it appears in the dataset as a “forecast” for the next $T_{\text{forward}} = 24$ time intervals. The controller is non-causal; however, its performance offers an upper bound on the performance of the MPC algorithm.

- **MPC-LSTM.** The controller described in Algorithm [1] with a planning horizon of $T = 50$ using the model described in Section [3.4.3] to forecast customer demand for the next $T_{\text{forward}} = 24$ time intervals.

- **TV-Reactive.** The controller described in Algorithm [1] with a planning horizon of $T = 50$, but with the prediction set empty. In essence, this controller is simply a time-variant planner that “reacts” to outstanding customer demand.

- **Reactive.** The controller described in [Pavone et al., 2012]. The controller is based on a time-invariant model and reactive; however, it is shown to offer superior performance compared to several state-of-the-art rebalancing algorithms in [Zhang et al., 2016a].

For all scenarios, the rebalancing costs $c^r_{ijt}$ are proportional to the travel time $\tau_{ij}$. Moreover, we assume that the operator’s goal is to satisfy as many customer requests as possible. Thus, the cost of dropping predicted demand, $c^d_{ijt}$, is orders of magnitude larger than the cost of rebalancing, and the cost of waiting is $c^w_{ijt} = c^d_{ijt}/T$, such that making a customer wait for the entire planning horizon is equivalent to not satisfying the request.

To find an appropriate fleet size, we solved Problem (3.1) for the selected day: we found that the minimum number of vehicles required to satisfy customer demand without any waiting is 4206. In
order to account for the effect of imperfect information, we selected a fleet size of 5000 vehicles for the ensuing simulations. The initial location of the vehicles was equally distributed among the 66 regions. The forecasting model was trained with the first 15 days of the dataset, using a look-back horizon of $T_{\text{back}} = 60$ time steps, and a forecasting horizon of $T_{\text{forward}} = 24$ time steps.

A summary of the results can be appreciated in Table 3.1. As expected, the MPC controller with perfect information has the best performance, with a mean wait time of 3.6 seconds and a median wait time of 0. The TV-Reactive scenario has the worst performance, with a mean wait time double than that of Reactive. This is also not surprising, given that the Reactive algorithm in Pavone et al., 2012 has been shown to have excellent performance, and, without any forecasts, the MPC algorithm reacts only to outstanding passengers. Notably, however, the best performing causal controller, MPC-LSTM, has a mean wait time 91.3% shorter than that of the Reactive controller. This highlights the value of demand forecasting, even when imperfect, for operating the fleet.

<table>
<thead>
<tr>
<th>Wait time</th>
<th>MPC-Perfect</th>
<th>MPC-LSTM</th>
<th>Reactive</th>
<th>TV-Reactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean [s]</td>
<td>3.6</td>
<td>24.1</td>
<td>276.3</td>
<td>543.5</td>
</tr>
<tr>
<td>Median [s]</td>
<td>0.0</td>
<td>0.0</td>
<td>72.0</td>
<td>414.0</td>
</tr>
</tbody>
</table>

Table 3.1: Wait times for each scenario.

More detailed results for MPC-Perfect, MPC-LSTM, and Reactive are shown in Figure 3.1. We can see the stark difference in performance from the top chart showing the number of waiting customers at any given point in time. Notably, the Reactive scenario has significantly more waiting customers at any given point in time, peaking at 8,892 in the afternoon rush. In contrast, the maximum number of waiting customers with the MPC-LSTM algorithm is 843, during the morning rush. Much of the performance gain can be attributed to the possibility of preemptively rebalancing enabled by the forecasts. For example, both MPC-Perfect and MPC-LSTM issue a considerable number of rebalancing tasks around 6AM, right before the morning rush, while the Reactive controller only starts rebalancing once the rush begins. Note, however, that not all of the difference is due to preemptive rebalancing tasks. For example, the Reactive controller issues, in total, more than 3 times as many rebalancing tasks as MPC-LSTM, but, on average, has only 37% more vehicles rebalancing. These numbers imply that the rebalancing trips are shorter for the Reactive controller, and, thus, occur between nearby regions. However, in its attempt to keep equal availability across the city, vehicles are sent to regions where they might not be needed.

### 3.4.5 Comparison for different forecasting horizons

For the next set of experiments, we tested the sensitivity of the controller to different forecasting scenarios. Specifically, we varied the forecasting forward horizon, $T_{\text{forward}}$, and backward horizons, $T_{\text{back}}$, to different values spanning from 3 to 48 time steps (15min to 4 hours). Thus, we trained an LSTM specifically for each $T_{\text{forward}}$ and $T_{\text{back}}$ combination. Note that we kept the planning horizon
Figure 3.1: Results from the MPC-Perfect (gray), MPC-LSTM (blue), Reactive [Pavone et al., 2012] (orange), and TV-Reactive (green) scenarios at each time step. From top to bottom: Number of waiting customers; number of vehicles carrying passengers; number of vehicles rebalancing; and, number of rebalancing tasks issued.
fixed at $T = 50$ (4 hours and 10 minutes).

![Graph showing mean wait times for different combinations of $T_{\text{forward}}$ and $T_{\text{back}}$.](image)

Figure 3.2: Mean wait times for different combinations of $T_{\text{forward}}$ and $T_{\text{back}}$

Figure 3.2 shows the results. On the one hand, the backward horizon $T_{\text{back}}$ has little effect on the performance. This is likely due to the fact that time of the day and demand at the previous time step carry much more predictive power than the demand at even two time steps before (this was also observed empirically in [Kea et al., 2017]). On the other hand, the length of $T_{\text{forward}}$ follows a diminishing returns pattern, having significant, positive influence on the performance of the system at first, and leveling later. The early performance gains are likely due to the ability to foresee demand that would require long rebalancing travel times to satisfy.

### 3.4.6 Computational complexity

In general, mixed integer linear programs require expensive computations to solve. However, we show in simulation that the MILP in Problem (3.5) can be solved efficiently on commodity hardware with state-of-the-art solvers. We kept track of all the MILP instances realized during the experiments in Section 3.4.5. The simulations were run in a PC equipped with a 3.0 GHz Intel Core i7-5960 with 64GB of RAM, and we used IBM CPLEX [IBM, 1987] to solve the MILP instances. Table 3.2 reports our results. The average time required to solve a single instance of Problem (3.5) was 15.1s. In 7175 instances, no instance required more than 61 seconds to solve. This suggests that the MILP can be solved in real-time for control of real transportation networks.

<table>
<thead>
<tr>
<th>Samples</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>STD [s]</th>
<th>Max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>7175</td>
<td>15.1</td>
<td>14.0</td>
<td>8.7</td>
<td>61.0</td>
</tr>
</tbody>
</table>

Table 3.2: Optimization Running Time.
3.5 Conclusions

In this chapter, we presented a model-predictive control strategy that leveraged predicted customer demand to control the operations of an AMoD fleet. We first proposed a time-expanded network flow model for AMoD systems: the model allows us to compute the optimal rebalancing strategy and the minimum fleet size required to satisfy a given customer demand without waiting. We leverage this model to propose an MPC algorithm that relies on forecasted demand to control AMoD systems in real-time. Numerical simulations based on real-world data show that the algorithm scales well to large systems and outperforms state-of-the-art rebalancing strategies. Collectively, our results show that the incorporating forecasted demand in the synthesis of a rebalancing algorithm can yield very significant improvements in customer satisfaction, with 91.3% shorter customer wait times.

The work presented in this chapter opens several new avenues of research. First, it is of interest to extend this approach to incorporate other relevant and promising aspects of AMoD systems, such as including congestion, integration with the power grid, and coordination with public transit. As the model is expanded to include these approaches, in order to preserve computational tractability, it will be necessary to explore approximate solution techniques for mixed-integer linear programs, including rounding and randomized routing approaches. Thus, a second line of research would devise optimization algorithms that provide constraint satisfaction guarantees and bounded suboptimality. Third, the MPC algorithm outperformed the state-of-the-art Reactive controller in presence of a forecast - however, its performance when no forecast is available is significantly worse than the reactive algorithm’s. One possible approach to close this gap is to formulate a risk-averse MPC that takes into account uncertainty of the forecasts. Indeed, this idea is pursued in the next chapter where we develop a stochastic version of the MPC algorithm. This research route will require devising forecasting models that are able to predict not only the expected demand, but also its probability distribution. Therefore, a fourth route is to improve existing short-term forecasting techniques such that they enhance the end-to-end performance of the control system. Models that provide well-calibrated uncertainty distributions for their forecasts are of particular interest, as are forecasting models that leverage heterogeneous sources of data, such as weather, traffic, or cellular tower records.
Chapter 4

Stochastic Model Predictive Control for Autonomous Mobility on Demand

This chapter presents a stochastic, model predictive control (MPC) algorithm that leverages short-term probabilistic forecasts for dispatching and rebalancing Autonomous Mobility-on-Demand systems (AMoD), i.e. fleets of self-driving vehicles. We first present the core stochastic optimization problem in terms of a time-expanded network flow model. Then, to ameliorate its tractability, we present two key relaxations. First, we replace the original stochastic problem with a Sample Average Approximation, and provide its performance guarantees. Second, we divide the controller into two submodules. The first submodule assigns vehicles to existing customers and the second redistributes vacant vehicles throughout the city. This enables the problem to be solved as two totally unimodular linear programs, allowing the controller to scale to large problem sizes. Finally, we test the proposed algorithm in two scenarios based on real data and show that it outperforms prior state-of-the-art algorithms. In particular, in a simulation using customer data from the ridesharing company DiDi Chuxing, the algorithm presented here exhibits a 62.7 percent reduction in customer waiting time compared to state of the art non-stochastic algorithms.

4.1 Introduction

The opportunity presented by self-driving technology has spurred the development of controllers that attempt to optimally rebalance AMoD systems in real time. However, as we discuss in the literature review, most of the existing controllers either ignore future demand, assume deterministic future demand, or do not scale to large systems. In particular, while travel demand follows relatively
CHAPTER 4. STOCHASTIC MODEL PREDICTIVE CONTROL OF AMOD SYSTEMS

predictable patterns, it is subject to significant uncertainties due to externalities such as, e.g., weather and traffic. Successful AMoD systems must cope with these uncertainties. Thus, the goal of this chapter is to propose a stochastic model-predictive control approach for vehicle rebalancing that leverages short-term travel demand forecasts while considering their uncertainty. The controller relies on the time-varying network flow framework of Chapter 3 that presents practitioners with a significant modeling flexibility. Moreover, by introducing a series of mild relaxations, the controller consists of solving two totally unimodular linear programs, making it tractable for very large systems.

Literature Review. To keep this chapter concise, we limit our review to work that specifically addresses AMoD systems, although similar ideas can be found in the MoD literature as presented in the previous chapters. We categorize prior work in real-time control of AMoD systems in two broad classes: i) reactive control methods that do not make assumptions about future demand and ii) Model Predictive Control (MPC) algorithms that are able to leverage signals about future demand. Reactive, time-invariant methods span from simple bipartite matching, to control methods based on fluidic frameworks. A good comparison of different reactive controllers can be found in [Zhang et al., 2016a] and [Hörl et al., 2018], where, notably, both studies show that the controller first proposed in [Pavone et al., 2012] performs competitively across tests. However, these controllers do not provide a natural way to leverage travel demand forecasts.

In contrast, time-varying MPC algorithms, such as those proposed in [Zhang et al., 2016a], [Miao et al., 2017] and in Chapter 3, provide a natural way to leverage travel forecasts. However, [Zhang et al., 2016a] suffers from computational complexity as the fleet size grows and does not account for forecast uncertainty. While the approach presented in Chapter 3 shows encouraging experimental results, it also ignores forecast uncertainty and it can be shown that the difference between the stochastic optimum and the certainty equivalent one can be arbitrarily large. To address stochasticity of demand, [Miao et al., 2017] proposes a distributionally robust approach leveraging semidefinite programming. However, their model makes a restrictive Markovian assumption that exchanges fidelity for tractability. Moreover, the authors do not address how to recover integer rebalancing tasks from the fractional strategy provided by the controller.

To the best of our knowledge, there is no existing AMoD controller that i) exploits travel demand forecasts while considering its stochasticity, ii) produces actionable integer solutions for real-time control of AMoD systems, and iii) scales to large AMoD systems.

Statement of Contributions. The contributions of this chapter are threefold. First, we develop a stochastic MPC algorithm that leverages travel demand forecasts and their uncertainties to assign and reposition empty, self-driving vehicles in an AMoD system. Second, we provide high probability bounds on the suboptimality of the proposed algorithm when competing against an oracle controller which knows the true distribution of customer demand. Third, we demonstrate through experiments that the proposed algorithm outperforms the aforementioned deterministic counterparts when the demand distribution has significant variance. In particular, on the same DiDi Chuxing dataset, our
controller yields a 62.7 percent reduction in customer waiting time compared to the work presented in Chapter 3.

**Organization.** The remainder of the chapter is organized as follows. We introduce the AMoD rebalancing problem in Section 4.2 and we formulate it as an explicit stochastic integer program using a Sample Average Approximation (SAA) approach in Section 4.3. In Section 4.4 we discuss approximation algorithms to rapidly solve such an integer program, while in Section 4.5 we leverage the presented results to design a stochastic MPC scheme for AMoD systems. In Section 4.6 we compare the proposed MPC scheme against state-of-the-art algorithms using numerical simulations. Section 4.7 concludes the chapter with a brief discussion and remarks on future research directions.

### 4.2 Model and Problem Formulation

In this section, we first present a stochastic, time-varying network flow model for AMoD systems that will serve as the basis for our control algorithms. At its core, the model is the same the one presented in in Chapter 3. However, unlike in Chapter 3, we no longer assume perfect information about the future, instead we assume that customer travel demand follows an underlying distribution, which we may estimate from historical and recent data. Then, we present the optimization problem of interest: how to minimize vehicle movements while satisfying as much travel demand as possible. Finally, we end with a discussion on the merits and challenges of the model and problem formulation.

#### 4.2.1 Model

Let $G = (\mathcal{V}, \mathcal{E})$ be a weighted graph representing a road network, where $\mathcal{V}$ is the set of discrete regions (also referred to as stations), and the directed edges $\mathcal{E}$ represent the shortest routes between pairs of stations. We consider $G$ to be fully connected so there is a path between any pair of regions. Accordingly, let $n = |\mathcal{V}|$ denote the number of stations. We represent time in discrete intervals of fixed size $\Delta t$. The time it takes for a vehicle to travel from station $i$ to station $j$, denoted $\tau_{ij}$, is an integer multiple of $\Delta t$ for all pairs $i,j \in \mathcal{V}$.

At time $t$, we consider a planning horizon $\mathcal{T}$ of $T$ consecutive time intervals, i.e. $\mathcal{T} = [t + 1, t + 2, \ldots, t + T]$. For notational convenience and without loss of generality, we will always assume that the beginning of the planning horizon is at time $t = 0$. For each time interval in $\mathcal{T}$, $\lambda_{ijt}$ represents the number of future passengers that want to go from station $i$ to station $j$ at time interval $t$. However, the travel demand is a random process. Thus, we assume that the travel demand $\Lambda = \{\lambda_{ijt}\}_{i,j \in \mathcal{V}, t \in \mathcal{T}}$ within the time window $\mathcal{T}$ is characterized by a probability distribution $P$. Additionally, $\lambda_{ij0}$ denotes the number of outstanding passengers who have already issued a request to travel from $i$ to $j$ some time in the past but have not yet been serviced. Note that it is safe to assume that $\lambda_{ij0}$ is always known (since keeping track of waiting customers is relatively trivial) and, therefore, deterministic.

Within the same time window, there are $m$ self-driving vehicles which are either idling, serving
a customer, or executing a rebalancing task. Thus, the availability of these vehicles is location and
time-dependent. Specifically, \( a_i \) is the number of idle vehicles at the beginning of the time window
at station \( i \), and \( v_{it} \) the number of vehicles which are currently busy, but will finish their current
task and become available at time \( t \) at station \( i \). Thus, the total number of available vehicles in the
system as a function of location and time is given by

\[
s_{it} := \begin{cases} 
  a_i + v_{it} & \text{if } t = 1, \\
  v_{it} & \text{if } t > 1. 
\end{cases}
\]

Vehicle movements are captured by \( x \), i.e., \( x_{ijt} \) is the number of cars, rebalancing or serving
customers, which are departing from \( i \) at time \( t \) and traveling to \( j \). Vehicles must satisfy flow
conservation, such that, the number of vehicles arriving at a station at a particular time equals the
number of departing vehicles. Formally:

\[
\sum_{j=1}^{n} x_{ijt} = s_{it} + \sum_{j=1}^{n} x_{ji(t-\tau_{ji})}, \forall i \in \mathcal{V}, t \in \mathcal{T}.
\]  

(4.1)

Note that, unlike in Chapter 3, we do not make an explicit distinction between rebalancing and
passenger-carrying vehicles, instead we have lumped them together into \( x \). This will not be an issue
since it is possible to recover the rebalancing tasks for a given \( x \).

Finally, \( w \) captures outstanding customers, such that \( w_{ijt} \) is the number of outstanding customers
who waited until time \( t \) to be transported from station \( i \) to station \( j \). All outstanding customers
must be served within the planning horizon:

\[
\sum_{t=0}^{T} w_{ijt} = \lambda_{ij0} \forall i, j \in \mathcal{V}.
\]  

(4.2)

4.2.2 Problem Formulation

Our objective is to minimize a combination of i) the operational cost based on vehicle movement,
ii) the waiting time for outstanding customers and iii) the expected number of customers who upon
arrival do not find an available vehicle in their region. Given \((\zeta)_+ := \max\{0, \zeta\}\) and a vehicle
availability state \( \{s_{it}\}_{i \in \mathcal{V}, t \in \mathcal{T}} \), the goal is to solve the following optimization problem:
The first term in the objective, where \( c_x := \{c_{x,ijt}\}_{i,j \in V, t \in T} \), is the operational cost, i.e. the cost of operating the fleet (including, e.g., fuel, maintenance, depreciation) in proportion to total distance traveled. Similarly, the second term \( c_T w \) penalizes customer waiting times by a cost vector \( c_w \), where \( c_{w,ijt} \) is the cost of making an outstanding customer wanting to travel between stations \( i \) and \( j \) wait until time interval \( t \) to be served. The last term penalizes the expected mismatch between customer demand and the vehicle supply, that is \( c_{\lambda,ijt} \) is the cost of not being able to serve a customer wanting to travel between \( i \) and \( j \) at time \( t \). Finally, in addition to the previously mentioned constraints, \( x \) and \( w \) must be positive integers since fractional vehicles and customers are non-physical.

### 4.2.3 Discussion

There are two key challenges in solving (4.3). First, \( P \) is a time varying high dimensional probability distribution which is generally not known. Hence, one cannot evaluate the objective function explicitly. Second, due to the integer constraints on \( x, w \), (4.3) is an instance of integer programming which is NP-hard, such that no polynomial time algorithms exist and the problem remains computationally intractable for large inputs.

In the following sections, we present a series of relaxations that allow us to efficiently obtain solutions to a surrogate problem that approximates (4.3). Specifically, to address the unknown distribution in the objective function, we fit a conditional generative model on historical data to predict future demand given recent realizations of demand. To address the computational complexity of integer programming, we perform several relaxations to arrive at a linear programming surrogate problem. Finally, we present bounds on the optimality gap induced by making these relaxations.

### 4.3 Sample Average Approximation Techniques

Since \( P \) is an unknown, time varying distribution, we cannot explicitly evaluate the objective in (4.3). To address this issue, we present a SAA problem whose objective function approximates the objective of (4.3) in section 4.3.1. In section 4.3.2 we give sufficient conditions under which the solution to the SAA problem from 4.3.1 is near optimal for the original problem. We address the trade-off between solution accuracy and problem complexity in section 4.3.3.
4.3.1 Sample Average Approximation for AMoD control

Despite not knowing $P$, nor being able to sample from it, we have historical data from $P$ that we use to train a conditional generative model $\hat{P}$ to mimic the behavior of $P$. With a generative model in hand, one can consider solving (4.3) with $\hat{P}$ instead of $P$.

However, in many cases solving a stochastic optimization problem exactly is not possible if the underlying distribution does not have a computationally tractable form. Many popular probabilistic generative models, such as Bayesian networks and Bayesian neural networks fall into this category. To overcome this issue, we can sample from the generative model and replace expectations with Monte Carlo estimates to get approximate solutions, a method commonly referred to Sample Average Approximation (SAA) [Homem-de Mello and Bayraksan, 2014, Birge and Louveaux, 2011]. To this end we generate $K$ samples $\{(\lambda_{ijt}^k)_{i,j \in [n], t \in [T]}\}_{k=1}^K \overset{i.i.d.}{\sim} \hat{P}$ and approximate expectations under $\hat{P}$ with Monte Carlo estimates, i.e.

$$
\mathbb{E}_{\hat{P}} \left[ \sum_{ijt} (\lambda_{ijt} + w_{ijt} - x_{ijt})^+ \right] \approx \frac{1}{K} \sum_{k=1}^K \sum_{ijt} (\lambda_{ijt}^k + w_{ijt} - x_{ijt})^+.
$$

Using this approximation, we consider the following SAA surrogate problem:

$$
\min_{\{u_k\}_{k=1}^K, x, w} \begin{array}{l}
 c_x^T x + c_w^T w + c_{\lambda} \sum_{k=1}^K \sum_{ijt} u_{ijt}^k \\
 \text{s.t.} \sum_{t \in T} w_{ijt} = \lambda_{ij0} \quad \forall i, j \in [n] \\
 \sum_{j=1}^n x_{ijt} - x_{ji(t-\tau_{ji})} = s_t \quad \forall i \in [n], t \in T \\
 u_{ijt}^k \geq 0 \quad \forall k \in [K], i, j \in [n], t \in T \\
 u_{ijt}^k \geq \lambda_{ijt}^k + w_{ijt} - x_{ijt} \quad \forall k \in [K], i, j \in [n], t \in T \\
 \{u_k\}_{k=1}^K, x, w \in \mathbb{N}^{n^2 T} \quad \forall k \in [K], i, j \in [n], t \in T,
\end{array}
$$

where, in addition to the Monte Carlo estimate, we include a series of inequalities to make the objective function linear. Specifically, minimizing $(x)_+$ is equivalent to minimizing $u$ with the constraints $u \geq 0, u \geq x$. The surrogate SAA problem (4.4) is directly solvable by off-the-shelf mixed integer linear programming (MILP) solvers.

4.3.2 Oracle inequality performance guarantees for SAA

Sample Average Approximation is not guaranteed in general to provide asymptotically optimal solutions to the population problem as the number of samples goes to infinity. While the objective
of an SAA problem converges pointwise to the population objective, if the convergence is not uniform, SAA may return solutions that do not converge to the optimal population value even as the number of samples goes to infinity. In this section, we compare the quality of the solutions to (4.3) and (4.4) when evaluated by the objective in (4.3). Specifically, we present a result stating that if \( \hat{P} \) is close to \( P \) in an appropriate sense and we use enough samples for the SAA in (4.4), then the obtained solution is with high probability, provably near optimal for the original problem in (4.3) that we would have solved had we known \( P \). Such a result is called an oracle inequality. Using the notation

\[
F(x, w) := c_{\lambda} \mathbb{E}_P \left[ \sum_{ijt} (\lambda_{ijt} + w^*_ijt - x^*_ijt)_+ \right] \quad \text{and} \quad \hat{F}_K(x, w) := \frac{c_{\lambda}}{K} \sum_{k=1}^{K} \left[ \sum_{ijt} (\lambda^k_{ijt} + \hat{w}_{ijt} - \hat{x}_{ijt})_+ \right],
\]

the difference between the objectives in (4.3) and (4.4) is \( F(x, w) - \hat{F}_K(x, w) \). Consider the following lemma:

**Lemma 4.3.1** (\( \| \cdot \|_\infty \)-continuity of function minima). Let \( f, g : \mathcal{X} \to \mathbb{R} \) denote two real valued functions that have finite global minima, i.e., both \( x_f := \arg \min_{x \in \mathcal{X}} f(x) \) and \( x_g := \arg \min_{x \in \mathcal{X}} g(x) \) exist. Then,

\[
f(x_g) \leq f(x_f) + 2 \sup_{x \in \mathcal{X}} |f(x) - g(x)|.
\]

**Proof.** Given \( f, g : \mathcal{X} \to \mathbb{R} \) define \( x_f := \arg \min_{x \in \mathcal{X}} f(x) \) and \( x_g := \arg \min_{x \in \mathcal{X}} g(x) \). Let \( ||f - g||_\infty := \sup_{x \in \mathcal{X}} |f(x) - g(x)| \). We then see that

\[
\begin{align*}
f(x_g) &\leq g(x_g) + ||f - g||_\infty \\
&\leq g(x_f) + ||f - g||_\infty \\
&\leq [f(x_f) + ||f - g||_\infty] + ||f - g||_\infty \\
&= f(x_f) + 2||f - g||_\infty,
\end{align*}
\]

which is the desired result. \( \square \)

Applying this idea to the AMoD setting, let \( (x^*, w^*) \) be a solution to (4.3), and \( (\hat{x}, \hat{w}) \) a solution to (4.4). If \( \max_{x, w} |F(x, w) - \hat{F}_K(x, w)| < \epsilon \) is small, then \( (\hat{x}, \hat{w}) \) will be at most \( 2\epsilon \) worse than \( (x^*, w^*) \) when evaluated by \( F \). It is then of interest to understand the conditions for which \( \hat{F}_K \) will
be uniformly close to $F$. Since $\hat{F}_K$ is a random object, its error in estimating $F$ has two contributors: stochastic error and model error. Specifically, the stochastic error is due to the error induced by estimating expectations under $\hat{P}$ using SAA, and the model error is the error incurred when estimating the true distribution $P$ using $\hat{P}$. For the analysis, we will need the following definition.

**Definition 4.3.2. Sub-exponential Random Variables**

A random vector $X \in \mathbb{R}^d$ is sub-exponential with parameters $\sigma^2, b < \infty$ if, for any $v \in \mathbb{R}^d$ satisfying $||v||_2 \leq b^{-1}$, the following inequality holds:

$$\log \mathbb{E} \left[ e^{v^T (X - \mathbb{E}X)} \right] \leq \frac{||v||_2^2 \sigma^2}{2}.$$ 

Intuitively, a random variable is sub-exponential if its tails decay at least as fast as that of an exponential random variable.

**Lemma 4.3.3 (Uniform Convergence for SAA).** Let $P$ be the true distribution of customer demand, $\hat{P}$ be the distribution of predicted customer demand and let $P_{ijt}, \hat{P}_{ijt}$ be the distribution of $\lambda_{ijt}$ under $P, \hat{P}$ respectively. Assuming that $\lambda \sim \hat{P}$ is $(\sigma^2, b)$ sub-exponential, then for any $\delta > 0$, with probability $1 - \delta$, the following holds:

$$\max_{x,w} |F(x,w) - \hat{F}_K(x,w)| \leq \frac{2\sigma}{\sqrt{K}} \sqrt{n^3T \log(m) + \log \frac{1}{\sqrt{\delta}}} + \sqrt{\text{Var}_P(||\lambda||_2)}.$$ 

where $\chi(\hat{P}||P) \in \mathbb{R}_+^{n^2T}$, $\chi(\hat{P}||P)_{ijt} = \chi(\hat{P}_{ijt}||P_{ijt})$ and $\chi^2(\cdot||\cdot)$ represents the $\chi^2$-divergence between probability distributions which is non-negative and zero if and only if its arguments are equal.

**Proof.** First recall the definition of the $\chi^2$-divergence $\chi^2(P||Q)$ between two probability distributions $P, Q$.

$$\chi^2(P||Q) := \mathbb{E}_Q \left[ \left(1 - \frac{dP}{dQ}\right)^2 \right] \text{ if } P \ll Q, +\infty \text{ else}.$$ 

The function $\chi^2(P||Q)$ is non-negative and is zero if and only if $P = Q$. Since $(\cdot)_+$ is $1$-Lipschitz,
4.3. SAMPLE AVERAGE APPROXIMATION TECHNIQUES

we have:

\[
F(x, w) - \mathbb{E}_{\hat{P}} \hat{F}_K(x, w) = \sum_{ijt} \sum_{\lambda \in \mathbb{N}} (\lambda + w_{ijt} - x_{ijt})_+ (P_{ijt}(\lambda) - \hat{P}_{ijt}(\lambda))
\]

Define \( \ell_{ijt}(\lambda) := (\lambda + w_{ijt} - x_{ijt})_+ - \mathbb{E}_{P_{ijt}} (\lambda + w_{ijt} - x_{ijt})_+ \), we have:

\[
F(x, w) - \mathbb{E}_{\hat{P}} \hat{F}_K(x, w) = \sum_{ijt} \sum_{\lambda \in \mathbb{N}} \ell_{ijt}(\lambda) (P_{ijt}(\lambda) - \hat{P}_{ijt}(\lambda))
\]

Since \( \sum_{\lambda \in \mathbb{N}} C (P_{ijt}(\lambda) - \hat{P}_{ijt}(\lambda)) = 0 \) for any constant \( C \), we let \( C = \mathbb{E}_{P_{ijt}} (\lambda + w_{ijt} - x_{ijt})_+ \).

\[
\sum_{ijt} \sum_{\lambda \in \mathbb{N}} \ell_{ijt}(\lambda) (P_{ijt}(\lambda) - \hat{P}_{ijt}(\lambda))
\]

\[
= \sum_{ijt} \sum_{\lambda \in \mathbb{N}} \ell_{ijt}(\lambda) \sqrt{P(\lambda)}_{ijt} \left( \sqrt{P(\lambda)}_{ijt} - \frac{\hat{P}_{ijt}(\lambda)}{P_{ijt}(\lambda)} \right)
\]

\[
\leq \sqrt{\text{Var}_P(||(\lambda + w - x)_+||_2)} \sqrt{\sum_{ijt} \lambda^2 (\hat{P}_{ijt}||P_{ijt})}
\]

The first inequality is due to Cauchy-Schwarz in \( L^2_P \), and the second inequality is due the fact that \( \text{Var}(X_+) \leq \text{Var}(X) \) for any random variable \( X \) by the following calculations:

\[
\text{Var}(X_+) := \mathbb{E}[(X_+ - \mathbb{E}[X_+])^2]
\]

\[
\leq \mathbb{E}[(X_+ - \mathbb{E}[X_+])^2] + (\mathbb{E}[X_+] - \mathbb{E}[X_+])^2
\]

\[
= \mathbb{E}[(X_+ - \mathbb{E}[X_+])^2]
\]

\[
\leq \mathbb{E}[(X - \mathbb{E}[X])^2]
\]

\[
= \text{Var}(X)
\]

The second inequality is because \((\cdot)_+\) is a 1-Lipschitz function. It is also possible to control the model error using the RMSE of the generative model \( \hat{P} \), but that bound is weaker than what is presented here. For the standard deviation bound, we use the concentration of measure for sub-exponential random variables. A random variable \( X \) is sub-exponential if there exists parameters \( \sigma^2, b \) so that
for any $|\lambda| \leq b^{-1}$, we have

$$\log \mathbb{E}[e^{\lambda(X - \mathbb{E}X)}] \leq \frac{\lambda^2 \sigma^2}{2}$$

The following probability bounds for sub-exponential random variables are well known: If $X$ is $(\sigma^2, b)$ sub-exponential, then:

$$\mathbb{P}[|X - \mathbb{E}X| > t] \leq \exp \left(-\frac{t^2}{2\sigma^2}\right)$$

if $t \leq \frac{\sigma^2}{b}$

$$\mathbb{P}[|X - \mathbb{E}X| > t] \leq \exp \left(-\frac{t}{2b}\right)$$

otherwise

Let $\{\lambda^1, ..., \lambda^K\}$ be samples from $\hat{P}$ used to form the objective function. Let $\hat{F}_{x,w}(\lambda) := \sum_{ijt} (\lambda_{ijt} + w_{ijt} - x_{ijt})$. $L$-Lipschitz functions of sub-exponential random variables are also sub-exponential with the same parameters, thus $\{\hat{F}_{x,w}(\lambda^k)\}_{k=1}^K$ are i.i.d. $(\frac{\sigma^2}{K}, \frac{b}{K})$-sub-exponential random variables. Thus, the objective of (4.4), $\frac{1}{K} \sum_{k=1}^K \hat{F}_{x,w}(\lambda^k)$ is $(\frac{\sigma^2}{K}, \frac{b}{K})$ sub-exponential. Applying the first part of the probability bound, we see that:

$$\mathbb{P} \left( \left| \frac{1}{K} \sum_{k=1}^K \hat{F}_{x,w}(\lambda^k) - \mathbb{E}\hat{F}_{x,w}(\lambda) \right| > t \right) \leq \exp \left( -\frac{Kt^2}{2\sigma^2} \right)$$

for any $t < \frac{\sigma^2/K}{K} = \sigma^2/b$. For any $\delta > 0$ error tolerance, setting $t = \frac{2\sigma}{\sqrt{K} \sqrt{n^2 T \log(m) + \log(\delta^{-1/2})}}$, for sufficiently large $K$ the bound evaluates to $\delta (m)^{-n^2 T}$. However this inequality only applies to a particular pair of $x, w$. Since $x \in \mathbb{R}^{n^2 T}$ and $||x||_\infty \leq m$, $x$ can take at most $|m|^{n^2 T}$ many distinct values. Note that if $w_{ijt} > m$ we can always set $w_{ijt} = m$ without affecting performance because the system cannot pick up more than $m$ waiting customers at any time. Thus, we also have $||w||_\infty \leq m$ and hence $w$ can take at most $m^{n^2 T}$ values. Thus there are at most $m^{2n^2 T}$ possible plans $(x, w)$. Taking a union bound over all possible $x, w$ gives, with probability at least $1 - \delta$,

$$\left\| \frac{1}{K} \sum_{k=1}^K \hat{F}_{x,w}(\lambda^k) - \mathbb{E}\hat{F}_{x,w}(\lambda) \right\|_\infty \leq \frac{2\sigma}{\sqrt{K} \sqrt{n^2 T \log(m) + \log(\delta^{-1/2})}}$$

Applying lemma 4.3.1 with this bound yields the desired result.

Note that the assumption of sub-exponential $\lambda$ is not very restrictive. Indeed, many common distributions including gaussian, Poisson, chi-squared, exponential, geometric, and any bounded random variables are all sub-exponential [Vershynin, 2018]. If we denote the solution to (4.3) as $(x^*, w^*)$ and the solution to (4.4) as $(\hat{x}, \hat{w})$, then applying lemmas 4.3.1 and 4.3.3 the following
happens with probability at least $1 - \delta$.

\[
\frac{1}{2} \left( F(\widehat{x}, \widehat{w}) - F(x^*, w^*) \right) \leq 2\sigma \sqrt{\frac{n^2T \log(m) + \log \frac{1}{\sqrt{\delta}}}{K \sqrt{n^2T \log(m) + 0.5 \log \delta}}}.
\]

This result implies that, for a desired accuracy $\epsilon > 0$, if we fit a generative model $\widehat{P}$ satisfying $||\chi(\widehat{P}|P)||_2 \leq 0.25\epsilon \text{Var}(||\lambda||_2)^{-1/2}$ and we use at least $K \epsilon \sigma^2 \geq 64 \epsilon^2 (n^2T \log(m) - 0.5 \log \delta)$ samples for the SAA in (4.4), then the solution to (4.4) will be at most $\epsilon$ worse than the optimal solution to (4.3) with known $P$.

### 4.3.3 Computational Complexity

As shown in lemma 4.3.3, the sampling error of (4.4) is $O(K^{-1/2})$, where $K$ is the number of samples used to form the SAA objective. On the other hand, the computational complexity of (4.4) is an increasing function of $K$, so in this section we discuss how the problem size of (4.4) depends on $K$.

A naive implementation of (4.4) would allocate $Kn^2T$ decision variables for $\{u^k\}_{k=1}^K$, and a linear dependence on $K$ which would lead to scalability issues since integer programming is NP hard in the worst case. However, note that in an optimal solution, we will have $u^k_{ijt} = (\lambda^k_{ijt} + w_{ijt} - x_{ijt})_+$. Thus if for some $k, l$ we have $\lambda^k_{ijt} = \lambda^l_{ijt}$, then the optimal solution has $u^k_{ijt} = u^l_{ijt}$. In this case, solving (4.4) with the additional constraint of $u^k_{ijt} = u^l_{ijt}$ will still yield the same optimal value while reducing the number of decision variables by one. Therefore, for each trip type $(i, j, t)$, instead of needing $K$ decision variables $\{u^k_{ijt}\}_{k=1}^K$, we only need $c$ decision variables, where $c$ is the number of unique values in the set $\{\lambda^k_{ijt}\}_{k=1}^K$. The following lemma demonstrates the reduction in complexity achievable by this variable elimination procedure.

**Lemma 4.3.4** (SAA Problem Size for Subexponential Demand). Assume that $\lambda \sim \widehat{P}$ is subexponential with parameters $\sigma^2, b$. For any $\delta > 0$, with probability at least $1 - \delta$, the number of distinct realizations of the customer demand is no more than $O \left( n^2T \min \left( \log \frac{Kn^2T}{\delta}, K \right) \right)$. Thus, as long as $n^2T$ is not exponentially larger than $K$, a variable elimination procedure ensures that the number of decision variables scales as $O(\log K)$, as opposed to the linear scaling $O(K)$ that the naive implementation would lead one to believe.

**Proof.** Here we use sub-exponential concentration inequalities to obtain a bound on the maximum
and minima of i.i.d. sub-exponential random variables. Lemma 4.3.4 is related to the distribution of maxima of sub-exponential random variables. Let $X_1, ..., X_n$ be i.i.d. zero mean $(\sigma^2, b)$ sub-exponential random variables. We proceed by the standard Chernoff bounding technique. For any $0 < \lambda \leq b^{-1}$, we have:

$$
\Pr \left[ \max_{1 \leq i \leq n} X_i \geq t \right] = \Pr \left[ e^{\lambda \max_{1 \leq i \leq n} X_i} \geq e^{\lambda t} \right]
= \Pr \left[ \max_{1 \leq i \leq n} e^{\lambda X_i} \geq e^{\lambda t} \right]
= e^{-\lambda t} \mathbb{E} \left[ \max_{1 \leq i \leq n} e^{\lambda X_i} \right]
\leq e^{-\lambda t} \mathbb{E} \left[ \sum_{1 \leq i \leq n} e^{\lambda X_i} \right]
\leq n \exp \left( -\lambda t + \frac{\lambda^2 \sigma^2}{2} \right)
= \exp \left( -\lambda t + \frac{\lambda^2 \sigma^2}{2} + \log n \right)
$$

If $t > \frac{\sigma^2}{b}$ then setting $\lambda = \frac{1}{b}$ gives the tightest upper bound, in which case we have:

$$
\Pr \left[ \max_{1 \leq i \leq n} X_i \leq t \right] \leq \exp \left( -\frac{t}{b} + \frac{\sigma^2}{2b^2} + \log n \right)
$$

but recall that $t > \frac{\sigma^2}{b} \implies \frac{\sigma^2}{2b^2} \leq \frac{t}{2b}$, meaning

$$
\Pr \left[ \max_{1 \leq i \leq n} X_i \leq t \right] \leq \exp \left( -\frac{t}{2b} + \log n \right)
$$

thus for any $\delta > 0$, setting $t = 2b \log \frac{n}{\delta}$, the upper bound is equal to $\delta$. Hence,

$$
\Pr \left[ \max_{1 \leq i \leq n} X_i \geq 2b \log \frac{n}{\delta} \right] \leq \delta
$$

The concentration of the minimum is analogous, by noting that $-X$ is also sub-exponential, and applying the above argument. Applying this to our problem, if the demand $\lambda_{ijt}$ for each $(i, j, t)$ is $(\sigma_{ijt}^2, b)$ sub-exponential, and $K$ samples $\lambda_{ijt}^1, ..., \lambda_{ijt}^K$ are observed, then by the above argument, with probability at least $1 - \frac{\delta}{n^2T}$ all samples fall in the interval

$$
\left[ \mathbb{E} [\lambda_{ijt}] - 2b \log \frac{Kn^2T}{\delta}, \mathbb{E} [\lambda_{ijt}] + 2b \log \frac{Kn^2T}{\delta} \right].
$$
But since these samples are integer valued, if they lie in an interval of size $O(\log K)$, then there can be at most $O(\log K)$ distinct samples of the demand. Taking a union bound over all tuples $(i, j, t)$, we have that with probability at least $1 - \delta$, for each tuple $(i, j, t)$, the number of unique elements in $\{\lambda_{kij}^k\}_{k=1}^K$ is at most $4b \log \frac{K n^2 T}{\delta}$. Summing over all $(i, j, t)$ we then have that the total number of decision variables is at most $4bn^2T \log \frac{K n^2 T}{\delta}$. The number of decision variables is trivially at most $Kn^2T$, therefore taking the better of the two bounds yields the result.

Thus with high probability the number of decision variables will be logarithmic in $K$, which is an exponential improvement over the linear dependence that the naive implementation proposes. This is especially important since using large $K$ gives an objective function with less variance.

4.4 Totally Unimodular Linear Relaxations

Recall from (4.5) that increasing the number of samples $K$ used for Monte Carlo reduces the standard deviation of the random objective in (4.4), thereby increasing the quality of the algorithm’s output. While we showed that the number of decision variables is only logarithmic in the sample size $K$, the problem is still NP-hard. Thus, increasing the number of samples used in (4.4) may not be tractable in large scale settings. In this section, we propose a modified algorithm that solves a convex relaxation of (4.4), which is scalable to large problem sizes.

Our relaxation separately addresses the tasks of servicing existing customers and rebalancing vacant vehicles that are jointly solved in (4.4). Note that information about future customers can affect scheduling of waiting customers and vice versa in the optimal solution. In such a situation, servicing existing customers and rebalancing vacant vehicles with two separate algorithms prevents the sharing of information and can lead to suboptimal solutions. Nevertheless, this procedure runs in polynomial time, as opposed to integer programming. It is important to note, however, that solutions to convex relaxations of combinatorial problems need not be integral, and in this case naive rounding techniques can lead to violations of the network flow constraints. We obtain integer solutions by showing that our convex relaxations are totally unimodular linear programs. A linear program being totally unimodular means that it always has optimal solutions that are integer valued [Ahuja et al., 1993], and can thus be obtained using standard interior point optimization methods.

Network flow minimization problems are linear programs with constraints of the form (4.1), and preserve total unimodularity. However, in the case of problem (4.4), the inclusion of the constraints (4.2) break this totally unimodular structure, and hence solving a relaxation of (4.4) with the $x, w \in \mathbb{N}^{n^2 T}$ constraint removed is not guaranteed to return an integer solution. Alternatively, if we first assign vehicles to service existing customers, then the problem of rebalancing the empty vehicles no longer has constraints of type (4.2), and becomes totally unimodular. Inspired by this fact, in section 4.4.1 we discuss a bipartite matching algorithm we use to assign vacant vehicles to waiting customers, and in Section 4.4.2 we solve a totally unimodular version of (4.4) to determine
4.4.1 Bipartite Matching for Servicing Waiting Customers

We use a bipartite matching algorithm to pick up waiting customers in a way that minimizes the total waiting time. Specifically, the current state of the system is \( z, d \in \mathbb{N}^n \), where \( z_i \) is the number of vehicles currently available at station \( i \), and \( d_i \) is the number of outstanding customers at station \( i \). The decision variable is a vector \( x \in \mathbb{R}^{n^2} \) where \( x_{ni+j} \) is the number of vehicles sent from station \( i \) to station \( j \). Let \( A := \mathbb{1}_n^T \otimes I_n - I_n \otimes \mathbb{1}_n^T \), where \( \mathbb{1}_n \) is the vector of all 1’s in \( \mathbb{R}^n \), \( I_n \) is the identity matrix of size \( n \times n \) and \( \otimes \) is the matrix Kronecker product. Using this notation, the resulting state of taking action \( x \) in vehicle state \( z \) is simply \( z + Ax \). To satisfy the customers, we want \( Ax + z \geq y \) elementwise. If this is not possible, we will pay a cost of \( c_\lambda \) for every customer we do not pick up. To capture this, we define a drop vector \( u = (y - Ax - z)_+ \). The cost vector \( c \in \mathbb{R}^{n^2} \) is defined so that \( c_{i+n+j} \) is the travel time between \( i, j \). Thus, the optimal solution to the bipartite matching problem is obtained by solving the following linear program:

\[
\begin{align*}
\min_{x,u} & \quad c^T x + \mathbb{1}_n^T u \\
\text{s.t.} & \quad u \succeq 0 \\
& \quad u \succeq y - (Ax + z) \\
& \quad x \in \mathbb{R}^{n^2}, u \in \mathbb{R}^n.
\end{align*}
\]

It can be shown that bipartite matching has the totally unimodular property, and, therefore, will return integer solutions when the constraints are also integer.

4.4.2 Network Flow Optimization for Rebalancing Vehicles

To rebalance vacant vehicles in anticipation of future demand, we now solve (4.4) with \( w = 0 \) to obtain a rebalancing flow. We have \( w = 0 \) because the task of picking up outstanding customers is given to a bipartite matching algorithm, and hence does not need to be considered here. In this case, we can relax the integer constraints on \( x \) to obtain a totally unimodular linear program according to Lemma 4.4.1.

Lemma 4.4.1 (Totally Unimodular SAA Rebalancing Problem). Consider the following convex
relaxation of (4.4) where \( w = 0 \):

\[
\begin{align*}
\min_{x,w} c^T x + & \frac{1}{K} \sum_{k=1}^{K} \sum_{ijt} u_{ijt}^k \\
\text{s.t.} & \sum_{j=1}^{n} x_{ijt} - x_{j(i-t)} = s_{it} \text{ for all} \\
& i \in [n] \text{ and } t_0 < t \leq t_0 + T \\
& u_{ijt}^k \geq 0 \quad \forall k \in [K], i,j \in [n], t \in T, \\
& u_{ijt}^k \geq \lambda_{ijt}^k - x_{ijt} \quad \forall k \in [K], i,j \in [n], t \in T \\
& \{u_{ijt}^k\}_{k=1}^{K}, x \in \mathbb{R}^{n^2T} \forall k \in [K], i,j \in [n], t \in T.
\end{align*}
\]

This problem is totally unimodular.

Proof. Define \( u = [u^1, ..., u^K] \in \mathbb{R}^{n^2TK} \) so that the decision variable for (4.7) is \( z = [x, u] \). To show that (4.7) is totally unimodular, it is necessary and sufficient to show that all extreme points of the constraint polyhedron are integer vectors. Recall that a point \( z^* = [x^*, u^*] \) is an extreme point if and only if the matrix of active constraints \( B(z^*) \) has rank \( n^2T(K+1) \), where the active constraint matrix is the matrix whose rows are the equality constraints and active inequality constraints of the problem at \( z^* \). Since \( z \in \mathbb{R}^{n^2T(K+1)} \), a point \( z^* \) is extreme if and only if \( B(z^*) \) has full column rank.

We can express the active constraints as:

\[
\begin{bmatrix}
A & 0 \\
C & D
\end{bmatrix}
\begin{bmatrix}
x^* \\
u^*
\end{bmatrix}
= \begin{bmatrix}
b \\
0
\end{bmatrix}
\]

where \( A, b_1 \) are chosen so that \( Ax^* = b \) is equivalent to the network flow constraints specified by (4.1), and \( C, D \) are chosen so that \( Cx^* + Du^* = 0 \) represents the active inequality constraints \( u_{ijt}^k = \lambda_{ijt}^k - x_{ijt} \), and/or \( u_{ijt}^k = 0 \). Noting that

\[
B(z^*)z = \begin{bmatrix}
A & 0 \\
C & D
\end{bmatrix}
\begin{bmatrix}
x \\
u
\end{bmatrix}
= \begin{bmatrix}
Ax \\
Cx + Du
\end{bmatrix}
\]

\( B(z^*) \) has full column rank only if \( A \) has full column rank, meaning \( x^* \) must be an extreme point of the polyhedral constraints defined by \( A \). However, recalling that \( A \) arises from network flow constraints and is unimodular matrix, this immediately implies that \( x^* \) must be an integer vector.

For each tuple \((i,j,t,k)\), the decision variable \( u_{ijt}^k \) is subject to exactly two constraints: \( u_{ijt}^k \geq \lambda_{ijt}^k - x_{ijt} \), and \( u_{ijt}^k \geq 0 \). In any extreme point, at least one of these constraints is active. This can be shown easily via contradiction. If \( z^* \) is extreme and for some \( u_{ijt}^k \), both (i.e. all) constraints are inactive, then define \( I \) to be the index of \( u_{ijt}^{k^*} \) in \( z^* \). Since there are no active constraints
involving $u^{k^*}_{ijt}$, the $I$th column of $B(z^*)$ is zero, hence $B(z^*)$ cannot have full column rank. Therefore $u^{k^*}_{ijt} \in \{0, \lambda^k_{ijt} - x_{ijt}\}$. Since we showed $x^*$ is integer, and $\{\lambda^k\}_{k=1}^K$ are integer, this implies that $u^*$ must be integer, finally implying that $z^*$ is integer. Since all extreme points are integer valued, (4.7) is a totally unimodular linear program.

Thus, in the setting where $w = 0$, the convex relaxation from (4.4) to (4.7) is tight in the sense that the solution to the latter is feasible and optimal for the former. For practical use the control strategy is to perform the tasks specified by the solutions to (4.6) and (4.7). The main strength of this approach is that both optimizers efficiently solve linear programs, as opposed to integer programs like (4.4) which can take orders of magnitude longer to solve in practice.

### 4.5 Stochastic optimization for Model Predictive Control of AMoD systems

When controlling an autonomous fleet of cars in real time, using a receding horizon framework allows the controller to take advantage of new information that is observed in the system. We propose a model predictive control approach to control AMoD systems online whereby a controller periodically issues commands obtained from solutions to optimization problems. Algorithm 2 outlines the details of the MPC controller for one timestep. Every $\Delta t$ minutes, the controller queries the system to obtain information about the current state $\{s_{it}\}_{i \in V, t \in T}$, the current number of waiting customers $\lambda_0$, and recent demand measurements $\rho$. The controller then draws $K$ samples from $\hat{P}(\lambda|\rho)$ and uses those samples to solve a stochastic optimization problem. The solve mode $\mathcal{I}$ specifies if a solution results from integer programming (cf. 4.3.1) or linear programming (cf section 4.4). Specifically, if $\mathcal{I} = 1$, the controller solves the integer program specified by (4.4), otherwise it solves the convex relaxation specified by (4.6) and (4.7). The controller executes the plan resulting from the optimization for the next $\Delta t$ seconds after which it repeats this process with updated information.

### 4.6 Numerical Experiments

In this section we evaluate the performance of Algorithm 2 in a MPC framework. We simulate the operation of an AMoD system servicing requests from the two different datasets and compare performance to recent state of the art algorithms. The AMoD system services trip requests in Hangzhou, China from a DiDi Chuxing ridesharing company dataset in the first experiment, and requests from the New York City Taxi and Limousine Commission dataset in the second experiment.
Algorithm 2: Model Predictive Control for AMoD systems using Stochastic Optimization

1. Stochastic AMoD Control \((I, s, \lambda_0, \rho)\);

2. Parameters: Road Network \(G = (V, E)\), Conditional generative demand model \(\hat{P}\);

   **Input**: Solve mode \(I\), System state \(\{s_{it}\}_{i \in V, t \in T}\), Waiting customers \(\lambda_0\), recent demand \(\rho\).

   **Output**: Control action \(x\).

3. Sample \(\{\lambda^k\}_{k=1}^K\) i.i.d. \(\sim \hat{P}(\lambda|\rho)\);

4. if \(I = 1\);
   
   5. Obtain \(\{x_{saa}(t)\}_{t \in T}\) by solving (4.4) with samples \(\{\lambda^k\}_{k=1}^K\);
   
   6. return \(x_{saa}(1)\);

7. else
   
   8. Obtain \(\{x_{bm}(t)\}_{t \in T}\) by solving (4.6) for waiting customers \(\lambda_0\);
   
   9. Obtain \(\{x_{saa}(t)\}_{t \in T}\) by solving (4.7) with samples \(\{\lambda^k\}_{k=1}^K\);

10. return \(x_{bm}(1), x_{saa}(1)\).

end

4.6.1 Scenarios

For Hangzhou, we leveraged a dataset provided by the Chinese ridesharing company Didi Chuxing. The dataset contains all trips requested by users from January 1 to January 21, 2016, resulting in a total of around eight million trips. The dataset separates Hangzhou into 793 discretized regions. However, the dataset contains only trips that started in a core subset consisting of 66 regions. For simplicity, we disregard trips that do not start and end in this core subset (approximately one million trips). For each trip, the records provide origin region, destination region, a unique customer ID, a unique driver ID, the start timestamp and the price paid. The dataset contains neither geographic information about the location of the individual districts, nor information on the duration of the trip. Thus, we used RideGuru \cite{RideGuru, 2017} to estimate the travel time of each trip from the trip price, which in turn allowed us to infer average travel times between regions. For the simulation, we used the first 15 days to train the forecasting model, and the last day to test in simulation by “playing back” the historical demand.

The second scenario is based on the well-known New York City Taxi and Limousine Commission dataset. It contains, among others, all yellow cab taxi trips from 2009 to 2018 in New York City. For each trip, the start and end coordinates and timestamps are provided. For our simulation, we looked only into the trips that started and ended in Manhattan. Additionally, we partitioned the island into 50 regions. We used the trips between December 22, 2011 and February 29, 2012 to train the forecasting model, and used the evening rush hour (18:00-20:00) of March 1, 2012 for testing in simulation.

\footnote{http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml}
4.6.2 Experimental Design

For each scenario, we simulate the operation of an AMoD system by “playing back” the historical demand at a 6 second resolution. That is, vehicle and customer states get updated in 6 second timesteps. If on arrival, a customer arrives to a region where there is an available vehicle, the customer is assigned to that vehicle. Otherwise, the customer joins the region’s customer queue. A customer’s trip duration corresponds to the travel time recorded in the dataset. However, vehicle speeds are such that travel time between any two region centroids corresponds to the average travel time between those respective regions in the dataset.

Every $\Delta t = 5$ minutes, the simulation invokes an AMoD controller. The controller returns the rebalancing tasks for each region. These tasks, in turn, are assigned to idle vehicles as they become available. After $\Delta t$ minutes, unused tasks are discarded, and the controller is invoked again. We tested the following controllers:

- **Reactive** is a time-invariant reactive controller presented in [Pavone et al., 2012] which rebalances vehicles in order to track uniform vehicle availability at all stations.

- **MPC-LSTM-MILP** is the model predictive controller presented in Chapter 3 which relies on point forecasts and mixed integer linear programming.

- **MPC-LSTM-LP** is a relaxation of the MPC-LSTM-MILP controller attained by the ideas described in Section 4.4 by running two linear programs.

- **MPC-LSTM-SAA** is the controller implementing Algorithm 2 with $I = 0$ and $K = 100$ samples.

- **MPC-Perfect** is a non-causal golden standard where the MPC controller is given perfect forecasts instead of samples of predicted demand.

All MPC controllers are using a planning horizon of 4 hours.

4.6.3 Forecasting

In these experiments, the generative model $\hat{P}$ for Algorithm 2 first estimates the mean of the future demand using a Long Short Term Memory (LSTM) neural network. The LSTM networks were trained on a subset of the data that does not include the test day. We trained a different network for each of the scenarios. Specifically, the LSTM takes as input the past 4 hours of observed customer demand and then predicts the expected demand for the next 2 hours. We assume that the demand follows a Poisson distribution. Moreover, to account for model uncertainty, we sample from the LSTM with dropout, a standard procedure to approximate Bayesian inference [Gal and Ghahramani, 2016]. Thus, we draw $K$ samples $\{\lambda^k\}_{k=1}^K$ from the LSTM using dropout, and then sample the demand predictions from a Poisson process whose mean is specified by $\bar{\lambda}$ so that $\lambda^k \sim \text{Poisson}(\bar{\lambda})$. 
4.6. NUMERICAL EXPERIMENTS

Figure 4.1: The top plot shows the number of waiting customers as a function of time for both our controller (MPC-LSTM-SAA) in blue and the controller from Chapter 3 in green. Under the operation of our controller, there are significantly fewer waiting customers throughout the day, which is reflected by the waiting times. Unexpectedly, as a price of this improved service, our controller issues more rebalancing requests, since it is planning for many outcomes, as shown by the middle plot. Looking at the middle and bottom plots together, we see that our controller does additional rebalancing precisely when there is significant variance in the demand.

4.6.4 Results

In the Hangzhou scenario, the MPC-LSTM-SAA controller, based on 2, greatly outperforms the other controllers: it provided a 62.7% reduction in mean customer waiting time over the MPC-LSTM-MILP controller from Chapter 3 and a 96.7% reduction from Reactive (see Table 4.1). Qualitatively, Figure 4.1 shows how the MPC-LSTM-SAA controller shows the greatest improvement over MPC-LSTM-MILP in times of the day where there is relatively high variance (in day-to-day travel demand variation). This suggests that the proposed algorithm’s rebalancing strategy is better at handling future demand with high uncertainty than prior work. Naturally, handling uncertainty requires being prepared for a large variety of demand realizations. Thus, it is not unexpected that, as seen in Figure 4.1 and Table 4.1, MPC-LSTM-SAA rebalances slightly more than MPC-LSTM-MILP; nonetheless, it still issues less rebalancing tasks than Reactive.
Moreover, the performance of the MPC-LSTM-MILP and MPC-LSTM-LP controllers are essentially the same, which suggests that the relaxations described in 4.4 yield reliable runtimes without significantly sacrificing performance quality.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>99 Percentile [s]</th>
<th>Reb. Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive</td>
<td>276.3</td>
<td>72.0</td>
<td>1890.0</td>
<td>139,927</td>
</tr>
<tr>
<td>MPC-LSTM-MILP</td>
<td>24.1</td>
<td>0.0</td>
<td>582.0</td>
<td>40,097</td>
</tr>
<tr>
<td>MPC-LSTM-LP</td>
<td>23.3</td>
<td>0.0</td>
<td>558.0</td>
<td>39,687</td>
</tr>
<tr>
<td>MPC-LSTM-SAA</td>
<td>9.0</td>
<td>0.0</td>
<td>264.0</td>
<td>68,150</td>
</tr>
<tr>
<td>MPC-Perfect</td>
<td>3.6</td>
<td>0.0</td>
<td>102.0</td>
<td>32,950</td>
</tr>
</tbody>
</table>

Table 4.1: Results summary for the DiDi scenario. Wait times are in seconds.

The New York City scenario also demonstrates benefits of using stochastic optimization in the control. Table 4.2 summarizes the results of this case study. While the 99 percentile wait time for the deterministic algorithm MPC-LSTM-LP is 32 percent smaller than that of Reactive, its mean waiting time is larger by 16 percent. Leveraging stochastic optimization, MPC-LSTM-SAA further improves the 99 percentile wait time of MPC-LSTM-LP by 17 percent and offers a 22 percent reduction in mean waiting time over Reactive. In summary, MPC-LSTM-SAA outperforms both Reactive and MPC-LSTM-LP in both mean and 99 percentile wait times. As a tradeoff, both MPC-LSTM-LP and MPC-LSTM-SAA issue more rebalancing tasks than Reactive.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Mean [s]</th>
<th>Median [s]</th>
<th>99 Percentile [s]</th>
<th>Reb. Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactive</td>
<td>19.2</td>
<td>0.0</td>
<td>732.0</td>
<td>7,196</td>
</tr>
<tr>
<td>MPC-LSTM-LP</td>
<td>22.7</td>
<td>0.0</td>
<td>504.0</td>
<td>7,907</td>
</tr>
<tr>
<td>MPC-LSTM-SAA</td>
<td>15.1</td>
<td>0.0</td>
<td>420.0</td>
<td>10,952</td>
</tr>
<tr>
<td>MPC-Perfect</td>
<td>10.8</td>
<td>0.0</td>
<td>384.0</td>
<td>8,356</td>
</tr>
</tbody>
</table>

Table 4.2: Results summary for the NYC scenario. Wait times are in seconds.

4.7 Conclusions

In this chapter, we developed a stochastic Model Predictive Control algorithm for AMoD systems that leverages uncertain travel demand forecasts. We discussed two variants of the proposed algorithm, one using integer programming and a relaxed, linear programming approach that trades optimality for scalability. Through experiments, we show that the latter algorithm outperforms state-of-the-art approaches in the presence of uncertainty.

Future work will incorporate traffic congestion by modeling a detailed road network with finite capacities as is done in [Zhang et al., 2016b]. This, in turn, can be coupled with models for public transit to provide a multi-modal [Salazar et al., 2018], real-time stochastic control of AMoD systems. Similarly, ongoing research is studying coordination between the power network and electric AMoD
4.7. CONCLUSIONS

Figure 4.2: The top plot shows the number of waiting customers as a function of time for several controllers. As expected, compared to the reactive controller, the predictive controllers have more customers wait at the beginning of the simulation in order to better prepare for the customers appearing later. Leveraging stochastic optimization, MPC-LSTM-SAA outperforms MPC-LSTM-LP and Reactive in terms of mean waiting time. As a tradeoff, the bottom plot shows that the reactive controller issues the least amount of rebalancing tasks, while MPC-LSTM-SAA issues the most.

systems [Rossi, 2018]. Such a setting would be particularly interesting since both travel and power demand are stochastic. Due to its central role in the proposed algorithm, another area of interest is the development of principled algorithms for predicting short-term demand. In particular, we will tackle the challenge of capturing spatiotemporal demand distribution in the face of a myriad of factors, such as weather, traffic and vehicle availability. Finally, accurate forecasts will enable robust and risk-averse objectives.
Chapter 5

Real world implementation: System Analysis and Control of Toyota Ha:mo

In this chapter, we take a deep dive into the operations of Toyota Ha:mo, a carsharing system in Toyota City, Japan. As discussed in the previous chapters, vehicle imbalance is a constant affliction of Mobility-on-Demand (MoD) systems. Unlike the models presented in the previous chapters, Toyota Ha:mo is not equipped with self-driving vehicles and therefore must employ humans to rebalance the vehicle stock. Nonetheless, we leverage the modeling framework of the previous chapters to develop a rebalancing control algorithm and build an end-to-end software system to coordinate human rebalancers in real-time. A limited pilot study shows promising results for the controller and encourages further development.

5.1 Introduction

The Toyota Ha:mo system is a one-way, station-based carsharing system with mostly single-capacity vehicles (there are some double-capacity vehicles, but they are a small fraction of the fleet). Ha:mo has currently several areas of service including Tokyo, Okinawa and Toyota City. In this work, we focused on the system deployed in Toyota City, and hereafter when we refer to the Ha:mo system we will be referring to the one deployed in Toyota City unless we specifically say otherwise.

As any Mobility-on-Demand (MoD, e.g., carsharing, ridesharing) system, the Toyota Ha:Mo system suffers from vehicle imbalance: vehicles tend to accumulate in certain stations, while other stations in the system become depleted of vehicles. To this end, full-time employees (referred to as rebalancers in the rest of this document) periodically rebalance empty vehicles to achieve better
availability across the different stations. When they are not rebalancing a vehicle, rebalancers travel between stations on foldable bicycles or with public transit. Currently, the decisions of when and where to reposition the vehicles are made by a human and are based on offline spreadsheet computations that specify how many vehicles are desired at each station at a given time of the day. The desired distribution is computed using a combination historical demand and human intuition.

In collaboration with the Toyota Research Institute (TRI) and Toyota Ha:Mo, we have adapted the algorithmic framework of the previous chapters, particularly that of Chapter 3, to non-autonomous systems where human drivers are needed to rebalance the vehicles, with the goal of deploying and testing them in Toyota City’s Ha:Mo system.

This chapter presents an end-to-end MoD controller that generates rebalancing tasks algorithmically in real-time. The structure of this chapter is as follows: Section 5.2 presents an analysis of the Ha:mo system as it is currently being operated with a special emphasis on the two key performance indicators (KPI): coverage and availability for vehicles and parking. Section 5.3 presents the MoD controller from the algorithmic and software engineering perspectives as well as results from a limited deployment. Finally, in Section 5.4 we present our recommendations for next steps as well as some concluding remarks.

5.2 System Analysis

The Ha:mo system consists of 62 active stations distributed throughout Toyota City (see Figure 5.1a) and 126 electric vehicles (see Figure 5.1b). To use a vehicle, a customer must first reserve a vehicle and a destination parking spot through the Ha:mo mobile application, then it can approach the vehicle and drive it. In general, there are no limits on how long the vehicle can be used: payment is based on time used.

As seen in previous chapters, the asymmetric travel demand of any one-way transportation system will cause vehicles to accumulate in some areas and while depleting in others. Bike and carsharing systems need to employ rebalancers to reposition these vehicles periodically. The following subsections characterize the travel demand in the Ha:mo system (Section 5.2.1), how it affects the system performance (Section 5.2.2), and what is currently being done to avoid imbalance problems (Section 5.2.3).

5.2.1 Demand Analysis

To understand the customer travel demand in the Ha:mo system we analyzed all the trip data for the year 2018. In 2018, there were 44,931 customer trips of which 94% took place during weekdays (see Figure 5.2a) and 74% occurred during the morning and evening rush hours (see Figure 5.2b). Thus, for the rest of the chapter we focus on the weekday demand, and in particular in three segments of the day: morning rush (orange in Figure 5.2b), midday (blue), and evening rush (red).
5.2. SYSTEM ANALYSIS

Station locations

(a) Station locations in Toyota City.  
(b) The single-capacity Ha:mo vehicle.

Figure 5.1: The Ha:mo system consists of 62 stations and 126 single-capacity vehicles.

Figure 5.2: 94% of trips occurred during weekdays, 74% occurred during rush hours (depicted in orange and red in the right figure).
In addition to the temporal aspect, one-way vehicle sharing systems are highly sensitive to the spatial variations. To understand how this plays a role, we computed the average number of trips that originate and end at each station for each segment of the day. As seen in Figure 5.3, in the morning rush most trips originate in the downtown of Toyota City (and some other peripheral stations) and most end in the area around Toyota Headquarters (HQ). The pattern is reversed for the evening rush: trips originate in Toyota HQ but end in Toyota City downtown. Thus, the morning and evening demand, as expected, is dominated by work commutes. In contrast, the demand through the midday is relatively symmetric: stations with relatively high departure rates have also relatively high arrival rates.

The imbalance in the rush hours is easier to appreciate when we compute the netflow: the average arrival rate minus the average departure rate. Figure 5.4 reflects the fact that imbalance is mostly aggravated during the rush hours, whereas the midday is relatively stable.
5.2. SYSTEM ANALYSIS

5.2.2 Stockout Analysis

The demand patterns tell one part of the story, the other part is told by the supply side, i.e. how many vehicles and parking spots are available and how does availability change. In particular, we care about two KPIs:

- **Availability**: The fraction of time that a particular station has either an available vehicle (vehicle availability) or an available parking spot (parking availability).

- **Coverage**: The fraction of stations that have at least one vehicle (vehicle coverage) or one parking spot (parking coverage).

Together they capture how supply behaves at each station and in the system as a whole. For example, Figure 5.5 shows the average vehicle availability vs average parking availability for each station. Notably, there is a correlation: stations with high vehicle availability tend to have high parking availability. However, in general, vehicle shortage is a bigger problem than parking shortage.

To understand how availability varies spatio-temporally, Figure 5.6 shows the vehicle and parking availability by segment of the day for each station. In the morning, vehicle are generally more scarce in the the stations that do not take part in the morning commute demand, while parking is scarce in the stations near the Toyota HQ. During the midday, vehicle scarcity aggravates in some of the downtown stations and remains a problem in the non-commute stations, while parking availability is only an issue in the stations near Toyota HQ. Finally, in the evening, the vehicle availability is overall very similar to the morning rush, but parking scarcity aggravates in the downtown area and improves in the area around Toyota HQ. In summary, vehicle availability is relatively similar throughout the day, while parking availability depends on the commute flow.

A similar picture is portrayed from the coverage perspective. Figure 5.7 shows a boxplots of the vehicle and parking coverages by hour. The vehicle coverage is relatively stable throughout day, with
Figure 5.5: Vehicle vs parking average availability by station.

Figure 5.6: Availability by segment of the day. *Top*: Vehicle availability. *Bottom*: Parking availability.
5.2. SYSTEM ANALYSIS

a small variance increase in during the day. In contrast, parking coverage *increases* during the day. Since vehicles always occupy a parking spot, even when in use, it is surprising to see that parking is more widely available during the day.

Naturally, the importance of availability should weighed with respect to how much demand there is at the given station. Figure 5.8a shows the relationship between yearly departing trips and vehicle availability. There is a clear correlation between availability and number of departing trips. There are two notable outliers: station 55 and 22. These two stations are located in the Toyota HQ and show very low vehicle availability relative to their yearly departures. This hints at a potential loss of customer demand, but, how much? Note that we are dealing with *censored demand*, that is, we only observe the demand whenever there is an available vehicle. A simple way to estimate how much demand is lost due to unavailability is to assume that the demand is fixed, and that whenever a customer wants to travel he or she reserves the vehicle if there is one, otherwise it uses some other transportation method. But this has no impact on later decisions. Using this assumption, the estimated loss of demand is

\[ d_{\text{censored}} = (1 - a) \frac{d_{\text{observed}}}{a}, \]

where \( d \) is the demand and \( a \) is the availability. Using this simple method and assuming average availabilities (a reasonable assumption given their stability throughout the day), we obtain the potential departing demand lost for each station and depict it in Figure 5.8b. Note that, once again, stations 55 and 22 show the greatest potential loss with a combined total of 3,000 trips (5% of the realized demand).

A similar study can be applied to parking availability. As shown in Figure 5.9a, the correlation between demand and availability is more tenuous. Moreover, station 7 is also in the “high demand, low supply” regime. This is in Figure 5.9b where the greatest estimated lost demand is for stations 7, 44 and 22, with a combined total of 2,200 estimated lost trips.
CHAPTER 5. REAL WORLD IMPLEMENTATION: TOYOTA HA:MO

(a) Yearly departures vs vehicle availability.  
(b) Potential censored departures by station.

Figure 5.8: Relationship between departures and vehicle availability.

(a) Yearly arrivals vs parking availability.  
(b) Potential censored arrivals by station.

Figure 5.9: Relationship between arrivals and parking availability.
5.2. SYSTEM ANALYSIS

There is a slight issue with considering each station’s availability independently. Several of the stations are within walking distance of each other, thus a potential customer can decide to use any of the nearby stations. To account for this, we must cluster the stations into “walkable clusters”, clusters of stations such that any pair of stations is within “walking distance”. To define what can be considered walking distance, we plot the frequency of trips that occur between stations of different walking travel times. Figure 5.10 shows that 99% of the trips occur between stations that are at least 331 seconds apart, thus we define “walking distance” as any pair of stations that are within 331 seconds apart.

![Figure 5.10: Fraction of Ha:mo trips for different walking distance buckets. The red line denotes the bottom 1% percentile.](image)

Using this definition, we created a walkability graph (see Figure 5.11a) that shows which pair of functions are within walking distance. We greedily selected the clusters with the largest combined demand to arrive to a set of walkability clusters (see Figure 5.11b). The hypothesis is that these clusters behave more like “super-stations”.

We are now in a position to repeat the estimates on censored demand, but now using the clusters as a stations. Figures 5.12b and 5.13b show the resulting study. In general, estimated censored demand due to vehicle unavailability is similar: around 3,000 trips for the cluster containing stations 22, 49 and 55. However, censored demand due to parking unavailability is greatly changed: stations 7, 44, and 22 which where showing the greatest loss prior to clustering, are now showing significantly less estimate censored demand. Station 2014 is now estimated to have the largest loss of demand due to parking unavailability.

This analysis show that overall vehicle availability is a bigger issue than parking availability. Moreover, the biggest challenge is in stations in the Toyota HQ area. This despite the fact that Ha:mo employs rebalancers whose main rebalancing effort, as we will see next, is to keep vehicle and parking availability high in that area.
CHAPTER 5. REAL WORLD IMPLEMENTATION: TOYOTA HA:MO

Walkability graph

Walkability clusters

(a) Walkability graph.
(b) Walkable clusters.

Figure 5.11: Walkability graph and clusters.

(a) Yearly departures vs vehicle availability.
(b) Potential censored departures by cluster.

Figure 5.12: Relationship between departures and vehicle availability using walkability clusters.
5.2. SYSTEM ANALYSIS

5.2.3 Existing Rebalancing Efforts

Currently, the Ha:mo system in Toyota City is operated by a team of operators (“Takumi”) whose duties are primarily to (1) periodically rebalance the fleet of Ha:mo vehicles in order to maintain vehicle demand and supply in balance across the city, and (2) provide periodic maintenance to the Ha:Mo vehicles. For rebalancing, in the current system the Takumi operators query the state of the system (e.g., number of available vehicles at a station) through an app, and based on prior knowledge and heuristic insights decide what to do next (e.g., whether to move a vehicle to a station depleted of vehicles), see Figure 5.14.

As Figure 5.15 shows, their weekday rebalancing efforts are largely concentrated on the morning and evening rushes. Takumi kindly provided a schematic of their typical morning and evening rebalancing tactics and are shown in Figure 5.16. The overall strategy is to move vehicles out of stations 55 and 22 in the morning, and into 55 and 22 in the evening.

These strategies are further confirmed by data: Figure 5.17 shows a similar “empty station 55/22” strategy in the morning, and “fill station 55/22” in the evening, although the evening tasks show
Figure 5.15: Average number of rebalancing trips per hour.

(a) Morning rush.  
(b) Evening rush.

Figure 5.16: Typical rebalancing strategies.
more diversity. The current rebalancing strategy during the midday is almost non-existent: Takumi spends this time mostly for maintenance and manning the customer kiosk.

Figure 5.17: Main rebalancing effort per segment.

5.2.4 Discussion

We now proceed to discuss some high-level observations on the system. The Ha:mo system in Toyota City has been functioning for several years, and its now-mature user base has come to depend on Ha:mo primarily for corridor-like operations between the Toyota City train station in the downtown and Toyota HQ. Over time, Ha:mo staff has focused its efforts on accommodating this demand and devotes the vast majority of its time during morning and afternoon peak travel periods servicing stations at Toyota HQ. Not captured by the previous censored demand analysis, we hypothesize that the effort of Ha:mo staff to exclusively service this demand corridor has led to a cyclical pattern in which high quality of service leads to higher user demand, to increased rebalancer effort, and in turn higher quality of service. Similarly, low service quality leads to low demand, which leads to limited rebalancer investment and low service quality – accordingly, the attrition in the poorly serviced stations is significant.

5.3 System Control

With the objectives of optimizing the performance of Ha:mo systems and allowing their operation with large-scale fleets (e.g. with more than 1,000 vehicles, which would make the current “manual” operation very challenging), we developed an optimization-based MoD controller, which comprises (1) an optimization-based controller, which, based on demand predictions, automatically computes optimal tasks for the Takumi operators, and (2) an app user interface that presents such actions to the operators in an intuitive way. This section describes the MoD controller. In Sections 5.3.1
and 5.3.2, we formalize the rebalancing problem and propose a model predictive control algorithm to address it. Then, in Section 5.3.3, we describe the pilot program and its results.

5.3.1 Problem formulation

We begin by modeling the system in the same framework as in Chapters 3 and 4 but with the additional nuances that are inherent to non-autonomous mobility systems. Consider an MoD system with a set of stations \( \mathcal{N} \) and with \( m \) vehicles. Each station \( i \) has \( c_i \) parking spots such that \( \sum_{i \in \mathcal{N}} c_i \geq m \). Let \( G = (\mathcal{V}, \mathcal{E}) \) be a directed graph representing a time-augmented model of the MoD system. We consider a time horizon of \( T \) time steps, and let \( T = \{1, \ldots, T\} \) represent their sequence.

The set of nodes \( \mathcal{V} \) contains each station at a given time step, i.e. \( \mathcal{V} := \{(i, t) : i \in \mathcal{N}, t \in \mathcal{T}\} \), such that the tuple \((i, t)\) represents station \( i \) at time step \( t \) and. Let \( \tau_{i,j}^v \) and \( \tau_{i,j}^d \) be the required number of time steps to traverse from \( i \) to \( j \) for the passenger and rebalancing vehicles, and the drivers, respectively. Then, the edge set \( \mathcal{E} \) represents the possible inter-station travels within \( T \) time steps, and can be divided into \( \mathcal{E}_v \), the set of edges possible only for the vehicle flows, and \( \mathcal{E}_d \), the edges possible only for the driver flows. Thus, there is a vehicle edge \((i, t, j, t + \tau_{i,j}^v) \in \mathcal{E}_v \) connecting stations \( i \) and \( j \) if the vehicle travel time \( \tau_{i,j}^v \leq T - t \). The same holds for the driver edges. Finally, to represent the ability to remain in a station, there exist vehicle, \((i, t, i, t + 1) \in \mathcal{E}_v \), and driver, \((i, t, i, t + 1) \in \mathcal{E}_d \), edges that connect a station \( i \) at time \( t \) with the same station at the next time step \( t + 1 \). Note that, given the origin station \( i \), the destination station \( j \), and departure time \( t \), it is possible to infer the arrival time \( t' \). Thus, for the remainder of this exposition we subscript edges by the origin, destination, departing time tuple \((i, j, t)\).

Passenger demands are represented by the set \( \Lambda := \{\lambda_{i,j,t}\}_{i,j,t} \) such that there are \( \lambda_{i,j,t} \) customers that want to travel from station \( i \) to station \( j \) at time step \( t \). We assume customers take the shortest route, i.e. we do not model congestion or non-direct travel. We denote the passenger-carrying vehicles departing \( i \) at time \( t \) with destination \( j \) as \( x_{i,j,t}^p = \lambda_{i,j,t} \).

In order to keep adequate vehicle and parking availability, the system operator manages a team of rebalancers whose job is to reposition the vehicles. Let \( x_{i,j,t}^r \) and \( x_{i,j,t}^d \) be the vehicle rebalancing and driver flows, respectively. Additionally, denote as \( g_v : \mathcal{V} \to \mathcal{P}(\mathcal{V}) \) and \( g_d : \mathcal{V} \to \mathcal{P}(\mathcal{V}) \) (where \( \mathcal{P}(\cdot) \) denotes the power set), a pair a functions that given a node \((i, t) \in \mathcal{V} \) return the set of reachable nodes by vehicles and drivers, i.e. \( g_v(i, t) = \{(j, t') : (i, t, j, t') \in \mathcal{E}_v\} \). Similarly, \( h_v : \mathcal{V} \to \mathcal{P}(\mathcal{V}) \) and \( h_d : \mathcal{V} \to \mathcal{P}(\mathcal{V}) \) are functions that given a node \((i, t) \) return the set parent nodes by vehicles and drivers, i.e. \( h_v(i, t) = \{(j, t') : (j, t', i, t) \in \mathcal{E}_v\} \). For compactness, we let \( \mathcal{X} := \{x_{i,j,t}^r\}_{i,j,t} \cup \{x_{i,j,t}^v\}_{i,j,t} \cup \{x_{i,j,t}^d\}_{i,j,t} \) represent the set of decision variables.

We are now in a position to describe the operational constraints and objectives of the MoD system operator.
5.3. SYSTEM CONTROL

The *time varying conservation of vehicles* is preserved by the following constraint:

\[
\begin{align*}
& a_{i,t} + \sum_{(j,t') \in \mathcal{h}_v(i,t)} x^e_{j,i,t'} + x^p_{j,i,t'} = \sum_{(j,t') \in \mathcal{g}_v(i,t)} x^e_{i,j,t'} + x^p_{i,j,t'}, \quad \forall (i,t) \in \mathcal{V},
\end{align*}
\]

where the \( a_{i,t} \) represents the number of vehicles that are added to the system (e.g. at the beginning of the time horizon, \( t = 0 \), we use \( a_{i,t} \) as the number of vehicles available at each station).

Similarly, we preserve the *time variant conservation of drivers* by the following constraint:

\[
\begin{align*}
& b_{i,t} + \sum_{(j,t') \in \mathcal{h}_d(i,t), j \neq i} x^e_{j,i,t'} + \sum_{(j,t') \in \mathcal{g}_d(i,t)} x^d_{j,i,t'} = \\
& \sum_{(j,t') \in \mathcal{g}_d(i,t), j \neq i} x^e_{i,j,t'} + \sum_{(j,t') \in \mathcal{g}_d(i,t)} x^d_{i,j,t'}, \quad \forall (i,t) \in \mathcal{V},
\end{align*}
\]

where the \( b_{i,t} \) represents the number of drivers that are added to the system. Note that in this conservation constraint the flows of empty vehicles that remain at a station are not taken into account: a vehicle does not need a driver to remain idle.

In addition to vehicle and driver conservation, the system operator must ensure that parking constraints are satisfied. In particular to the Ha:mo system (as described previously), vehicles always occupy a parking spot whether at their current station if they are available or at their destination station if they are in use. Thus, *parking feasibility* is preserved by:

\[
\sum_{t' \geq t} a_{i,t'} + \sum_{(j,t') \in \mathcal{h}_v(i,t')} x^e_{j,i,t'} + x^p_{j,i,t'} \leq c_i, \quad \forall (i,t) \in \mathcal{V}.
\]

The goal of the system operator is to i) minimize the demand lost due to parking and vehicle unavailability, ii) minimize its operational costs. The *demand lost cost* is defined as:

\[
J_\lambda(\mathcal{X}, \Lambda) = \sum_{(i,j,t,t') \in \mathcal{E}} \hat{w}^p_{i,j,t} \max(\lambda_{i,j,t} - x^p_{i,j,t}, 0) + \\
\sum_{(i,t) \in \mathcal{V}} \hat{w}^{p}_{i,t} \max(a_{i,t} + \sum_{(j,t') \in \mathcal{h}_v(i,t)} x^e_{j,i,t'} + x^p_{j,i,t'} - \sum_{j \in \mathcal{N}} \lambda_{i,j,t}, 0) + \\
\sum_{(i,t) \in \mathcal{V}} \hat{w}^{p}_{i,t} \max(\sum_{t' \geq t} a_{i,t'} + \sum_{(j,t') \in \mathcal{h}_v(i,t')} x^e_{j,i,t'} + x^p_{j,i,t'} + \lambda_{j,i,t'}, 0 - c_i, 0),
\]

where \( \{w^p_{i,j,t}\}_{i,j,t}, \{\hat{w}^p_{i,t}\}_{i,t}, \) and \( \{\hat{w}^{p}_{i,t}\}_{i,t} \) are the weighted cost of lost demand, vehicle unavailability, and parking unavailability. Note that the second and third term penalize the mismatch between the supply and demand for vehicles and parking. This is a parameterization choice to account for systems whose demand is *sparse* such that the first term is rarely larger than 1. The *operational
cost is defined as:

$$J_x(\mathcal{X}, \Lambda) = \sum_{(i,j,t,t') \in E} w_{i,j,t}^r x_{i,j,t}^r + w_{i,j,t}^d x_{i,j,t}^d.$$  \hspace{1cm} (5.5)

Thus, the operator’s objective is

$$\min_{\mathcal{X}} \quad J_{x}(\mathcal{X}, \Lambda) + J_{x}(\mathcal{X}, \Lambda),$$  \hspace{1cm} (5.6a)

subject to

$$\text{(5.1), (5.2), (5.3).}$$  \hspace{1cm} (5.6b)

### 5.3.2 Model predictive control

Equipped with the optimization problem from the last section, we now proceed to cast the control problem within the model predictive control framework as postulated in Chapter 3 which leverages predictions of future demand (see Figure 5.18). We begin by outlining the overall algorithm and then we delve into the details of each subcomponent.

**Algorithm**

The proposed algorithm, summarized in Algorithm 5, is as follows: at a given time $t_0$ we first observe the system state to capture the vehicle and driver locations and state, $A := \{a_{i,t}\}_{i,t}$ and $B := \{b_{i,t}\}_{i,t}$ respectively. We then proceed to predict future customer demand $\Lambda$ for the next $T$ time steps. Using this information, we compute the optimal rebalancing strategy $\mathcal{X}^r$ by solving the mixed-integer linear program (5.6). Finally, we assign the rebalancing tasks corresponding to the first time interval to available vehicles as they become available. After a period $\Delta t$, i.e. at $t_0 + 1$, we recompute the rebalancing strategy using Algorithm 5. Thus, this process is repeated during the entire operation of the system.

![Figure 5.18: MPC framework.](image)
Algorithm 3: Model Predictive Control

1: procedure MPC
2: \[ S \leftarrow \text{count idle vehicles and estimate trip arrivals} \]
3: \[ \lambda_{ij0} \leftarrow \text{count outstanding customers} \]
4: \[ \hat{\Lambda}_{t_0, t_0 + T_{\text{forward}}} \leftarrow f(\theta_t) \]
5: \[ X^p, X^r, W, D \leftarrow \text{solve Problem (3.5)} \]
6: Assign \( \{x_{ij1}\}_{ij1} \) to available vehicles

State observation  
At the beginning of each iteration of Algorithm 3, the first step is to capture the current system state in terms of vehicle availabilities and outstanding passengers.

Vehicles are considered available either when they are idling or after they complete a trip. Thus, at the start of the optimization process, let \( s_i \) be the current number of idling vehicles at region \( i \in \mathcal{N} \). Additionally, let \( v_{i,t} \) be the number of vehicles traveling to region \( j \in \mathcal{N} \) expected to arrive at time interval \( t \). Then, the vehicle starting locations for the planning horizon are

\[
a_{i,t} = \begin{cases} 
  s_i + v_{i,t}, & \text{if } t = 1, \\
  v_{i,t}, & \text{otherwise} 
\end{cases} \quad \forall (i, t) \in \mathcal{V}. \tag{5.7}
\]

Drivers are accounted in a similar fashion. Let \( z_i \) be the current number of idling drivers at region \( i \in \mathcal{N} \), and \( u_{i,t} \) the number of drivers traveling to region \( j \in \mathcal{N} \) expected to arrive at time interval \( t \). Then, the driver starting locations for the planning horizon are

\[
b_{i,t} = \begin{cases} 
  z_i + u_{i,t}, & \text{if } t = 1, \\
  u_{i,t}, & \text{otherwise} 
\end{cases} \quad \forall (i, t) \in \mathcal{V}. \tag{5.8}
\]

Forecasting  
The second part of Algorithm 3 consists of predicting future customer demand. Let \( f \) be a forecasting model trained with historical data, \( \phi_t \) a diverse set of features relevant to the model available at the time of prediction (e.g. current traffic conditions, weather, recent travel demand, etc.), and \( T \) the forecasting horizon. Then, we estimate the travel demand by \( \Lambda = f(\phi_t) \).

Controller  
The third step computes the rebalancing strategy for the planning horizon \( T \) using the observed state and the predicted demand. With the state variables, \( \mathcal{A} \) and \( \mathcal{B} \), and the forecasted demand \( \Lambda \), rebalancing strategy \( X^* \) is given by solving (5.6).

5.3.3 Pilot description
In order to faithfully capture not only the issues faced by the system in general but also the rebalancers during their work shift, we developed the control system via an incremental schedule consisting of three phases (see Figure 5.19):
• **Phase 1.** During this phase, we developed the control algorithm such that it can capture the relevant nuances of operating the Ha:mo system.

• **Phase 2.** During this phase we developed the backend infrastructure required to deploy the controller and test the controller outputs in a supervised manner. Instead of directly instructing rebalancing tasks to the rebalancers, the controller suggested these tasks to the operator in the form of “recommended actions.” This phase provided valuable insights to revisit the control algorithm and guide the required features for the fully functional controller.

• **Phase 3.** During this phase, we developed the software necessary to allow the controller to directly instruct the rebalancers which actions they should take. The development of the mobile interface as well as the refinement of both the backend and the control algorithm relied heavily on feedback from both TMC and Takumi. This final phase included a week-long user study with Takumi concluding on a two day pilot test.
Software architecture

The developed control system consists of three main components (as seen in Figure 5.20): the backend, the Operator User Interface (OUI), and the Mobile User Interface (MUI).

**Backend** The backend is the core of the control system. Its responsibility comprises of i) managing and updating the state of the system by periodically calling the API exposed by the Ha:mo web infrastructure to obtain the station state and by interacting with the MUI to update the state of the drivers, ii) expose an API for the OUI and MUI to interact with the system, and iii) periodically recompute the rebalancing strategies by executing Algorithm 3. The web service is built on Python with a Postgres database, but the core optimization problem (Problem (5.6)) is solved using a MATLAB subprocess with a CPLEX MILP solver.

**OUI** The Operator User Interface provides the necessary features for i) reviewing current and past strategies (as displayed in Figure 5.21), ii) reviewing the current state of the system, iii) user management, iv) run what-if-scenarios based on different driver states. The OUI is a web application built on React.
CHAPTER 5. REAL WORLD IMPLEMENTATION: TOYOTA HA:MO

Figure 5.21: Screenshot of the OUI.

MUI  The Mobile User Interface is the mobile application that rebalancers use to automatically communicate their location, start/end tasks, and requests new tasks. As seen in Figure ??, the emphasis of the application is in providing an intuitive interface for rebalancers to receive new tasks. In particular, the rebalancers are given instructions to (1) walk to a nearby MoD vehicle, (2) drive the vehicle to a station with higher anticipated demand, or (3) wait at the location for further instructions. The app interface permits feedback from Ha:mo and TMC employees, and is available in both English and Japanese versions.

Results

We ran a limited study where a rebalancer used the MUI from 13:00-14:30 on 12/17/2019 to sample the end-to-end controller performance. During this time, the rebalancer completed 4 different tasks issued by the MoD controller.

As seen in Figure 5.22, the tasks involved the movement of vehicles between the Toyota City downtown and Toyota HQ areas, as well as supplying a station that was removed from the two core areas. This is in contrast to usual Midday rebalancing tasks which, as seen in Figure 5.17, tend to limit themselves to the confines of the two core areas.

While the limited time span of the study prohibits robust statistical analyses of the controller on average availability, we can investigate the impact of the controller by observing the impact it had on availability and coverage. Of the four tasks, three were involved moving vehicles to empty stations and two involved moving vehicles out of full parking. One of the tasks involved both, while one of the tasks was mainly a way to move the rebalancer from Toyota downtown to Toyota HQ. In Figure 5.23a, we see the vehicle coverage for the length of the study period. Note that this is a
relatively high coverage (79% at the beginning of the period) for this time of the day (compare with Figure 5.7a), however, the main contribution is a net increase in coverage of 4.8 percentage points. This is a significant change in coverage when compared to the same time periods for all weekdays year-to-date (see Figure 5.23b). A similar improvement of 3.2 basis points can be appreciated on parking coverage (the net change during the time period is 10 basis points, but only 3.2 of those are due to rebalancing, see Figure 5.24a and 5.24b).

5.3.4 Discussion

We built an MoD controller that has seen limited testing on the field. In the process, we have gained valuable operational insights, most of which has been used to refine the MoD controller. In this section, we reflect on some of these insights.

First, the most valuable lesson learned in this process has been that there is a wealth of “verbal” knowledge and constraints that is never captured in paper. It was paramount to shadow Takumi while they operated as normal and while using the MoD Controller to be able to infuse a fraction of this knowledge into the controller. For example, some stations are used only as “overflow” stations, while others, due to contractual obligations with nearby companies, are required to have at least a certain number of vehicles available at all times. Neither of these requirements are written anywhere, and, thus, are difficult to capture systematically.

Second, since we rely heavily on the rebalancers for both executing the tasks and providing
Figure 5.23: Left: Vehicle coverage during the pilot. The red vertical lines denote when the different rebalancing tasks ended. Right: Vehicle coverage change over the same time period for all weekdays year-to-date. The red vertical line denotes the change observed during the pilot.

Figure 5.24: Left: Parking coverage during the pilot. The red vertical lines denote when the different rebalancing tasks ended. Right: Parking coverage change over the same time period for all weekdays year-to-date. The red vertical line denotes the change observed during the pilot.
feedback on them, it is important that these tasks are interpretable, i.e. that the rebalancer can understand what the motive is behind the given task. In the current iteration of the MoD controller, tasks provide little to no information as to “why” their being issued, thus making it difficult to receive proper feedback from the rebalancer and diminishing his or her trust on the application.

Third, there was a potential that under the MoD controller an operator could be left stranded at a station far away from Toyota HQ. This has been addressed by an optional constraint to Problem (5.6) enforcing that operators at the end of the planning horizon will have a task bringing them back to Toyota HQ.

Fourth, the current version of the MoD controller does not account for the vehicles’ state of charge. This might lead to vehicles with low charge being relocated to stations without chargers. Future iterations of the MoD controller should address this issue.

Finally, Takumi has found (and concurrently induced) a niche demand between Toyota HQ and Toyota City train station, and optimized fully for this demand pattern. Arguably, this is at the expense of service to the rest of the stations. The MoD controller is unlikely to improve performance for this niche demand, but appears well suited to improve the level of service for the rest of the demand and, with time, mitigate user attrition and increase service adoption at stations other than Toyota HQ / Toyota City train station. An MoD controller equipped with a cost function weighted such that it prioritizes stations with high potential censored demand (as seen in 5.2.2) could be a good next step for inducing demand in untapped stations.

5.4 Conclusion

We have provided a demand and supply study of the Ha:mo system and developed an MoD controller that is able to be used by multiple rebalancers in the field. The initial pilot shows promising results, but more work is required in order to make it fully operational.

Specifically, the following are some new directions of work. First, enable the MoD controller to account for the vehicle state of charge. Second, there is ongoing work on including the operation of private vehicles into the optimization, and valuable gains have been made, however, further improvements in optimization techniques are required. Third, the method used to estimate lost demand relies on strong assumptions, namely, availability does not affect demand. Future work should look into probabilistic methods to infer latent demand. Fourth, we should automate the reservation of vehicles as the rebalancer accepts a task. Currently, it is necessary to do it in the customer Ha:mo application. Fifth, as mentioned before, further work should be devoted into making tasks more explainable. Sixth, we would like to carry out a multi-week testing plan to be able to perform statistical analyses on the performance of the MoD controller. Finally, we recommend investigating how autonomous vehicles could help with the task of rebalancing, and planning for autonomous MoD pilots. In particular, the co-design of autonomous vehicles along with which
routes should be autonomous is an interesting venue of research.
Chapter 6

Conclusions

The forthcoming years will likely see the first large-scale deployments of AMoD systems. Given the potentially outsized impact that these systems will have on our urban environments, it is paramount that we develop the proper methodology for analyzing and controlling these systems.

The frameworks presented in this thesis collectively provide a systematic approach for the analysis, design, control, and deployment of AMoD systems. By traversing from the theoretical realm to a real-world implementation, we expose the nuanced concerns that AMoD systems are subject to and how to address them in the presence of uncertainty.

The numerical experiments based on real-world data from New York City and the city of Hangzhou highlight the impact that effective coordination of self-driving vehicles can have on the overall system performance. And the carsharing implementation in Toyota Ha:mo shows that the methods presented in this thesis have promising use cases even in the absence of self-driving vehicles.

We conclude this thesis by summarizing the work herein presented and postulating several new promising directions of research.

6.1 Summary

In this thesis, we present i) a queueing-theoretical framework for modeling AMoD systems, and ii) a stochastic model predictive control framework for controlling AMoD systems.

Specifically, in Chapter 2 we show how to cast AMoD systems as BCMP queueing networks while capturing stochastic passenger arrivals, vehicle routing on a road network, congestion effects, and battery charging-discharging for electric vehicles. We show how to synthesize routing, rebalancing and charging policies and how to analyze a variety of performance metrics for a given policy. The merits of this framework is showcased in a case study of New York City wherein we present how stakeholders can leverage existing data to make structural decisions of the AMoD system.

Chapters 3 and 4 present the model predictive control framework. First, in Chapter 3, we propose
a scalable MPC algorithm for operating the system in real-time by leveraging short-term forecasts of customer demand. A simulation-based case study shows that the MPC algorithm outperforms a state-of-the-art algorithm with a 91.3% reduction in mean customer wait time. Then, in Chapter 4, we extend the MPC algorithm to capture the uncertainty of demand forecasts. This further improves the performance of the control algorithm by making it more robust to deviations from the expected demand. In simulation, the stochastic MPC algorithm further reduces the mean waiting time by 62.7% when compared to the non-stochastic version.

Encouraged by the results observed in the numerical experiments of Chapters 3 and 4, in Chapter 5 we implement the MPC algorithm in the real-world. In a collaboration with Toyota, we designed and built a software system to coordinate in real-time the rebalancing efforts of the carsharing system Toyota Ha:mo. A limited pilot showed promising results prompting further development.

6.2 Future Directions

In this section, we would like to discuss some interesting avenues of research that were not discussed within the scope of the individual chapters, but are still enabled by the work herein presented.

Vehicle and system co-design The design of autonomous vehicles is influenced by the operational domain in which they will be used. For example, restricting the operational domain to low speed limits might influence the level of safety guarantees required by the vehicle, and thus its cost. Similarly, the operational design of an AMoD system depends on the capabilities of the vehicles comprising its fleet. A fleet of vehicles limited to low speed limit areas might require a very different rebalancing policy than that of a unrestricted fleet. Thus, an avenue of research would explore the co-design of AVs and the AMoD systems in which they operate. Broadly, the idea would be to leverage co-design optimization frameworks, such as [Censi, 2015], along with autonomous vehicle design models and the AMoD modeling framework presented in Chapter 2 to find optimal designs of AV features and AMoD operations. A first step in this direction can be found in [Zardini et al., 2020].

Social equity A common concern with new technologies is how its adoption impacts social equity. Future research could leverage the queueing theoretical models presented in this thesis to study how different populations might observe different levels of service. For example, wheelchair users require larger vehicles and potentially additional help to board the vehicles. How would an AMoD operator plan for ensuring acceptable level of service? How can public authorities frame regulations such that they provide the right incentives for acceptable service for wheelchair users?

AMoD marketplaces Throughout this thesis we only considered scenarios where there is a single AMoD system operating in the service area. However, in the future we will likely see several AMoD operators competing in a given service area. How do we characterize the economic equilibrium of
6.2. FUTURE DIRECTIONS

such competitions? How should AMoD operators change their rebalancing policies in the presence of competition? Additionally, this avenue of research could introduce a new lever for the AMoD operator, that of pricing. The combination of rebalancing and pricing strategies in a purely competitive setup might settle in an equilibrium that is suboptimal from a social-welfare point of view. How can regulators steer this equilibrium towards a socially optimum point? We believe that this area of research could be explored by combining the queueing theoretical framework of Chapter 2 with economic theory.

**Mixed fleets** As self-driving technology matures, there will be a period of time during which human driven MoD systems might co-exist with AMoD systems in the same service area. How should an operator in charge of a mixed human and autonomous fleet model this hybrid system? What strategies should operators take if the MoD and the AMoD systems are, in fact, competitors?

**Importance sampling from forecasting models** In Chapter 4 we relied on the assumption of the access to a generative model for sampling the scenarios needed for the optimization problem. However, as we showed, the number of sampled scenarios poses a trade-off between the quality of the solution and the complexity of the optimization problem. Future research should investigate strategies for sampling the generative model under limited budgets. One approach could be devising forecasting generative models that are amenable to adaptive importance sampling algorithms (such as in [Ryu, 2016]).

**Human mobility forecasting and control** Within the scope of this thesis, we assumed we had no control over the travel demand. However, in the near future it might be possible to influence human mobility patterns, for example, via the routes offered by route planning applications such as Google Maps. In such cases, it might be possible to nudge travelers to take different modes of transportation such that the overall transportation ecosystem is optimized. Future research should study how to optimally route travel requests across multiple modes of transportation, as proposed in [Salazar et al., 2019], while being able to forecast not only travel requests but also how individual travelers might react to different modal combinations.

**Ridesharing-aware rebalancing** In this thesis we did not consider ridesharing (in the sense of multiple travel requests sharing a single vehicle). This type of rides are increasingly popular and hold promise to reduce the number of vehicle-miles traveled. However, the ability to pool different travel requests together depends highly on the state of supply at the time of making the match. Thus, it is important that future rebalancing algorithms are able to model ridesharing capabilities. In [Tsao et al., 2019], a first step in this direction, the authors present a model predictive controller that accounts for ridesharing with double capacity vehicles. Future work in this direction should consider multi-capacity vehicles.
Real-world implementation in ridesharing context  Finally, future research should study large scale deployments of rebalancing algorithms such as the ones presented in this thesis. A potential approach could be within the context of ridesharing but with fully compliant drivers (rather than simply incentive driven). In this setup, the drivers would be tasked to follow the tasks of the ridesharing application even in the absence of passengers. Such a setup might need to consider drivers that are fully employed, so a particular experiment might choose to have only a subset of the fleet to be fully compliant. An ancillary interesting aspect of such an experiment would be to study whether a combined fleet of fully-compliant and incentive driven drivers can perform better (from the operator’s point of view, such as profit-maximization) than a purely incentive driven fleet. In this sense, this avenue of research is very similar to the mixed fleet proposed earlier.


