MODELS AND LARGE-SCALE COORDINATION ALGORITHMS FOR AUTONOMOUS MOBILITY-ON-DEMAND

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

Rick Zhang
June 2016

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

(Marco Pavone) Principal 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.

[illegible]

(Stephen Rock)

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.

[illegible]

(Mac Schwager)

Approved for the Stanford University Committee on Graduate Studies

Abstract

Urban mobility in the 21st century faces significant challenges, as the unsustainable trends of urban population growth, congestion, pollution, and low vehicle utilization worsen in large cities around the world. As autonomous vehicle technology draws closer to realization, a solution is beginning to emerge in the form of autonomous mobility-on-demand (AMoD), whereby fleets of self-driving vehicles transport customers within an urban environment. This dissertation introduces a systematic approach to the design, control, and evaluation of these systems. In the first part of the dissertation, a stochastic queueing-theoretical model of AMoD is developed, which allows both the *analysis* of quality-of-service metrics as well as the *synthesis* of control policies. This model is then extended to one-way car sharing systems, or human-driven mobility-on-demand (MoD) systems. Based on these models, closed-loop control algorithms are designed to efficiently route empty (rebalancing) vehicles in very large systems with thousands of vehicles. The performance of the algorithms and the potential societal benefits of AMoD and MoD are evaluated through case studies of New York City and Singapore using real-world data.

In the second part of the dissertation, additional structural and operational constraints are considered for AMoD systems. First, the impact of AMoD on traffic congestion with respect to the underlying structural properties of the road network is analyzed using a network flow model. In particular, it is shown that empty rebalancing vehicles in AMoD systems will *not* increase congestion, in stark contrast to popular belief. Finally, the control of AMoD systems with additional operational constraints is studied under a model predictive control framework, with a focus on range and charging constraints of electric vehicles.

The technical approach developed in this dissertation allows us to evaluate the societal benefits of AMoD systems as well as lays the foundation for the design and control of future urban transportation networks.

To my parents, Liping and Baochun

Acknowledgments

The past five years at Stanford has been an incredible journey for me professionally and personally. I am extremely fortunate to have been able to pursue my research interests in robotics and in autonomous cars, to conduct cutting edge research in a blossoming field, and to contribute knowledge that has the potential to have a real impact on society.

First and foremost, I would like to thank my advisor Prof. Marco Pavone for his guidance throughout my PhD. He taught me many invaluable skills that I will carry forward throughout my career, including the ability to communicate effectively through technical writing, always being aware of research trends so as to stay on the cutting edge, and the ability to assimilate techniques from different fields to push the boundaries of research. Marco is an exceptional teacher, a great mentor, and a supportive friend. Throughout my PhD career, Marco has challenged me to be the best researcher I can be and I look forward to continuing to work together in the years to come.

I would also like to thank my reading committee members, Prof. Steve Rock and Prof. Mac Schwager. In the first two years of my graduate career, Prof. Rock was an incredible mentor who taught me how to conduct research. In the final months of my PhD, Prof. Rock gave valuable advice for my career as well as asked challenging questions that strengthened and advanced my research. Through our conversations early on in my PhD career, Prof. Schwager gave me valuable input that would help shape my research direction and my thesis.

I was lucky to have a great defense committee, who provided a lot of interesting and insightful feedback. I would like to thank Prof. Mykel Kochenderfer, whose interest in autonomous vehicles parallels my own. Last but certainly not least, I would like to thank my committee chair Prof. Ilya Strebulaev, who offered interesting insights into my research from a financial perspective, something I have previously not thought much about.

Among my friends at Stanford, a special thanks goes to Federico Rossi. He has done so much for me as a dedicated researcher, a talented photographer, and an avid astronomer. I thoroughly enjoyed our collaborations and I hope we can continue to work together. To my other friends in the ASL and the Aero & Astro department, in particular Joseph Starek, Ashley Clark, David Manosalvas, Christopher Pepper, Todd Chapman, Ed Schmerling, Sumeet Singh, Ross Allen, Yin-Lam Chow, Zach Sunberg, Brian Ichter, and Ben Hockman – thank you for making the past 5 years at Stanford fun and memorable. Thank you for all your support through the good times and the bad. The hard work and dedication that exemplifies each of you has been an inspiration for me.

A special thanks goes to Edward Mazenc, from whom I have learned so much, particularly in the finer thing in life. More importantly, I will always cherish our conversations about anything and everything. To Trevor, Sean, Konstantine, Manan, Amy, Jamie, Catherine, and Lily – thanks for your continuing support, even if they are from a distance.

I would like to acknowledge the Dr. Cleve B. Moler Stanford Graduate Fellowship for providing the financial backing for my research, as well as the National Science Foundation grant CMMI-1454737.

Finally, I would like to thank my mother Liu Liping and father Zhang Baochun. They have sacrificed their careers and their way of life for me, and I would not be here without their love and unwavering support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a systematic approach to the design, control, and evaluation of personal urban mobility systems including one-way car sharing (also called mobility-on-demand (MoD)) and autonomous mobility-on-demand (AMoD), where self-driving vehicles transport customers within an urban environment. First, we develop queueing-theoretical models to answer high level strategic questions such as: how many vehicles are needed in a city to ensure satisfactory service? Using these theoretical models, we develop computationally efficient coordination algorithms to route vehicles in very large systems (with thousands of vehicles). These models and algorithms allow us to assess the societal and sustainability benefits of shared autonomous vehicles. Coordinating a fleet of autonomous vehicles brings many operational challenges. For example, AMoD services should (1) ensure high quality-of-service throughout the city (achieved through rebalancing), (2) avoid causing excess congestion in the road network, and (3) account for operational constraints such as range constraints, charging constraints, parking availability, etc. Together, the techniques introduced in this thesis allow us to account for these complexities.

The models and algorithms presented in this thesis use ideas from queueing network theory, receding horizon control, and convex optimization. These techniques constitute the first rigorous, stochastic approach to the problem of system-wide coordination of autonomously driving vehicles, and the first rigorous study of the impact of autonomously rebalancing vehicles on traffic congestion.

1.1 Future of Personal Urban Mobility

Private vehicles have dominated the transportation landscape of the past century by enabling fast and convenient point-to-point mobility within large sprawling cities. In the US, private vehicles accounted for 83% of all passenger trips in 2009 (with public transit at less than 5%) [1]. The widespread use of private automobiles has had detrimental consequences for the environment and urban infrastructure. Urban transportation in the US account for over half of the total oil consumption [2] while producing 20% of total carbon dioxide emissions [3]. Meanwhile, the availability of urban space for roads and parking is becoming increasingly limited, causing excessive congestion (the average commuter spent 42 additional hours in traffic in 2014 [4]) and wasting fuel for cars looking for parking in dense urban areas. In addition, private vehicles are vastly under-utilized, with average utilization rates of only 5 to 10% [5].

The problems are even worse on a world-wide level, where urban populations in existing cities are projected to grow by *2.5 billion* by the year 2050 [6]. The continued reliance on private vehicles for personal transportation is widely viewed as unsustainable for the future [7].

1.2 Mobility-on-Demand

Recent efforts have led to several solutions aimed at increasing sustainability and increasing vehicle utilization. Examples of these efforts include bike sharing systems (currently in over 600 cities worldwide [8]) and one-way car sharing systems such as Car2go [9], Autolib', and DriveNow. A complete solution, according to [7], involves the merging of four key enabling technologies: (1) the mobility internet, (2) new vehicle designs, (3) clean, smart energy, and (4) dynamically priced markets. Many of these technologies are approaching maturity. For example, the mobility internet is being realized through ride booking apps used by Uber and new electric vehicle designs are being pioneered by Tesla Motors. Collectively, vehicle sharing systems employing these technologies are known as *mobility-on-demand* (MoD) systems.

An MoD system typically contains a number of stations scattered throughout an urban area where vehicles are parked. A customer arrives at a station, reserves a vehicle, drives the vehicle to a station near his/her destination, and returns the vehicle there. Since the distribution of customer origins and destinations are not the same, this type of one-way service will inevitably lead to vehicle imbalances among the stations. In other words, vehicles will rapidly aggregate at some stations and be depleted at others. Surprisingly, even if the system parameters are homogeneous (all stations have identical customer arrival rates, the routing distributions are uniform, and the travel time distribution between any two stations is the same), the stochastic nature of the customer arrivals will quickly drive the system out of balance [10]. Imbalances in the system result in long wait times (in the case where customers form queues at stations) or high rates of customer loss.

The problem of redistributing vehicles to meet customer demand, known as the *rebalancing problem*, is ubiquitous for all MoD systems. For bike sharing systems, the problem is two-fold: high demand stations experience a shortage of bikes while low demand stations often have full racks of bikes preventing customers from returning their rentals. The most common approach to this problem is to load excess bikes onto trucks and drop them off at stations where they are needed. To reduce cost, bike sharing operators seek to find truck routes that rebalances as many bikes as possible while traversing the shortest distance. The optimal bike rebalancing problem is NP-hard [11] and solution techniques have included mixed integer linear programming [11, 12], constraint programming [13], and heuristic approaches [14].

For one-way car sharing systems, it is generally not possible to rebalance multiple cars at the same time as in bike sharing systems. Instead, two rebalancing approaches are used: (1) user-based rebalancing and (2) operator-based rebalancing. In user-based approaches, the operator offer financial incentives to the users to influence their booking origins or destinations, or encourage ride sharing and ride splitting. However, user-based approaches usually cannot meet all of the system's rebalancing needs, since modeling user choice is difficult and control over user behavior is not guaranteed [15]. Operator-based rebalancing occurs when the system operator sends hired drivers to different parts of the city to rebalance the vehicles. Research in this regime often formulate the rebalancing problem as a mixed integer linear program with the objective of maximizing profit and rebalancing modeled as a fixed cost [16, 17]. However, these formulations do not account for the transportation of the rebalancing drivers, who may get "stuck" at a station after rebalancing, especially if public transit is not widely available. Several strategies have been proposed to route these drivers including shared shuttles [18] and a hybrid approach allowing drivers to take select customers to their destinations [19, 20].

Though efficient rebalancing strategies for bike sharing and car sharing MoD systems are still topics of active research, an emerging concept of using *autonomous vehicles* in an MoD system is quickly gaining momentum.

1.3 Autonomous Mobility-on-Demand

Recently, a transformational technology is emerging where personal on-demand mobility is provided by self-driving cars [21, 5]. Autonomous driving holds great promise for MoD systems because the vehicles can rebalance themselves (thus eliminating the rebalancing problem at its core), enable system-wide coordination, free passengers from the task of driving, increase mobility for those unable or unwilling to drive, and potentially increase safety. An autonomous mobility-on-demand (AMoD) system can have additional benefits: (1) it can offer the same level of convenience as privately owned vehicles in terms of providing personal door-to-door mobility, (2) it eliminates the need to find parking spaces in dense urban areas, (3) it can reduce travel costs when compared to both traditional taxi systems (since there would be no driver) and privately owned vehicles [22], and (4) by using electric vehicles, AMoD systems can be environmentally sustainable . These potential societal benefits and market demand have led to a race between automobile manufacturers and technology companies to develop and perfect autonomous vehicle technology. Indeed, while much research has been focused on autonomy for individual vehicles, relatively little is known about how to design and operate robotic transportation networks [21].

Specifically, an AMoD system consists of a fleet of robotic vehicles servicing spatially-localized customer requests within an urban environment. The requests constitute a dynamic process, meaning that they are revealed incrementally in time (as opposed to static, where all the requests are known in advance). Furthermore, the arrival times and locations of the requests are uncertain, which necessitates a probabilistic analysis. If requests cannot be immediately serviced, they may form queues within the environment or they may be lost. The performance of the system can be assessed by several quality-of-service metrics such as the average wait time of each request before it receives service, the total service time (waiting + travel), or the probability that each request can be immediately serviced by an available vehicle. Thus, the problem of coordinating the vehicles to service these requests is a joint task allocation, scheduling, and routing problem. Finally, the problem is further complicated by operational and structural constraints such as congestion constraints (managing the routes of empty rebalancing vehicles to avoid causing excessive congestion given an urban road network) and energy constraints (limited battery capacities of electric vehicles).

These operational complexities make it difficult to directly apply results from queueing theory or static combinatorial optimization. In the next section we summarize several current approaches that have been taken to address parts of the problem.

1.4 Previous Approaches

The problem of designing, analyzing, and controlling vehicle fleet operations has been studied in the fields of transportation science, operations research, and control theory. In this section we review approaches related to the control of MoD and AMoD systems from both the tranportation and robotics communities and discuss their merits and limitations.

1.4.1 Dynamic Traffic Assignment

In the field transportation science, Dynamic traffic assignment (DTA) is a general class of techniques for route planning for transportation networks. Its applications range from identifying and improving inefficiencies in road network infrastructure to providing real-time traffic information for travelers [23]. DTA seeks to optimize the time varying flows on each link of a transportation network, taking into account congestion effects along road links and at intersections. Typically, the objective is to either find a system optimum (SO), where routes are chosen such that the average travel time is minimized, or a user equilibrium (UE), where routes are chosen such that any user wishing to change routes would experience a longer travel time [24]. In DTA approaches, the (time dependent) origins and destinations are first specified, then the vehicle routes are optimized on a time-expanded network. A wide range of formulations exist to perform this optimization, including discrete-time mathematical programming, continuous-time optimal control, variational inequality, and simulation-based methods[25].

In the first three formulations, the travel time (or cost) associated with each link is typically a function of the volume of traffic that flows through the link. A major issue for these approaches has been choosing meaningful functions to model congestion. Furthermore, more complicated functions preclude the existence of efficient solution techniques for these optimization problems. The desire for more realistic representations of traffic flow has driven the development of simulation-based DTA methods [25].

The type of the simulation chosen is dependent on the trade-off between computation time and modeling accuracy. Simulation frameworks of different fidelity may be referred to as microscopic, mesoscopic, or macroscopic. The highest accuracy microscopic models [26, 27] take into account the dynamics of each vehicle and vehicle-to-vehicle interactions, where macroscopic models deal with local averages such as vehicle density and speed [28]. While simulation-based approaches are ideal for specific case studies, their results do not provide general insight into system behavior.

In general, DTA can solve the routing problem for customer-carrying vehicles, but does not take into account the additional operational constraints associated with AMoD such as vehicle rebalancing (routing of empty vehicles).

1.4.2 Approaches for the rebalancing problem

The operational constraints of AMoD are similar to constraints encountered in current car sharing or car rental systems. For rebalancing in car sharing and car rental systems, a popular approach is to use a mixed integer linear program formulation [16, 29]. However, these formulations do not take into account the dynamic arrival process of customers and the stochastic nature of the system. Furthermore, these approaches generally do not scale well to large systems.

A fluidic model for AMoD systems is introduced in [21], where customers and vehicles are modeled as a continuum. The existence of a balanced equilibrium is proven under this model and the optimal rebalancing rates could be solved as a linear program. However, this model, by its very nature, does not provide information about the effect of stochastic fluctuations in the system and, more importantly, does not allow the computation of quality-of-service metrics.

The stochastic aspects of the problem may be captured under a queueing-theoretical framework. A closed queueing network model of a car sharing system is described in [30] to study the required fleet size to maximize profit. Customer requests are modeled as a Poisson process and the queueing-theoretical framework allows the computation of vehicle availability (the probability that a vehicle is free upon the arrival of a customer request). However, vehicle rebalancing was not taken into account. Indeed, one of the key contributions of this thesis is applying this queueing-theoretical framework to *synthesize* an optimal rebalancing policy, rather than simply *analyzing* the evolution of the vehicle distribution under the customers' routing choices.

1.4.3 Dynamic Vehicle Routing

Another approach related to AMoD is called Dynamic Vehicle Routing (DVR), part of a family of routing and scheduling problems known as Vehicle Routing Problems (VRP) – a good summary can be found in [31]. VRPs have been well-studied for many application domains including freight transport, courier services, and emergency services. For example, a shipping company needs to route and schedule the vehicles in its fleet to ensure the timely delivery of goods. Commercial airlines must manage its fleet of aircraft and crew to ensure on-time service while satisfying the working hour constraints of the crew. A city is responsible for managing its fleet of emergency services vehicles to ensure adequate coverage for quick response. Most of these problems can be solved as a *static* routing problem, where the origin and destination of each trip is well known beforehand, and a good amount of time can be dedicated to optimizing the routes of the fleet. On the other hand, problems in which requests are not known *a priori*, but are revealed incrementally in time are *dynamic* vehicle routing problems.

More specifically, dynamic vehicle routing problems where vehicles transport people or goods between an origin and a destination are called dynamic pickup and delivery problems (DPDPs) [32]. Three types of pickup and delivery problems are typically studied: (1) many-to-many [11, 33], where each request can be routed to a number of destinations, (2) one-to-one, where each request has a single origin and destination [34], and (3) one-to-many-to-one, where goods initially located at a depot are delivered to customers, and goods that originate from the customers are collected and delivered to the depot [35].

The problem of controlling an AMoD system can be considered an instance of the dynamic one-to-one pickup and delivery problem. To more accurately characterize these systems, a number of additional attributes and constraints can be applied:

1. Requests are for immediate service, as opposed to prior booking with time windows.
2. Stochastic customer arrivals and travel times. Typically, customer requests are represented using a Poisson process [36].
3. Single-occupancy vehicles. This assumption captures the notion of *personal* mobility. Vehicles with capacity greater than one can be used for ride-sharing systems.

Solution approaches for DVR fall under three categories: (1) heuristic approaches, (2) online algorithms, and (3) algorithmic queueing theory.

Heuristic Approaches The key characteristic of heuristic approaches is that the performance of the control algorithms are evaluated by numerical, statistical, or experimental studies. No performance guarantees or notions of global optimality are available for these approaches, although they can perform well in real-world scenarios. Heuristic approaches have been commonly used for vehicle-to-customer assignment in taxi systems. AMoD systems are similar to taxi systems apart from one key difference: a taxi system operator typically cannot enforce a balanced vehicle distribution because taxi drivers have the final say on where to go to wait for new customers. Drivers in a taxi system will act selfishly to maximize their own profits, while vehicles in an AMoD system can act cooperatively. Thus, taxi systems are not as amenable to system-wide coordination and optimization. The simplest assignment heuristic is to form a queue of customer requests and assign the nearest vehicle to each request in a first-come-first-served fashion. This is the case for many existing taxi systems and is clearly sub-optimal [37]. Approaches that aim to improve upon this algorithm typically solve static routing problems [31] repeatedly with a heuristically chosen re-optimization horizon [37, 38]. Other approaches such as [39, 40] use heuristics to move empty vehicles to adjacent regions for rebalancing and validate their performance using simulation studies to show improvement over current taxi operations.

Online Algorithms Online algorithms are a class of control algorithms for dynamic routing problems whose performance are evaluated via *competitive analysis*. The competitive ratio is defined to be the ratio between the performance of the online algorithm and the performance of an optimal offline algorithm, where the time and locations of all the requests are known *a priori*. The control algorithms are then designed to minimize the worst-case competitive ratio. These algorithms have been used to study many vehicle routing problems including the traveling salesman problem (TSP) [41] and the single vehicle pickup and delivery problem [42]. However, comparing against optimal offline solutions may be overly pessimistic, and statistical information about the problems are not exploited.

Algorithmic Queueing Theory The algorithmic queueing theory approach extends traditional queueing theoretical methods to take into account the spatial distribution of the demands [43, 44, 45]. The general approach consists of three steps:

1. A distributed queueing model of the system is created, consisting of a number of vehicles servicing demands on a Euclidean space (or a road network [46]). Customer demands are generated via a spatial-temporal Poisson process.

2. Fundamental limitations on performance are established, independent of the routing scheme. For example, stability properties of the system are determined using queueing-theoretical tools to find the minimum number of vehicles required to service demand.

3. Routing algorithms that are optimal (or a constant-factor from optimal) are developed, often in asymptotic regimes of performance.

For example, in [34], a polynomial-time approximation algorithm that is asymptotically optimal is developed for the static one-to-one pickup and delivery problem as the number of requests increases.

For the dynamic version of the problem, necessary and sufficient conditions for the existence of a stable routing policy are derived using queueing-theoretical concepts. A load factor, ϱ, is defined as

$$\varrho = \frac{\lambda}{m}[\mathbb{E}_{\varphi_P \varphi_D}\|Y - X\| + W(\varphi_D, \varphi_P)],$$

where λ is the expected arrival rate of customers, m is the number of vehicles, φ_P is the spatial distribution of pickup locations, φ_D is the spatial distribution of destination locations, X and Y are random variables for the locations of pickups and destinations, respectively, and $W(\varphi_D, \varphi_P)$ is the Wasserstein distance (or Earth Mover's Distance) [47] between the pickup distribution and destination distribution. A load factor of less than one is deemed necessary and sufficient for the existence of stable routing policies. However, most of the algorithms developed using the algorithmic queueing theory approach are only provably optimal in asymptotic regimes, such as light load ($\varrho \to 0$), or heavy load ($\varrho \to 1$). In medium load regimes, where systems operate, it is difficult to assess the performance of these algorithms. Furthermore, it's also difficult for these approaches to account for additional operational complexity such as congestion and charging constraints.

1.5 Contributions

The objective of this thesis is to develop a systematic approach to the design, control, and evaluation of vehicle sharing systems, with a particular focus on autonomous mobility-on-demand systems. Our methodology consists of four steps:

1. A rigorous stochastic model of the vehicle sharing system (AMoD or MoD).
2. Practical closed-loop coordination algorithms inspired by the stochastic model. The coordination algorithms should be computationally efficient and scalable to large systems.
3. The performance of the coordination algorithms are evaluated through case studies using real-world data. These case studies also allow us to evaluate the societal benefits of AMoD and MoD systems.
4. Additional operational constraints including congestion constraints and charging limitations of electric vehicles are considered.

Together, these techniques enable us to gain a holistic understanding of the benefits and drawbacks of AMoD and MoD, and can be applied to study other types of transportation networks.

This thesis is divided into two parts. In Part 1, we present queueing-theoretical models of AMoD (Chapter 3) and MoD (Chapter 4) systems using the theory of Jackson networks. We use the insights taken from these models to develop practical closed-loop rebalancing policies for AMoD and MoD systems, and demonstrate their performance using case studies with real-world data. In Part

2, we investigate additional structural and operational constraints associated with AMoD systems. In particular, in Chapter 5 we study the impact of rebalancing on traffic congestion and how it relates to the underlying structural properties of the urban road network. In Chapter 6 we study the operational complexities associated with battery electric vehicle fleets with limited range.

In the following we summarize the specific contributions of each chapter.

Chapter 2: Preliminaries In this chapter we present several well known results from queueing theory, chiefly the theory of Jackson networks. We also present the formulation of the well-known multi-commodity flow problem.

Chapter 3: Queueing network analysis and control for AMoD systems In this chapter we study a queueing-theoretical model for an AMoD system using the theory of Jackson networks. Our AMoD model consists of N stations within a given geographical area and m (autonomous) vehicles providing service to customers. Customers arrive at each station according to a Poisson process, and choose a destination station with probability p_{ij}. If a customer arrives at a station to request a vehicle and none are available, the customer immediately leaves the system. This “passenger loss” assumption allows us to maintain analytical tractability and is suitable in situations where high quality of service is desired. In this case, the performance metric of interest is the availability of vehicles, or the probability that a new customer arrival sees at least one free vehicle at the station. The contribution of this chapter is threefold. First, we propose a queueing-theoretical model of an AMoD system cast within a Jackson network model. Second, we study the problem of synthesizing rebalancing algorithms, where the control objective is to minimize the number of (autonomously) rebalancing vehicles on the roads while keeping vehicle availabilities balanced throughout the network. Remarkably, we show that under certain assumptions an optimal policy can be solved as a linear program. Third, we relax our passenger loss assumption and develop a real-time rebalancing algorithm for the case where customers would form a queue to wait for vehicles rather than immediately departing. We apply this algorithm to case studies of New York City and Singapore as well as an experimental testbed with 8 mobile robots. The case study of New York City shows that the current taxi demand in Manhattan can be met with about 8,000 autonomous vehicles (roughly 70% of the size of the current taxi fleet operating in Manhattan). For Singapore, we show that hypothetically an AMoD system can meet all travel demands within the city with only 1/4 to 1/3 the size of the current vehicle fleet. These results were reported in [22, 48, 49].

Chapter 4: Queueing network analysis and control for Human-driven MoD In this chapter, we extend the Jackson network model developed in Chapter 3 to study the design and operation of a (human-driven) mobility-on-demand system (e.g. one-way car sharing system). Since the vehicles are not able to rebalance themselves, we consider a team of rebalancing drivers to drive excess vehicles to stations where they are needed. However, this in turn results in an imbalance of drivers across the stations. To rebalance these rebalancing drivers, we allow drivers to ride with select customers (or drive select customers) to their destinations, similar to a taxi service. The contribution in this chapter is fourfold. First, we model the coupled problem of rebalancing vehicles and drivers in an MoD system as two coupled closed Jackson networks with passenger loss. Second, we present two approaches for the open-loop control of an MoD system. In the first approach, the optimal rebalancing parameters are solved by two decoupled linear programs, and are therefore efficient to compute, but only approximately guarantee balance of the system. In the second approach, nonlinear optimization techniques are used (with higher computational cost) to balance the system exactly. Third, we apply such approaches to the problem of system sizing. Our key finding is that the optimal vehicle-to-driver ratio in an MoD system should be between 3 and 5. Finally, leveraging the aforementioned open-loop control strategies, we devise a real-time closed-loop rebalancing policy and demonstrate its performance for a case study of Manhattan. In particular, we show that an MoD system can satisfy all existing taxi demands in Manhattan with around the same number of vehicles as current taxis (approximately 11,000), but only needs 1/3 to 1/4 the number of drivers. The results of this chapter were published in [20].

Chapter 5: Congestion-aware AMoD There has been strong debate in the automotive community about the effects of AMoD systems on traffic congestion. Many argue that autonomous vehicles will in fact increase congestion on the road due to many empty vehicle (rebalancing) trips. These claims, however, do not account for the fact that in an AMoD system, the vehicles can be cooperatively routed to reduce congestion. We show that when vehicles are routed intelligently, rebalancing vehicles almost always do not increase congestion on the road. In this chapter we extend the work in Chapter 3 to address rebalancing on congested road networks by formulating a novel rebalancing algorithm that optimizes not only the origins and destinations of the rebalancing vehicles, but their routes as well. Specifically, the contribution of this chapter is threefold. First, we propose a network flow model of an AMoD system, whereby customer-carrying and empty rebalancing vehicles are represented as flows over a capacitated road network (in this model, when the flow of vehicles along a road reaches a critical capacity value, congestion effects occur). Within this model, we provide a condition for the road graph that needs to be satisfied for congestion-free customer and rebalancing flows to exist. Most importantly, under the assumption of a capacity-symmetric road network, we investigate an existential result that leads to two key conclusions: (1) *rebalancing does not increase congestion*, and (2) for certain cost functions, the problems of finding customer and rebalancing flows can be decoupled. Second, leveraging these theoretical insights, we propose a computationally-efficient algorithm for congestion-aware routing and rebalancing of an AMoD system that is broadly applicable to time-varying, possibly asymmetric road networks. Third, through numerical studies on real-world traffic data, we validate our assumptions and show that the proposed real-time routing and rebalancing algorithm outperforms the point-to-point rebalancing algorithm (introduced in Chapter 3) in terms of lower customer wait times by avoiding excess congestion on the road. The results of this chapter were published in [50].

Chapter 6: Model predictive control for AMoD with charging constraints A major challenge for the modeling and control of AMoD systems is the integration of practical operational constraints such as the availability of parking spaces and range constraints for electric vehicles. In this chapter we focus on the control of an AMoD system with electric vehicle range and charging constraints. To model these constraints we must keep track of the state-of-charge for each vehicle, which greatly increases the size of the state space (in previous chapters the vehicles are assumed to be homogeneous) and therefore the computational complexity. We approach this problem from the perspective of model predictive control (MPC) (also known as receding horizon control), whereby an open-loop optimization problem is solved at each discrete time step to yield a sequence of control actions up to a fixed horizon, and the first control action is executed. Due to its iterative nature, MPC can achieve closed-loop performance, is robust to modeling errors, and is well suited for complex, constrained systems. Specifically, the contribution of this chapter is threefold. First, we propose a novel discrete-time model of an AMoD system and we show that this formulation allows the easy integration of a number of operational constraints, with a special focus on electric vehicle charging constraints. Second, leveraging our model, we design a model predictive control algorithm for the optimal coordination of an AMoD system and prove its stability in the sense of Lyapunov. Finally, by using taxi data from New York City, we show that the MPC algorithm can be run in real-time for moderately-sized systems and compare its performance to four other AMoD control algorithms and taxi dispatch algorithms in the literature. We show that the MPC algorithm not only outperforms other algorithms in terms of customer wait times, but can also be used as an optimal performance benchmark to evaluate other AMoD control algorithms. The results of this chapter were published in [51].

Chapter 7: Conclusions Many interesting problems associated with the operation of AMoD systems remain. In this chapter we present our conclusions and propose directions for future research.

Chapter 2

Preliminaries

In this chapter we review some basic results in queueing theory and network flow. The queueing theory results are heavily used in Chapters 3 and 4. Results in network flow are primarily used in Chapter 5.

2.1 Queueing Theory

First we give basic definitions of Poisson and Markov processes. We then summarize key results of Markovian queues (including the M/M/1 queue) and present the theory of Jackson networks as a network of Markovian queues.

2.1.1 Poisson Process

There are many ways to characterize the Poisson process. In the following, we define it from the perspective of a *counting process*. A stochastic process $\{N(t), t \geq 0\}$ is called a counting process if $N(t)$ represents the total number of events that have occurred by time t. A counting process $N(t)$ thus has the following properties:

1. $N(t) \geq 0$ and is integer valued.

2. $N(t)$ is monotonically increasing. In other words, if $s < t$, then $N(s) \leq N(t)$.

3. For $s < t$, $N(t) - N(s)$ is the number of events that occur in the interval $(s, t]$.

A counting process has *independent increments* if events that occur in disjoint time intervals are independent. $N(t)$ has *stationary increments* if the distribution of the number of events that occur in any time interval depends only on the length of that interval.

A *Poisson process* with rate λ, $\lambda > 0$, is a counting process that satisfies

1. $N(0) = 0$.

2. The process has independent increments.

3. The number of events occurring in any interval t is Poisson distributed with mean λt, i.e.

$$\mathbb{P}\{N(t+s) - N(s) = n\} = e^{-\lambda t}\frac{(\lambda t)^n}{n!}, \qquad \text{for all } n \in \mathbb{N}. \tag{2.1}$$

It follows from 2.1 that a Poisson process also has stationary increments and that $\mathbb{E}[N(t)] = \lambda t$. Denote by T_n the amount of time elapsed between the $(n-1)^{\text{th}}$ event and the n^{th} event, also called the interarrival time. It can be shown that T_n, $n = 1, 2, ...$ are independent identically distributed (i.i.d.) random exponentially distributed variables with mean $1/\lambda$. This means that the Poisson process from time t is independent from what occurred prior to time t, also called the *memoryless* property.

The Poisson process has a number of other useful properties [52]. Two of these properties are exploited in this thesis:

1. **Merging:** Let $N_1(t), N_2(t), ..., N_n(t)$ be independent Poisson processes with rates $\lambda_1, \lambda_2, ..., \lambda_n$, respectively. Merge these processes into a single process $N(t) = N_1(t) + N_2(t) + ... + N_n(t)$. Then $N(t)$ is also a Poisson process with rate $\lambda = \sum_{i=1}^{n} \lambda_i$.

2. **Bernoulli Splitting:** Let $N(t)$ be a Poisson process with rate λ. Split $N(t)$ into two processes by independently assigning each arrival to the first process with probability p (and to the second process with probability $1-p$). Then the two resulting processes are Poisson with rates λp and $\lambda(1-p)$, respectively.

2.1.2 Markov Process

Let $\{X(t), t \geq 0\}$ be a stochastic process defined on the countable state space Ω. $X(t)$ is a *Markov process* if

$$\mathbb{P}\{X(t+s) = j \mid X(t) = i, X(u) = x(u), 0 \leq u < t\} = \mathbb{P}\{X(t+s) = j \mid X(t) = i\}, \tag{2.2}$$

for $i, j, x(u)$ in Ω, and s, t in $\mathbb{R}_{\geq 0}$. In other words, the probability of future events depend on only the current state, and not on any states in the past. In general, the conditional probability (2.2) may depend on both s and t. When (2.2) is independent of t for all i, j in Ω and $s \geq 0$, $X(t)$ is called a time-homogeneous Markov Process [53]. In this case,

$$\mathbb{P}\{X(t+s) = j \mid X(t) = i\} = p_{ij}^{s}, \tag{2.3}$$

where p_{ij}^s is called the transition function of X. The transition function p_{ij}^t satisfies (1) $p_{ij}^t \geq 0$, (2) $\sum_{k \in \Omega} p_{ik}^t = 1$, and (3) the Chapman-Kolmogorov equation $\sum_{k \in \Omega} p_{ik}^t p_{kj}^s = p_{ij}^{t+s}$.

When the state space is discrete, the path of a Markov process with transition function p_{ij}^t proceeds as follows: at time 0, the process is in an initial state $X_0 = X(0)$. It remains in that state for some positive time T_1, then jumps to another state X_1. Again, it remains at this state for some time T_2 before jumping to the next state X_2 at time $T_1 + T_2$, and so on. The sequence of states $\{X_n, n \in \mathbb{N}\}$ forms a Markov chain. Furthermore, the inter-transition times T_n (where T_n is the time spent in state X_{n-1}) are exponentially distributed depending on X_{n-1}. Many stochastic processes can be viewed as Markov processes. For example, the Poisson process is a Markov process. Other examples of Markov processes include the M/M/1 queue and Jackson networks, as we shall see in the following.

2.1.3 Markovian Queues

A queueing system is defined as follows:

1. A stream of customers arrive to a service center at random times.

2. The customers form a queue (a line) and wait to receive service.

3. A service center services the customer according to a specific service discipline (e.g. first-come-first-serve). Each customer receives service for some amount of time, then leaves the queueing system.

The queueing system is characterized by the arrival process of the customers, the service time distribution, the number of servers in the queue, and the service discipline. Queues are often described using the notation A/B/c, where A denotes the customer arrival process, B denotes the service time distribution, and c denotes the number of servers in the queue [52]. One of the most basic and most studied type of queueing systems is the M/M/1 queue (M stands for memoryless, or Markovian), where customers arrive according to a Poisson process, wait in line, and are serviced by a single server with an exponentially distributed service time. The customer arrival rate is denoted by λ and the mean service time is given by $1/\mu$ (μ is also called the mean service rate).

Since queueing models are often used to evaluate the performance of real-life processes in steady state, the quantities of interest in a queueing system include

- N = average number of customers in the system

- L = average number of customers waiting in the queue (average queue length)

- W = average customer waiting time in queue

- T = average time customer spends in the system.

Under a wide range of conditions, a relationship exists between the mean values of the number of customers, the arrival rate, and the waiting time, known as Little's theorem [52].

$$N = \lambda T \qquad \text{and} \qquad L = \lambda W. \tag{2.4}$$

This relationship formalizes the intuitive notion that heavily loaded systems will typically experience long wait times. The M/M/1 queue can interpreted as a Markov process $\{N(t), t \geq 0\}$, where the state space is the number of customers in the system. By analyzing the underlying Markov chain, it can be shown that the steady state probability of having n customers in the system is given by

$$\mathbb{P}\{N(t) = n\} = \rho^n(1-\rho), \qquad n \in \mathbb{N}, \tag{2.5}$$

where $\rho = \lambda/\mu$ is called the *load factor* of the system. The load factor is an important quantity for the stability of the queueing system. The average number of customers in the system in steady state can be characterized using ρ as follows:

$$\begin{aligned} N = \lim_{t\to\infty} \mathbb{E}[N(t)] &= \sum_{n=0}^{\infty} n\mathbb{P}\{N(t) = n\} \\ &= \sum_{n=0}^{\infty} n\rho^n(1-\rho) \\ &= \rho(1-\rho)\sum_{n=0}^{\infty} n\rho^{n-1} \\ &= \rho(1-\rho)\frac{1}{(1-\rho)^2} \\ &= \frac{\rho}{1-\rho} \end{aligned} \tag{2.6}$$

We see that as ρ approaches 1, N approaches ∞, so the queue becomes unstable when $\rho \geq 1$ (customers arrive at a faster rate than they can be serviced, hence the length of the queue will grow indefinitely).

Other types of Markovian queues include M/M/c (multiple server, $c > 1$) and M/M/∞ (infinite server). For example, the load factor for an M/M/c queue is given by $\rho = \lambda/c\mu < 1$. For M/M/∞ queues, the average service time is $T = 1/\mu$ since there is no queueing delay. More general distributions for customer arrival and service can also be accommodated, such as G, for general distributions, and D, for deterministic [52]. Furthermore, other service disciplines such as last-come-first-serve and processor-sharing have been studied, as well as service modifications such as priority queues, finite capacity queues, and queues with vacations. However, these models are outside the scope of this thesis.

Finally, multiple queues can be linked to form queueing networks, where customers may travel throughout the network and receive service from several queues before leaving the network. A particular class of queueing networks, called product-form networks, are particularly powerful because analytical results for their performance are available and are relatively easy to compute. The first of these networks were identified by James Jackson [54] and are known as Jackson networks.

2.1.4 Jackson Networks

Consider a network consisting of $|\mathcal{N}|$ first-come first-serve queues (also called nodes), where $\mathcal{N}$ represents the set of queues in the network. Discrete agents[1] arrive from outside the network according to a stochastic process or move among the nodes. Agents that arrive at each node are serviced by the node, and proceed to another node or leave the system. A network is called *closed* if the number of agents in the system remains constant and no agents enter or leave the network. A Jackson network is a Markov process where agents move from node to node according to a stationary routing distribution r_{ij} and the service rate $\mu_i(n)$ at each node i depends only on the number of agents at that node, n [55, p. 9]. For the remainder of this section, we consider only closed networks. The state space of a closed Jackson network with m agents is given by

$$\Omega_m := \Big\{(x_1, x_2, \dots, x_{|\mathcal{N}|}) : x_i \in \mathbb{N} \text{ for all } i \in \mathcal{N}, \sum_{i=1}^{|\mathcal{N}|} x_i = m\Big\},$$

where x_i is the number of agents at node i. Jackson networks are known to admit a product-form stationary distribution, where the stationary distribution of the network is given by a product of the distribution of each node. In equilibrium, the throughput at each node (average number of agents moving through a node per unit time), denoted by π_i for $i \in \mathcal{N}$, satisfies the balance equations

$$\pi_i = \sum_{j \in \mathcal{N}} \pi_j r_{ji} \quad \text{for all } i \in \mathcal{N}. \tag{2.7}$$

For a closed network, Equation (2.7) does not have a unique solution, and $\pi := (\pi_1, \pi_2, \dots, \pi_{|\mathcal{N}|})^T$ only determines the throughput up to a constant factor, and is therefore called the relative throughput. The stationary distribution of the network is given by

$$\mathbb{P}\{x_1, x_2, \dots, x_{|\mathcal{N}|}\} = \frac{1}{G(m)} \prod_{j=1}^{|\mathcal{N}|} \pi_j^{x_j} \prod_{n=1}^{x_j} \mu_j(n)^{-1}.$$

The quantity $G(m)$ is the normalization constant needed to make $\mathbb{P}\{x_1, x_2, ..., x_{|\mathcal{N}|}\}$ a probability

[1]Entities moving through the queues are more commonly referred to as "customers" rather than "agents" in the queueing theory literature. We use the term "agents" instead of "customers" to avoid confusion later on, since these entities in our abstract model of an AMoD system will represent self-driving vehicles. In our model, "customers" will represent people requiring transportation within the network.

measure, and is given by

$$G(m) = \sum_{x \in \Omega_m} \prod_{j=1}^{|\mathcal{N}|} \pi_j^{x_j} \prod_{n=1}^{x_j} \mu_j(n)^{-1},$$

where $x := (x_1, x_2, \ldots, x_{|\mathcal{N}|})$. Many performance measures of closed Jackson networks can be expressed in terms of the normalization factor $G(m)$. In [55, p. 27], it is shown that the actual throughput of each node is given by

$$\Lambda_i(m) = \frac{\pi_i \, G(m-1)}{G(m)}. \tag{2.8}$$

One can further define the quantity

$$\gamma_i = \frac{\pi_i}{\mu_i(1)} \quad \text{for all } i \in \mathcal{N}, \tag{2.9}$$

where γ_i is referred to as the relative utilization of node i. [56, p. 128] showed that the marginal distribution of the queue length variable X_i at node $i \in \mathcal{N}$ is given by

$$\mathbb{P}\{X_i = x_i\} = \frac{\gamma_i^{x_i} \left[G(m - x_i) - \gamma_i G(m - x_i - 1)\right]}{G(m)}.$$

A quantity of interest is the probability that a node has at least 1 agent, which we refer to as the *availability* of node i, $A_i(m)$. This is given by

$$A_i(m) = 1 - \mathbb{P}\{X_i = 0\} = 1 - \frac{G(m) - \gamma_i G(m-1)}{G(m)} = \frac{\gamma_i G(m-1)}{G(m)}. \tag{2.10}$$

There are two exact methods to solve for the performance metrics of the network (such as throughput and availability). The first method, called the convolution method [55], explicitly solves for the normalization constant $G(m)$, from which the equilibrium distribution can be computed. However, if only the mean values of the performance metrics are desired, a second method called mean value analysis (MVA) can be used, which does not explicitly compute $G(m)$. The MVA algorithm is an iterative algorithm that calculates the mean waiting times $W_i(n)$ and the mean queue lengths $L_i(n)$ at each node i of a closed system of queues, where $n = 1, \ldots, m$ is the numbers of agents over which the algorithm iterates. The MVA algorithm relies on an important result known as the *Arrival theorem*, which states that the average number of agents found by an agent arriving at queue i is equal to the average number of agents seen by an outside observer in a network with $m - 1$ agents [52]. The MVA algorithm is summarized in Algorithm 1.

Algorithm 1 Mean Value Analysis [57].

```
Input: Service rates $\mu_i, i \in \mathcal{N}$
       Relative throughput $\pi_i, i \in \mathcal{N}$
       Number of agents $m$
Output: Mean wait times $W_i(m)$, $i \in \mathcal{N}$
       Mean queue lengths $L_i(m)$, $i \in \mathcal{N}$
procedure MVA($\mu_i, \pi_i, m$)
    $W_i(0) = 0$
    $L_i(0) = 0$
    for $n = 1$ to $m$ do
        $W_i(n) = \frac{1}{\mu_i}(1 + L_i(n-1))$ for single-server queues
        $W_i(n) = \frac{1}{\mu_i(1)}$ for infinite-server queues
        $L_i(n) = \frac{n\pi_i W_i(n)}{\sum_{j\in\mathcal{N}} \pi_j W_j(n)}$ for all $i \in \mathcal{N}$
        $n = n + 1$
```

2.2 Multi-commodity flow

Consider a directed graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each edge (u, v), where $u, v \in \mathcal{V}$, has an associated capacity $c(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}$ and a cost (which can be interpreted as travel time) $T(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}$. Also consider a set of demands, or commodities $\mathcal{D} = \{(s_i, t_i, \lambda_i)\}_i$, each with an associated origin $s_i \in \mathcal{V}$ (also called a source), destination $t_i \in \mathcal{V}$ (also called a sink), and demand intensity $\lambda_i \in \mathbb{R}_{>0}$.

We denote by $f_i(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}, i \in \mathcal{D}$ the *flow* of commodity i along edge (u, v). The total flow along edge (u, v), $f(u, v)$, is given by the sum of the flows of each commodity on that edge: $f(u, v) = \sum_{i\in\mathcal{D}} f_i(u, v)$. Flows on graph G must satisfy conservation constraints and capacity constraints. Conservation constraints are given by

$$\sum_{u\in\mathcal{V}} f_i(u, v) - \sum_{w\in\mathcal{V}} f_i(v, w) = \begin{cases} -\lambda_i & \text{if } v = s_i \\ \lambda_i & \text{if } v = t_i \\ 0 & \text{if } v \neq s_i, v \neq t_i \end{cases} \qquad \text{for all } i \in \mathcal{D}, v \in \mathcal{V}. \tag{2.11}$$

In other words, for each commodity, the total flow into each node equals the total flow out of the node. If the node is an origin or a destination, the flow into or out of the node is equal to its demand intensity. The capacity constraints are given by

$$\sum_{i\in\mathcal{D}} f_i(u, v) \leq c(u, v), \qquad \text{for all } (u, v) \in \mathcal{E} \tag{2.12}$$

that is, the total flow on each edge must be less than the capacity of that edge. A set of flows F which satisfy Constraints (2.11) and (2.12) for each commodity is called a set of *feasible flows.* The goal of multi-commodity flow is often to find a set of feasible flows that minimize the total cost, given by

$$\sum_{i \in \mathcal{D}} \sum_{(u,v) \in \mathcal{E}} T(u,v) f_i(u,v). \tag{2.13}$$

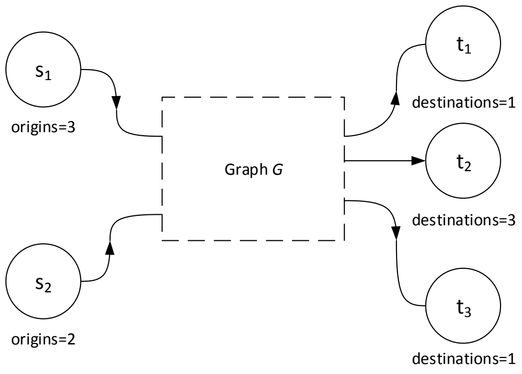
The multi-commodity flow problem, formulated in this fashion, is a linear program with optimization variables $f_i(u,v), i \in \mathcal{D}, (u,v) \in \mathcal{E}$, Objective (2.13), and Constraints (2.11) and (2.12). This problem can be solved in polynomial time to yield fractional flows. If the edge capacities and demand intensities are integers, and integer flows are desired ($f_i(u,v) \in \mathbb{N}$), the problem of finding feasible integral flows is known to be NP-complete, and the problem of finding the minimum cost integral multi-commodity flow is NP-hard [58].

2.2.1 Total unimodularity and minimum cost flow

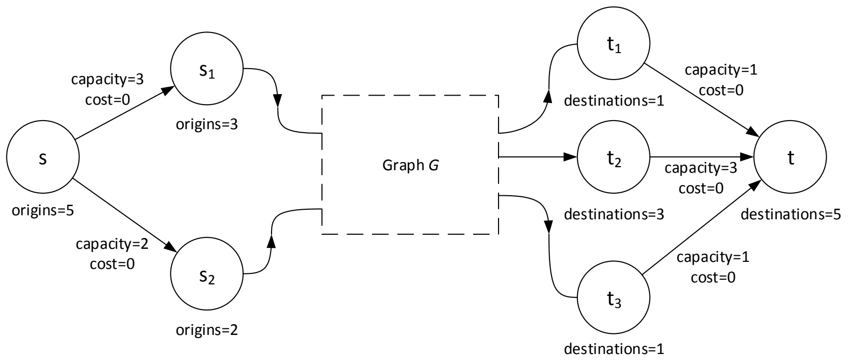
A square matrix with integer values is *unimodular* if its inverse is an integer matrix (equivalently, its determinant is $+1$ or -1). A (possibly non-square) matrix is *totally unimodular* if every single non-singular square submatrix is unimodular.

If the demands in the minimum cost multi-commodity flow problem are all of a single type, the problem is reduced to an instance of the well-known *minimum cost flow* problem. Note that it is possible to have multiple demands with multiple origins and destinations. If this is the case, the problem can be transformed to a problem with a single origin and single destination by augmenting the graph as shown in Figure 2.1.

If the edge capacities and demands are integral, the constraint matrix is totally unimodular [59] and the problem yields an integral optimal solution. This means that to solve the integer optimization problem, one simply has to solve the linear program relaxation (in polynomial time) to obtain the optimal integral solution. This property enables large-scale rebalancing problems, properly formulated, to be solved efficiently.



(a)



(b)

Figure 2.1: 2.1(a): A graph with a single commodity but multiple origins and destinations. Nodes s_1 and s_2 are origin nodes and t_1, t_2, and t_3 are destination nodes. 2.1(b): An augmented graph with two added nodes: origin node s and destination node t. An edge is created from s to s_1 (s_2) with 0 cost and capacity equal to the demand intensity of s_1 (s_2). Similarly, edges are created between t_i and t with 0 cost.

Part I

Queueing Network Models

Chapter 3

Queueing Network Analysis and Control for AMoD Systems

3.1 Introduction

The objective of this chapter is to provide a rigorous, stochastic approach to the problem of system-wide coordination of AMoD systems. Our approach for systematically designing and controlling these systems consists of three steps. First we develop a stochastic model of AMoD that is analytically tractable. Second, leveraging the insights from the model, we develop practical rebalancing algorithms suitable for large-scale implementation. Finally, we evaluate the potential benefits of AMoD systems through case studies using real-world data.

Rebalancing algorithms for AMoD systems have been investigated in [21] under a fluidic approximation (i.e., customers and vehicles are modeled as a continuum). While this approach provides valuable insights for the operation of an AMoD system, by its very nature, it does not provide information about the effect of stochastic fluctuations in the system (e.g., due to the customers' arrival process) and, most importantly, it does not allow the computation of key performance metrics such as availability of vehicles at stations and customer waiting times. This motivates the queueing-theoretical approach considered in this chapter. In this respect, the model presented here is related to [30, 60], where a transportation network comprising traditional (i.e., human-driven) shared vehicles is modeled within the framework of Jackson networks [55]. Queueing networks (including Jackson networks) have been used to analyze the behavior of communication networks and transportation systems. The key technical difference is that we employ queueing network theory to address the problem of *synthesizing* a control (rebalancing) policy, rather than *analyzing* the evolution of the vehicle distribution under the customers' routing choices.

The contributions of this chapter are as follows. First, we propose a queueing-theoretical model of an AMoD system cast within a Jackson network framework. Second, we study the problem of synthesizing rebalancing algorithms, where the control objective is to minimize the number of (autonomously) rebalancing vehicles on the roads while keeping vehicle availabilities balanced throughout the network. Remarkably, we show that under certain assumptions an optimal policy can be solved as a linear program. Third, we develop a practical real-time rebalancing algorithm based on the Jackson network model and we apply our theoretical and practical techniques to case studies of New York City and Singapore as well as an experimental testbed with 8 mobile robots. The case study of New York City shows that the current taxi demand in Manhattan can be met with about 8,000 robotic vehicles (roughly 70% of the taxi fleet operating in Manhattan). The case study of Singapore shows that an hypothetical AMoD system may be able to meet the mobility needs of the entire population of Singapore with a number of vehicles that is less than 40% of the current number of passenger vehicles. Finally, we conclude with a discussion of using queueing-theoretical methods to study the impact of AMoD systems on traffic congestion. Congestion effects are studied in greater detail in Chapter 5.

The remainder of this chapter is structured as follows: In Section 3.2 we model an AMoD system with rebalancing within a Jackson network framework. In Section 3.3 we formulate the optimal rebalancing problem, we show that it can be solved via a linear program, we leverage an iterative algorithm to compute relevant performance metrics (chiefly, vehicle availability at stations), and we use the theoretical insights to design a robust, real-time rebalancing policy. In Section 3.4 we apply our model and algorithms to case studies of New York City and Singapore, while in Section 3.6 we apply the same algorithms to a 8-vehicle hardware testbed. In Section 3.7 we discuss ways to extend our queueing-theoretical setup to include congestion effects. Finally, we draw our conclusions in Section 3.8.

3.2 Model of an AMoD system

In this section, we model an autonomous mobility-on-demand system within a queueing theoretical framework. Consider N stations placed within a given geographical area and m (autonomous) vehicles that provide service to passengers. Customers arrive at each station i according to a time-invariant Poisson process with rate $\lambda_i \in \mathbb{R}_{>0}$. Upon arrival, a customer at station i selects a destination j with probability p_{ij}, where $p_{ij} \in \mathbb{R}_{\geq 0}$, $p_{ii} = 0$, and $\sum_j p_{ij} = 1$. Furthermore, we assume that the probabilities $\{p_{ij}\}_{ij}$ constitute an irreducible Markov chain. If there are vehicles parked at station i, the customer takes the vehicle and travels to his/her selected destination. Instead, if the station is empty of vehicles, the customer immediately leaves the system. This type of customer model will be referred to as a "passenger loss" model (as opposed to a model where passengers form a queue at each station). A consequence of the passenger loss model is that the number of customers at each station at a fixed instant in time is 0 (since customers either depart immediately with a vehicle or leave the system). We assume that each station has sufficiently many parking spaces so that vehicles can always immediately park upon arrival at a station. The travel time from station i to station j is an exponentially distributed random variable with mean equal to $T_{ij} \in \mathbb{R}_{>0}$. The travel times for the different passengers are assumed to constitute an independently and identically distributed sequence (i.i.d.). The vehicles can *autonomously* travel throughout the network in order to rebalance themselves and best anticipate future demand. The performance criterion that we are interested in is the availability of vehicles at each station (the probability that at least one vehicle is at the station or conversely the probability that a customer will be lost).

A few comments are in order. First, this model closely resembles a setup with impatient customers, not willing to make use of an AMoD system if waiting is required. In such a system, customers may have the option to choose between several mobility service providers, and will choose the one with the highest vehicle availability. In this respect, the model appears to be suitable for studying the benefits of AMoD systems whenever high quality of service (as measured in terms of waiting times for available vehicles) is required. From a practical standpoint, the loss model assumption significantly simplifies the problem, as it essentially allows us to decouple the "vehicle process" and the "customer process" (see Section 3.2.1). Second, travel times, in practice, do not follow an exponential distribution. However we make this assumption as (1) it simplifies the problem considerably, and (2) reasonable deviations from this assumption have been found not to alter in any practical way the predictive accuracy of similar spatial queueing models used for vehicle routing [61]. Third, the assumption that the probabilities $\{p_{ij}\}_{ij}$ constitute an irreducible Markov chain appear appropriate for dense urban environments. Finally, this model does not consider congestion, which is a critical aspect for the efficient system-wide coordination of autonomous vehicles in an AMoD system. The inclusion of congestion effects will be discussed in Chapter 5.

3.2.1 Casting an AMoD system into a Jackson network model

The key idea to cast an AMoD system into a Jackson network model is to consider an *abstract queueing network* where we identify the stations with single-server (SS) nodes[1](also referred to as "station" nodes) and the roads with infinite-server (IS) nodes (also referred to as "road" nodes). Assume, first, the simplified scenario where vehicles do not perform rebalancing trips (in which case, the model is essentially identical to the one in [30]). In this case, at each station node, vehicles form a queue while waiting for customers and are "serviced" when a customer arrives. A vehicle departing from an SS node moves to the IS node that connects the origin to the destination selected by the customer. After spending an exponentially distributed amount of time in the IS node (i.e., the "travel time"), the vehicle moves to the destination SS node. According to this model, once a vehicle leaves an SS (station) node i, the probability that it moves to the IS (road) node ij is p_{ij}. The vehicle then moves to SS (station) node j with probability 1 (see Figure 3.1). Note that with

[1]Nodes and queues are used interchangeably in this chapter.

this identification we have modeled an MoD system (at least in the case without rebalancing) as a *closed queueing network with respect to the vehicles.* Note that the road queues are modeled as infinite-server queues as the model does not consider congestion effects. In Section 3.7 we discuss the potential to take into account congestion by modeling road queues as *finite*-server queues.

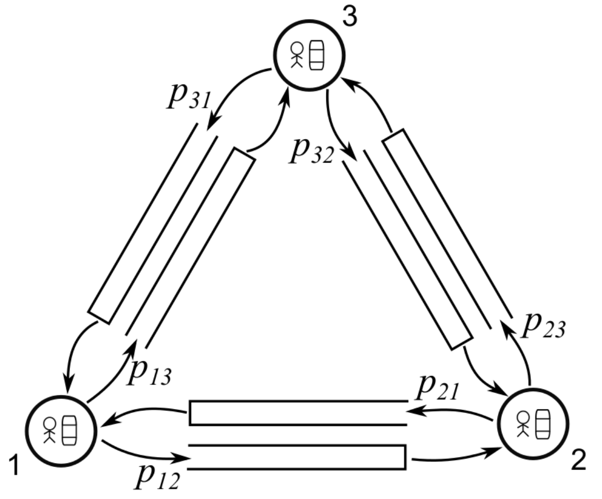


Figure 3.1: A 3-station network. Circles represent the SS nodes (stations) and rectangles represent the IS nodes (roads).

More formally, denote by S the set of single-server nodes and I the set of infinite-server nodes. Each station is mapped into an SS node, while each road is mapped into an IS node. The set of all nodes in the abstract queueing network is then given by $\mathcal{N} = S \cup I$. Since each SS node is connected to every other SS node, and since $p_{ii} = 0$ (hence the road node ii does not need to be represented), the number of nodes in the network is given by $N(N-1)+N = N^2$, in other words, $|\mathcal{N}| = N^2$. For each IS node $i \in I$, let Parent(i) and Child(i) be the origin and destination nodes of i, respectively. Vehicles in the abstract queueing network move between SS nodes and IS nodes according to the routing matrix $\{r_{ij}\}_{ij}$:

$$r_{ij} = \begin{cases} p_{il} & i \in S, j \in I \quad \text{where } i = \text{Parent}(j), l = \text{Child}(j), \\ 1 & i \in I, j \in S \quad \text{where } j = \text{Child}(i), \\ 0 & \text{otherwise}, \end{cases} \tag{3.1}$$

where the first case corresponds to a move from an SS node to an IS node (according to the destination selected by a customer), and the second case to a move from an IS node to the unique SS node corresponding to its destination. Furthermore, the service times at each node $i \in \mathcal{N}$ are exponentially distributed with service rates given by

$$\mu_i(n) = \begin{cases} \lambda_i & \text{if } i \in S, \\ n \cdot \mu_{jk} & \text{if } i \in I, j = \text{Parent}(i), k = \text{Child}(i), \end{cases} \tag{3.2}$$

where $n \in \{0, 1, \dots m\}$ is the number of vehicles at node i, and $\mu_{jk} = 1/T_{jk}$. The first case is the case where vehicles wait for customers at stations, while the second case is the case where vehicles spend an exponentially distributed travel time to move between stations (note that the IS nodes correspond to infinite-server queues, hence the service rate is proportional to the number of vehicles in the queue). As defined, the abstract queueing network is a closed Jackson network, and hence can be analyzed with the tools discussed in Chapter 2.1.4.

Assume, now, that we allow the vehicles to autonomously rebalance throughout the network. To include rebalancing while staying within the Jackson network framework, we focus on a particular class of stochastic rebalancing policies described as follows. Each station i generates "virtual passengers" according to a Poisson process with rate ψ_i, *independent* of the real passenger arrival process, and routes these virtual passengers to station j with probability α_{ij} (with $\sum_j \alpha_{ij} = 1$ and $\alpha_{ii} = 0$). As with real passengers, the virtual passengers are lost if the station is empty upon generation. Such class of rebalancing policies *encourages* rebalancing but does *not* enforce a rebalancing rate, which allows us to maintain tractability in the model.

One can then combine the real passenger arrival process with the virtual passenger process (assumed independent) using the independence assumption to form a model of the same form as the one described earlier in this section while taking into account vehicle rebalancing. Specifically, we consider the same set of SS nodes and IS nodes (since the transportation network is still the same). Let $\{A_t^{(i)}, t \geq 0\}$ be the total arrival process of real *and* virtual passengers at station $i \in S$, and denote its rate with $\tilde{\lambda}_i$. The process $A_t^{(i)}$ is Poisson since it is the superposition of two independent Poisson processes. Hence, the rate $\tilde{\lambda}_i$ is given by

$$\tilde{\lambda}_i = \lambda_i + \psi_i. \tag{3.3}$$

Equivalently, one can view the passenger arrival process and the rebalancing process as the result of Bernoulli splitting on $A_t^{(i)}$ with a probability p_i satisfying

$$\psi_i = p_i \tilde{\lambda}_i, \quad \lambda_i = (1 - p_i)\tilde{\lambda}_i. \tag{3.4}$$

Let us refer to passengers arriving according the the processes $\{A_t^{(i)}, t \geq 0\}$ as generalized passengers. The probability $\tilde{p}_{ij}$ that a generalized passenger arriving at station i selects a destination j is given by

$$\begin{aligned}\tilde{p}_{ij} &= \mathbb{P}\{i \to j \mid \text{virtual}\}\, p_i + \mathbb{P}\{i \to j \mid \neg\text{virtual}\}\,(1 - p_i) \\ &= \alpha_{ij} p_i + p_{ij}(1 - p_i),\end{aligned} \tag{3.5}$$

where $\mathbb{P}\{i \to j \mid \text{virtual}\}$ is the probability that a virtual passenger selects station j as its destination, and $\mathbb{P}\{i \to j \mid \neg\text{virtual}\}$ is the probability that a real passenger selects station j as its destination. One can then identify an AMoD system with rebalancing (for the specific class of rebalancing policies discussed above) with an abstract queueing network with routing matrix and service rates given, respectively, by Equations (3.1) and (3.2), where p_{il} is replaced by $\tilde{p}_{il}$, λ_i is replaced by $\tilde{\lambda}_i$, and r_{ij} is replaced by $\tilde{r}_{ij}$. In this way, the model is still a closed Jackson network model. As in Chapter 2.1.4, we define $\pi_i, i \in \mathcal{N}$ to be the relative throughput of node i, satisfying Equation (2.7) and γ_i to be the relative utilization of node i, given by Equation (2.9). For notational convenience, we order γ_i and π_i in such a way that the first N components correspond to the N stations (for example, γ_i corresponds to station i, or the ith SS node, where $i = 1, 2, \ldots, N$).

As already mentioned, in order to identify an AMoD system with rebalancing with a Jackson queueing model, we restricted the class of rebalancing policies to open-loop, "rebalancing promoting" policies. We will consider *closed-loop* policies in Section 3.3.3.

3.2.2 Problem formulation

Within our model, the optimization variables are the rebalancing rates ψ_i and α_{ij} of the rebalancing promoting policies. One might wonder in the first place if and when rebalancing is even required. Indeed, one can easily obtain that, for the case *without* rebalancing [30], $\lim_{m\to\infty} A_i(m) = \gamma_i/\gamma_S^{\max}$, for all $i \in S$, where γ_i is the relative utilization of node $i \in S$, $A_i(m)$ is the availability of vehicles at node $i \in S$ (as defined in Equation (2.10)) and $\gamma_S^{\max} := \max_{i\in S} \gamma_i$. Hence, as m approaches infinity, the set of stations $B := \{i \in S : \gamma_i = \gamma_S^{\max}\}$ can have availability arbitrarily close to 1 while all other stations have availability *strictly* less than 1 *regardless* of m. In other words, without rebalancing, an MoD system will always experience passenger losses no matter how many vehicles are employed!

The above discussion motivates the need for rebalancing. The tenet of our approach is to ensure, through rebalancing, that the network is (on average) in balance, i.e., $A_i(m) = A_j(m)$ for all $i, j \in S$ (or, equivalently, $\gamma_i = \gamma_j$ for all $i, j \in S$, as implied by Equation (2.10)). The motivation behind this philosophy is twofold (1) it provides a natural notion of service fairness, and (2) it fulfills the intuitive condition that as m goes to $+\infty$ the availability of *each station* goes to one (since in this case $\gamma_i = \gamma_S^{\max}$ for all i in S). The objective then is to manipulate the rebalancing rates α_{ij} and ψ_i such that all the γ_i's in S are equal while minimizing the number of rebalancing vehicles on the road. Note that the average number of rebalancing vehicles traveling between station nodes i and j is given by $T_{ij}\alpha_{ij}\psi_i$. The rebalancing problem we wish to solve is then as follows:

Optimal Rebalancing Problem (ORP): Given an AMoD system modeled as a closed Jackson network, solve

$$\begin{aligned}
\underset{\psi_i,\alpha_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\alpha_{ij}\psi_i \\
\text{subject to} \quad & \gamma_i = \gamma_j \\
& \sum_j \alpha_{ij} = 1 \\
& \alpha_{ij} \geq 0,\ \psi_i \geq 0 \qquad i,j \in S,
\end{aligned} \tag{3.6}$$

where

$$\gamma_i = \frac{\pi_i}{\tilde{\mu}_i} = \frac{\pi_i}{\lambda_i + \psi_i}$$

and π_i satisfies Equation (2.7).

Note that to solve the ORP one would need to explicitly compute the relative throughputs π's using the balance equations (2.7). This involves finding the 1-dimensional null space of a $\mathbb{R}^{N^2 \times N^2}$ matrix, which becomes computationally expensive as the number of stations become large. Furthermore, the objective function and the constraints $\gamma_i = \gamma_j$ are nonlinear in the optimization variables. In the next section we show how to reduce the dimension of the problem to $\mathbb{R}^N$ and how the ORP can be readily solved as a minimum cost flow problem.

3.3 Optimal Rebalancing

3.3.1 Optimal rebalancing

In this section, we show how the ORP can be readily solved as a minimum cost flow problem. The main result of this section is presented in Theorem 3.3.3. We first present two lemmas that will be key in the proof of the theorem. The first lemma shows how the balance equations (2.7) can be written only in terms of the SS nodes.

Lemma 3.3.1 (Folding of balance equations)**.** *Consider an AMoD system modeled as a closed Jackson network as described in Section 3.2.1. Then the relative throughputs π's for the SS nodes can be found by solving the* reduced *balance equations*

$$\pi_i = \sum_{k \in S} \pi_k \tilde{p}_{ki} \quad \textit{for all } i \in S, \tag{3.7}$$

where SS nodes are considered in isolation. The π's for the IS nodes are then given by

$$\pi_i = \pi_{\text{Parent}(i)}\, \tilde{p}_{\text{Parent}(i)\text{Child}(i)} \quad \textit{for all } i \in I. \tag{3.8}$$

Proof. For each node $i \in \mathcal{N}$ Equation (2.7) can be separated into SS nodes and IS nodes as follows

$$\pi_i = \sum_{j\in\mathcal{N}} \pi_j \tilde{r}_{ji} = \sum_{j\in S} \pi_j \tilde{r}_{ji} + \sum_{j\in I} \pi_j \tilde{r}_{ji}.$$

Consider, first, a SS node, i.e., consider i in S. Then one can write,

$$\pi_i = \underbrace{\sum_{j\in S} \pi_j \tilde{r}_{ji}}_{=0} + \sum_{j\in I} \pi_j \tilde{r}_{ji} = \sum_{\substack{j\in I\\ i=\text{Child}(j)}} \pi_j,$$

where $\sum_{j\in S} \pi_j \tilde{r}_{ji} = 0$ since SS nodes are connected exclusively by IS nodes. The last equality follows from the fact that whenever a child node of an IS node j is the SS node i, then $\tilde{r}_{ji} = 1$.

Consider, now, an IS node, i.e., consider i in I. Let Parent$(i) = k$ and Child$(i) = l$. Then one can write,

$$\pi_i = \sum_{j\in S} \pi_j \tilde{r}_{ji} + \underbrace{\sum_{j\in I} \pi_j \tilde{r}_{ji}}_{=0} = \pi_k \tilde{p}_{kl},$$

where $\sum_{j\in I} \pi_j \tilde{r}_{ji} = 0$ since IS nodes are connected *exclusively* to SS nodes, and the second equality follows from the fact that a single SS node feeds into each IS node with probability $\tilde{p}_{kl}$. This proves the second claim.

Collecting the results so far, we obtain, for each i in S,

$$\pi_i = \sum_{\substack{j\in I\\ i=\text{Child}(j)}} \pi_j = \sum_{\substack{j\in I\\ i=\text{Child}(j)}} \pi_{\text{Parent(j)}}\ \tilde{p}_{\text{Parent(j)}\,i} = \sum_{k\in S} \pi_k \tilde{p}_{ki},$$

which proves the first claim. □

Lemma 3.3.2. *For any rebalancing policy $\{\psi_i\}_i$ and $\{\alpha_{ij}\}_{ij}$, it holds for all $i \in S$*

1. $\gamma_i > 0$,

2. $(\lambda_i + \psi_i)\gamma_i = \sum_{j\in S} \gamma_j(\alpha_{ji}\psi_j + p_{ji}\lambda_j)$.

Proof. Let us prove the first part of the lemma. By assumption, the probabilities $\{p_{ij}\}_{ij}$ constitute an irreducible Markov chain. By Equation (3.5), the probabilities $\{\tilde{p}_{ij}\}_{ij}$ lead to an irreducible Markov chain as well. The π vector satisfying Equation (3.7) is the steady state distribution for the transition probabilities $\{\tilde{p}_{ij}\}_{ij}$ and by the Perron-Frobenius theorem, it is positive [62, p. 673]. In other words, $\pi_i > 0$ for all $i \in S$. By the definition of the relative utilizations γ_i (see Equation (2.9)), we obtain the first part of the claim.

Let us now consider the second part of the lemma. Recall that, by assumption, $p_{ii} = 0$ and $\alpha_{ii} = 0$. By Lemma 3.3.1, for any $i \in S$, one can write

$$\begin{aligned}
\pi_i &= \sum_{j \in S} \pi_j \tilde{p}_{ji} \\
&= \sum_{j \in S} \pi_j \Big(\alpha_{ji} p_j + p_{ji}(1 - p_j) \Big) \\
&= \sum_{j \in S} \pi_j \Big(\alpha_{ji} \frac{\psi_j}{\lambda_j + \psi_j} + p_{ji} \frac{\lambda_j}{\lambda_j + \psi_j} \Big) \\
&= \sum_{j \in S} \frac{\pi_j}{\lambda_j + \psi_j} (\alpha_{ji}\psi_j + p_{ji}\lambda_j) \\
&= \sum_{j \in S} \gamma_j (\alpha_{ji}\psi_j + p_{ji}\lambda_j),
\end{aligned}$$

where the second equality follows from Equation (3.5), the third equality follows from Equation (3.4), and the last equality follows from Equations (2.9), (3.2), and (3.3). This concludes the proof. □

The next theorem (which represents the main result of this section) shows that we can *always* solve problem ORP by solving a low dimensional linear optimization problem.

Theorem 3.3.3 (Solution to problem ORP)**.** *Consider the linear optimization problem*

$$\begin{aligned}
\underset{\beta_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\beta_{ij} \\
\text{subject to} \quad & \sum_{j \neq i} (\beta_{ij} - \beta_{ji}) = -\lambda_i + \sum_{j \neq i} p_{ji}\lambda_j \\
& \beta_{ij} \geq 0
\end{aligned} \tag{3.9}$$

The optimization problem (3.9) *is always feasible. Let* $\{\beta^*_{ij}\}_{ij}$ *denote an optimal solution. By setting* $\psi_i = \sum_{j \neq i} \beta^*_{ij}$*,* $\alpha_{ii} = 0$*, and, for* $j \neq i$*,*

$$\alpha_{ij} = \begin{cases} \beta^*_{ij}/\psi_i & \text{if } \psi_i > 0 \\ 1/(N-1) & \text{otherwise,} \end{cases}$$

one obtains an optimal solution to problem ORP.

Proof. First, we note that problem (3.9) is an uncapacitated minimum cost flow problem and thus is always feasible. Consider an optimal solution to problem (3.9), $\{\beta^*_{ij}\}_{ij}$, and set $\{\psi_i\}_i$ and $\{\alpha_{ij}\}_{ij}$ as in the statement of the theorem. We want to show that, with this choice, $\{\psi_i\}_i$ and $\{\alpha_{ij}\}_{ij}$ represent an optimal solution to the ORP. Since $\{\beta^*_{ij}\}_{ij}$ is an optimal solution to problem (3.9), then one easily concludes that $\{\psi_i\}_i$ and $\{\alpha_{ij}\}_{ij}$ are an optimal solution to problem

$$\begin{aligned} \underset{\psi_i,\alpha_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\alpha_{ij}\psi_i \\ \text{subject to} \quad & \lambda_i + \psi_i = \sum_j \alpha_{ji}\psi_j + p_{ji}\lambda_j \\ & \sum_j \alpha_{ij} = 1 \\ & \alpha_{ij} \geq 0, \quad \psi_i \geq 0 \end{aligned} \tag{3.10}$$

The objective is now to show that the constraint

$$\lambda_i + \psi_i = \sum_j \alpha_{ji}\psi_j + p_{ji}\lambda_j \tag{3.11}$$

is equivalent to the constraint

$$\gamma_i = \gamma_j. \tag{3.12}$$

Consider, first, the case where the $\{\alpha_{ij}\}_{ij}$ and $\{\psi_i\}_i$ satisfy Constraint (3.11). Then, considering Lemma 3.3.2, one can write, for all i,

$$\Big(\sum_j \alpha_{ji}\psi_j + p_{ji}\lambda_j\Big)\gamma_i = \sum_{j\in S} \gamma_j(\alpha_{ji}\psi_j + p_{ji}\lambda_j). \tag{3.13}$$

Let $\varphi_{ij} := \alpha_{ji}\psi_j + p_{ji}\lambda_j$ and $\zeta_{ij} := \varphi_{ij}/\sum_j \varphi_{ij}$. (Note that $\sum_j \varphi_{ij} = \lambda_i + \psi_i > 0$ as $\lambda_i > 0$ by assumption.) Since $\alpha_{ii} = 0$ and $p_{ii} = 0$, one has $\zeta_{ii} = 0$. The variables $\{\zeta_{ij}\}_{ij}$'s can be considered as transition probabilities of an *irreducible* Markov chain (since, by assumption, the probabilities $\{p_{ij}\}_{ij}$ constitute an irreducible Markov chain). Then, one can rewrite Equation (3.13) as $\gamma_i = \sum_j \gamma_j \zeta_{ij}$, which can be rewritten in compact form as $Z\gamma = \gamma$, where $\gamma = (\gamma_1, \ldots, \gamma_N)^T$ and Z is an irreducible, row stochastic matrix whose ith row is given by $[\zeta_{i1}, \zeta_{i2}, \ldots, \zeta_{i\,i-1}, 0, \zeta_{i\,i+1} \cdots \zeta_{iN}]$, where $i = 1, 2, \ldots, N$. Since Z is an irreducible, row stochastic matrix, by the Perron-Frobenius theorem [62, p. 673], the eigenspace associated with the eigenvalue equal to 1 is one-dimensional, which implies that the equation $Z\gamma = \gamma$ has a unique solution given by $\gamma = (1, \ldots, 1)^T$, up to a scaling factor. This shows that $\gamma_i = \gamma_j$ for all i, j. Conversely, assume that $\{\alpha_{ij}\}_{ij}$ and $\{\psi_i\}_i$ satisfy Constraint (3.12). Considering Lemma 3.3.2 (note, in particular, that $\gamma_i > 0$), since $\gamma_i = \gamma_j$ for all i, j, then one immediately obtains that $\{\alpha_{ij}\}_{ij}$ and $\{\psi_i\}_i$ satisfy Constraint (3.11). Hence, we can equivalently restate problem (3.10) as problem (3.6), which proves the claim. □

The importance of Theorem 3.3.3 is twofold: it allows us to efficiently find an optimal open-loop, rebalancing promoting policy, and it enables the computation of quality of service metrics (namely, vehicle availability) for AMoD systems as shown next.

3.3.2 Computation of performance metrics

By leveraging Theorem 3.3.3, one can readily compute performance metrics (i.e. vehicle availability) for an AMoD system. First, we compute an optimal solution to the ORP using Theorem 3.3.3, which involves solving a linear optimization problem with N^2 variables. Next, we compute the relative throughputs π's using Lemma 3.3.1. Finally, we apply mean value analysis (MVA) [63] (see Chapter 2.1.4) in order to avoid the explicit computation of the normalization constant in Equation (2.10), which is prohibitively expensive for large numbers of vehicles and stations. Let $W_i(n)$ be the mean waiting time and $L_i(n)$ be the mean queue length at each node i. For the Jackson model in Section 3.2.1, subject to the initial conditions $W_i(0) = L_i(0) = 0$, the equations for MVA read as:

- $W_i(n) = \frac{1}{\mu_i(1)} = T_{\text{Parent}(i)\,\text{Child}(i)}$ for all $i \in I$,
- $W_i(n) = \frac{1}{\mu_i}(1 + L_i(n-1)) = \frac{1}{\tilde{\lambda}_i}(1 + L_i(n-1))$ for all $i \in S$,
- $L_i(n) = \frac{n\pi_i W_i(n)}{\sum_{j\in\mathcal{N}} \pi_j W_j(n)}$ for all $i \in \mathcal{N}$.

Finally, the throughput (or mean arrival rate of vehicles) to each station is given by Little's theorem (see Chapter 2.1.3):

$$\Lambda_i(m) = \frac{L_i(m)}{W_i(m)} \qquad \text{for all } i \in S.$$

Combining Equations (2.9), (2.8) and (2.10), one readily obtains the availability at each station as

$$A_i(m) = \frac{\Lambda_i(m)}{\tilde{\lambda}_i}.$$

This procedure scales well to a large number of stations and vehicles, and is applied in Section 3.4 to real-world settings involving hundreds of stations and thousands of vehicles, to assess the potential performance of an AMoD system in New York City.

The rebalancing promoting policy considered so far, while providing useful insights into the performance and operations of an AMoD system, is ultimately an open-loop policy and hence of limited applicability. In the next section, we use insights gained from the ORP to formulate a *closed-loop* rebalancing policy for the robotic vehicles that appears to perform well in practice.

3.3.3 Real-time rebalancing policy

In this section, we introduce a practical real-time rebalancing policy that can be implemented on real AMoD systems. In reality, customers arriving at a station would wait in line rather than leave the system immediately (as in the loss model) if a vehicle is not available. In the meantime, information could be collected about the customer's destination and used in the rebalancing process. Let $v_i^{\text{own}}(t)$

be the number of vehicles "owned" by station i, that is, vehicles that are at station i, on their way to station i, or will be on their way to station i. We can write

$$v_i^{\text{own}}(t) = v_i(t) + \sum_j v_{ji}(t) + \sum_j \tilde{c}_{ji}(t),$$

where $v_i(t)$ is the number of vehicles at station i, $v_{ji}(t)$ is the number of vehicles enroute from station j to i, and $\tilde{c}_{ji}$ is the number of passengers at station j that are about to board an available vehicle to station i. Note that $\sum_i \tilde{c}_{ji}(t) \leq v_j(t)$. The number of excess vehicles station i will possess is given by

$$v_i^e(t) = v_i^{\text{own}}(t) - c_i(t),$$

where $c_i(t)$ is the number of customers at station i. The total number of excess vehicles is given by

$$\begin{aligned}\sum_i v_i^e(t) &= \sum_i \Big(v_i(t) + \sum_j v_{ji}(t) + \sum_j \tilde{c}_{ji}(t) - c_i(t) \Big) \\ &= m + \sum_i \min\{v_i(t), c_i(t)\} - \sum_i c_i(t) \\ &= m - \sum_i \max\{c_i(t) - v_i(t), 0\}.\end{aligned} \tag{3.14}$$

The second equality replaces $v_i(t) + \sum_i v_{ji}(t)$ with the total number of vehicles and asserts that in the current time step, either all of the customers or all the vehicles will leave the station. The last equality is obtained by considering both cases when $c_i(t) \geq v_i(t)$ and when $c_i(t) < v_i(t)$.

Through rebalancing, we may wish to distribute these excess vehicles evenly between all the stations, in which case each station will have no less than $v_i^d(t)$ vehicles, given by

$$v_i^d(t) = \left\lfloor \frac{(m - \sum_i \max\{c_i(t) - v_i(t), 0\})}{N} \right\rfloor.$$

Accordingly, every $t_{\text{hor}} > 0$ time periods, the number of vehicles to rebalance from station i to j, n_{ij}^v, is computed by solving the linear integer optimization problem

$$\begin{aligned} \underset{n_{ij}^v}{\text{minimize}} \quad & \sum_{i,j} T_{ij} n_{ij}^v \\ \text{subject to} \quad & v_i^e(t) + \sum_{j \neq i} (n_{ji}^v - n_{ij}^v) \geq v_i^d(t) \ \text{ for all } i \in S \\ & n_{ij}^v \in \mathbb{N} \ \text{ for all } i, j \in S, i \neq j \end{aligned} \tag{3.15}$$

This rebalancing policy takes all the current information known about the system and sets the rebalancing rates (in this case, the number of rebalancing vehicles) so that excess vehicles are distributed evenly to all the stations. This is in part inspired by the optimization problem in Theorem 3.3.3. It can be shown that the constraint matrix is totally unimodular, and the problem can be solved as a linear program, as the resulting solution will be necessarily integer-valued (see Chapter 2.2.1). The rebalancing policy presented here is closely related to the one presented in [21], the main difference being the inclusion of the current customers in line within the optimization process.

The real-time rebalancing policy will be used in Section 3.4 to validate the vehicle availability performance criterion.

3.4 Case Study: AMoD in Manhattan

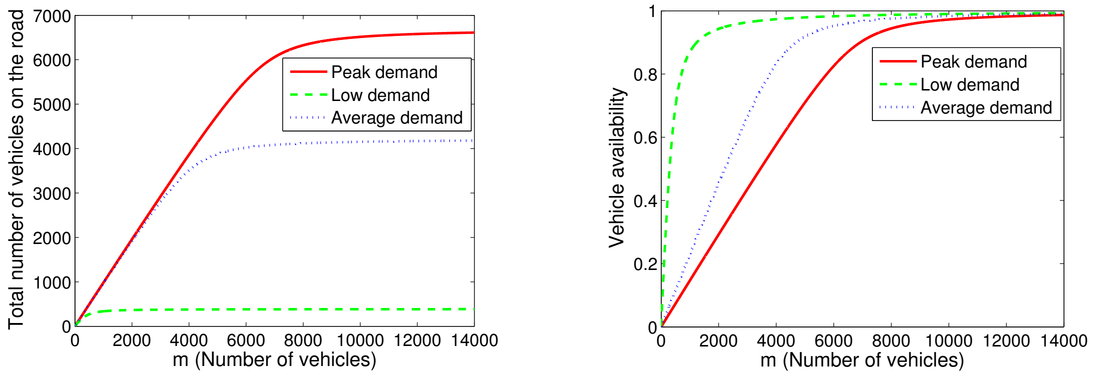
In this section we apply our availability analysis using the loss model to see how many robotic vehicles in an AMoD system would be required to replace the current fleet of taxis in Manhattan while providing quality service at current customer demand levels. We then discuss the impact of the placement/number of stations on the case study and ways to further improve the performance of the system.

3.4.1 Case study overview and results

In 2012, over 13,300 taxis in New York City made over 15 million trips a month or 500,000 trips a day, with around 85% of trips within Manhattan. This study uses taxi trip data collected on March 1, 2012[2] consisting of 439,950 trips within Manhattan. First, trip origins and destinations are clustered into $N = 100$ stations throughout the city using k-means clustering. The resulting locations of the stations are such that a demand is on average less than 300m from the nearest station, or approximately a 3-minute walk. An average speed is estimated for each hour of the day based on reported trip distance and reported trip time. Only trips where the reported distance was approximately equal to the Manhattan distance were used for speed estimation (to avoid trips with detours). The system parameters λ_i, p_{ij}, and T_{ij} are estimated for each hour of the day using trip data between each pair of stations with Laplace smoothing. Some congestion effects are implicitly taken into account in the computation of T_{ij}, which uses the Manhattan distance and the estimated average speed.

Vehicle availability is calculated for 3 cases - peak demand (29,485 demands/hour, 7-8pm), low demand (1,982 demands/hour, 4-5am), and average demand (16,930 demands/hour, 4-5pm). For each case, vehicle availability is calculated as a function of the fleet size using MVA techniques. In addition to vehicle availability, the average number of vehicles on the road is given by $\sum_{i,j} \Lambda_i \tilde{p}_{ij} T_{ij}$ by Little's theorem. The results are summarized in Figure 3.2. From Figure 3.2(a), we see that with 7,000 vehicles, the average number of vehicles traveling is 6,060, corresponding to a vehicle utilization rate of 86%. This utilization rate drops off significantly with more than 8,000 vehicles in the system, as the number of vehicles on the road levels off. This suggests that the operating point

[2]The data is courtesy of the New York City Taxi & Limousine Commission.



(a) (b)

Figure 3.2: 3.2(a): Average number of vehicles on the road as a function of system size for 100 stations in Manhattan. 3.2(b): Vehicle availability as a function of system size. Availability and number of vehicles on the road are calculated for peak demand (7-8pm), low demand (4-5am), and average demand (4-5pm).

of the system should be between 7,000 and 8,000 vehicles at peak demand. From Figure 3.2(b), for high vehicle availability (say, 95%), we would need around 8,000 vehicles (∼70% of the current fleet size operating in Manhattan[3]) at peak demand and 6,000 vehicles at average demand. The results suggest that an AMoD system with 8,000 vehicles would be able to meet 95% of the taxi demand in Manhattan, assuming 5% of passengers are impatient and are lost when a vehicle is not immediately available. However, in a real system, passengers would wait in line for the next vehicle rather than leave the system, thus it is important to determine how vehicle availability relates to customer wait times. We characterize the customer wait times through simulation, using the real-time rebalancing policy described in Section 3.3.3. Figure 3.4 shows a snapshot of the simulation environment with 100 stations and 8,000 vehicles. Simulation are performed with discrete time steps of 2 seconds and a simulation time of 24 hours. The time-varying system parameters λ_i, p_{ij}, and average speed are piecewise constant, and change each hour based on values estimated from the taxi data. Travel times T_{ij} are based on average speed and Manhattan distance between i and j, and rebalancing is performed every 15 minutes. Three sets of simulations are performed for 6,000, 7,000, and 8,000 vehicles, and the resulting average waiting times are shown in Figure 3.3.

Figure 3.3 shows that for a 7,000 vehicle fleet, the peak averaged wait time is less than 5 minutes (9-10am) and for 8,000 vehicles, the average wait time is only 2.5 minutes. The simulation results show that high availability (90-95%) does indeed correspond to low customer waiting time and that an AMoD system with 7,000 to 8,000 vehicles (60-70% of the size of the current taxi fleet) can provide adequate service with current taxi demand levels in Manhattan.

[3]we approximate the taxi fleet operating in Manhattan as 85% of the total New York City taxi fleet.

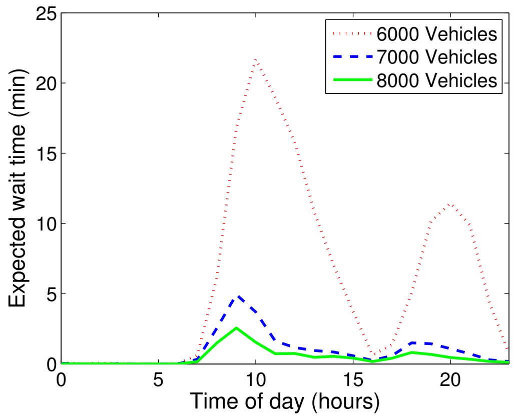


Figure 3.3: Average customer wait times over the course of a day, for systems of different sizes.

3.4.2 Placement and number of stations

In the previous analysis, the locations of the stations were determined by a k-means clustering algorithm based on historical demand data. We can evaluate the adequacy of these locations by looking at the waiting time distribution throughout the city. Figure 3.5(a) shows the distribution of waiting time for 100 stations and 8,000 vehicles at peak demand (between 9 and 10 am). For most stations, the wait times are very low (0 – 2 minutes) but a few stations have wait times much higher than the average. This is in part due to the geographic placement of the stations and in part due to the tuning of the real-time rebalancing policy. For example, in Figure 3.4, the station just northwest of central park sees high customer demand but is geographically isolated from midtown and lower Manhattan, where most excess vehicles are. During the rebalancing optimization (3.15), the number of rebalancing vehicles sent to this station takes into account the number of vehicles that will eventually reach the station, but does not take into account the length of time it will take to get there. Due to the longer travel time, there is an effective delay for rebalancing vehicles to reach the station, resulting in longer wait times. This delay can be reduced by tuning the parameters $v_i^d(t)$ to send more rebalancing vehicles to the few isolated stations with high customer demand.

The number of stations, n, may also impact the waiting time distribution. To gain insight into the effects of different n, additional simulations were performed with 8,000 vehicles at peak demand (9–10 am). We studied a range of values for n from 50 to 200. For each n, the stations were placed using k-means, and 10 simulations were performed. Figure 3.5(b) shows the maximum wait time across all stations for different n values. It is clear that a higher number of stations tends to yield a lower maximum wait time, due to the fact that vehicles are more distributed and rebalancing vehicles would not need to travel as far. However, as previously stated, the locations of the stations are just as important as the number of stations, and can significantly impact the maximum wait times. The spike in wait time at 120 stations in Figure 3.5(b) can be attributed to poor placement

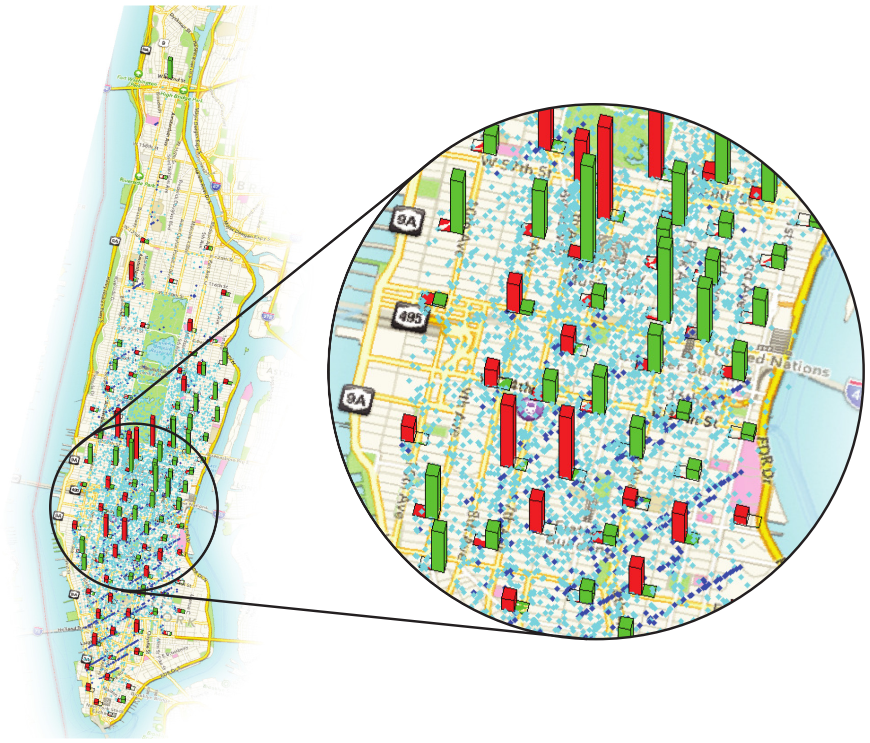
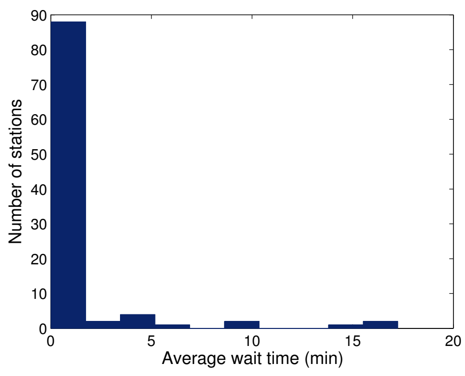


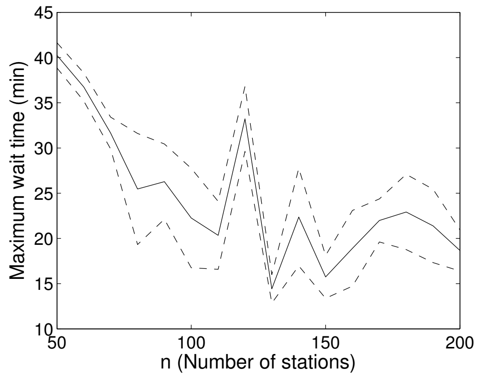
Figure 3.4: Simulation environment with 100 stations in Manhattan. Red bars indicate waiting customers, green bars indicate available vehicles, cyan dots are vehicles traveling with passengers and blue dots are rebalancing vehicles.

of the stations (a local optimum in the k-means algorithm). Overall, this analysis illustrates that the customer wait times can serve as a metric to determine the optimal number of stations and the locations of the stations.

In practice, an AMoD system may not necessarily need fixed infrastructure to serve as stations. Indeed, in a system where customer requests are made through a smartphone app and vehicles provide door-to-door service, the concept of a "station" may simply be a geographical region where vehicles in the vicinity service requests that arrive within the region. Hence, the number of "stations" and their locations may be changed to dynamically adapt to the current demand distribution. Demand prediction can be achieved with Bayesian non-parametric techniques such as Dirichlet process mixture models [64, 65]. In this way, the system can simultaneously predict demand and route vehicles to provide service, which can be an effective way of handling time-varying customer demands. This is an interesting avenue of future work.



(a)



(b)

Figure 3.5: 3.5(a) shows a histogram of the wait times of all 100 stations. 3.5(b) shows the maximum wait time in the system for systems with different n (number of stations). For 16 values of n ranging from 50 to 200, the simulation (8,000 vehicles, peak demand) was performed 10 times. The solid line denotes the mean of the 10 runs and the dashed lines denote one standard deviation above and below the mean.

3.5 Case Study: AMoD in Singapore

We apply the same analysis as Section 3.4 to a case study of Singapore, which considers an hypothetical deployment of an AMoD system in Singapore to replace its *entire* transportation infrastructure [22]. This case study should be interpreted as a thought experiment to investigate the potential capability of an AMoD solution to service all mobility demands within a city.

The study uses three sources of real-world data: (1) the Singapore Household Interview Travel Survey (HITS), (2) GPS data from Singapore Taxis, and (3) The Singapore road network. The HITS data was used to determine trip origins and destinations and demand rates. The GPS data was used to calculate average speeds of the vehicles. Finally, the road network was used to determine the ratio between the average trip distance (along the road network) and the Euclidean distance between origins and destinations. Using these data sources, the parameters of the Jackson network model (λ_i, p_{ij}, and T_{ij}) are computed.

As in the New York study, vehicle availability is analyzed in two representative cases. The first is chosen as the 2-3 pm bin, since it is the one that is the closest to the “average” traffic condition. The second case considers the 7-8 am rush-hour peak. Results are summarized in Figure 3.6(a). With about 200,000 vehicles availability is about 90% on average, but drops to about 50% at peak times. With 300,000 vehicles in the fleet, availability is about 95% on average and about 72% at peak times.

As in Section 3.4, waiting times are characterized through simulation. For 250,000 vehicles, the maximum wait times during peak hours is around 30 minutes, which is comparable with typical

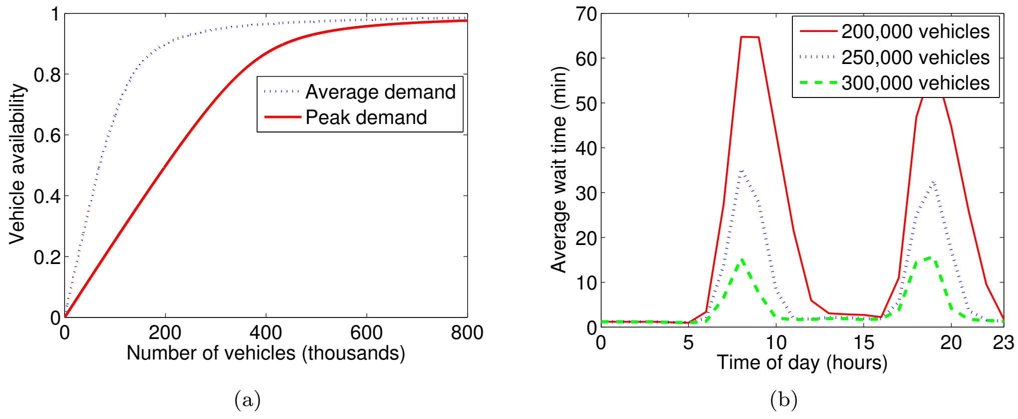


Figure 3.6: 3.6(a): Performance curve for Singapore with 100 stations, showing the availability of vehicles vs. the size of the system for both average demand (2-3 pm) and peak demand (7-8 am). 3.6(b): Average wait times over the course of a day, for systems of different sizes.

congestion delays during rush hour. With 300,000 vehicles, peak wait times are reduced to less than 15 minutes. To put these numbers into perspective, in 2011 there were 779,890 passenger vehicles operating in Singapore [66]. Hence, this case study suggests that an AMoD system can meet the personal mobility needs of the entire population of Singapore with a number of robotic vehicles that is less than 40% of the current number of passenger vehicles.

3.6 Hardware Experiments

To further compare the availability metric from the queueing model with customer wait times in a real system, a small-scale testbed was developed consisting of mobile robots driving in a "mock" city with 4 stations (see Figure 3.7). Customers are generated by a central computer according to a Poisson process with rate λ_i and routing distribution p_{ij}, $i, j \in \{1, 2, 3, 4\}$. Once generated, customers form first-come-first-serve queues at each station to wait for vehicles (as opposed to leaving immediately if a vehicle is not at the station). The vehicles are modified Pololu m3pi line-following robots, and communicate with the central computer through a shared WiFi network. When a customer arrives at a station, a vehicle takes the customer to the desired destination by following the black lines along a pre-programmed path. A Vicon motion capture system is used to implement collision avoidance between the vehicles, which (1) allows vehicles to stop behind a stopped vehicle traveling in the same direction, and (2) allow vehicles to traverse intersections using a first-in-first-out rule. Each vehicle takes around 15 seconds to travel to an adjacent station and 22 seconds to travel to the station on the opposite corner of the city. Eight vehicles were used in the experiment.

The experiments were performed by first randomly generating a demand intensity λ^0 and routing distribution p^0_{ij} that yielded a stable system in terms of customer wait times in simulation under the real-time rebalancing policy. Three runs were performed with demand intensity at $0.5\lambda^0$, λ^0, and $2\lambda^0$, while the routing distribution was held fixed. The real-time rebalancing policy in Section 3.3.3 was implemented with a time horizon of 2 seconds. Figure 3.8(a) shows the corresponding availability plots for reference (if the same demands were applied to the loss model) and Figure 3.8(b) shows the number of waiting customers in each run. The number of idle vehicles is shown in Figure 3.8(c). In these experiments, the high demand run (corresponding to 57% availability) was unstable, as the customers arrived faster than they can be serviced. The medium demand case (corresponding to 66% availability) had on average one customer waiting at one of the four stations, with an average wait time of 6.4 seconds. The low demand case (corresponding to 69% availability) only had an average of 0.07 waiting customers with an average wait time of less than 1 second. It is interesting to note that the low demand run had very low wait times, but its corresponding availability in the loss model was only 69%. This reveals a limitation of the loss model, where rebalancing is implemented by *randomly* generating virtual customers according to a Poisson process. The real-time rebalancing policy only rebalances vehicles when needed, and hence is able to provide a better quality of service (low wait times). Finally, we see from Figure 3.8(c) that in the medium demand case, the average number of idle vehicles is only 1.75, which corresponds to a 78% vehicle utilization rate. This high utilization rate suggests that the system is near its stability limit and is comparable to that of the Manhattan analysis (see Figure 3.2(a)) at peak demand with 8,000 vehicles.

It is interesting to see the emergence of traffic congestion due to vehicle-to-vehicle interactions in the experiments. With eight vehicles, some interactions were present, which created small travel delays on the road. Future work on the testbed will focus on increasing the number of vehicles (20+ vehicles) so that congestion effects can be visualized and studied.

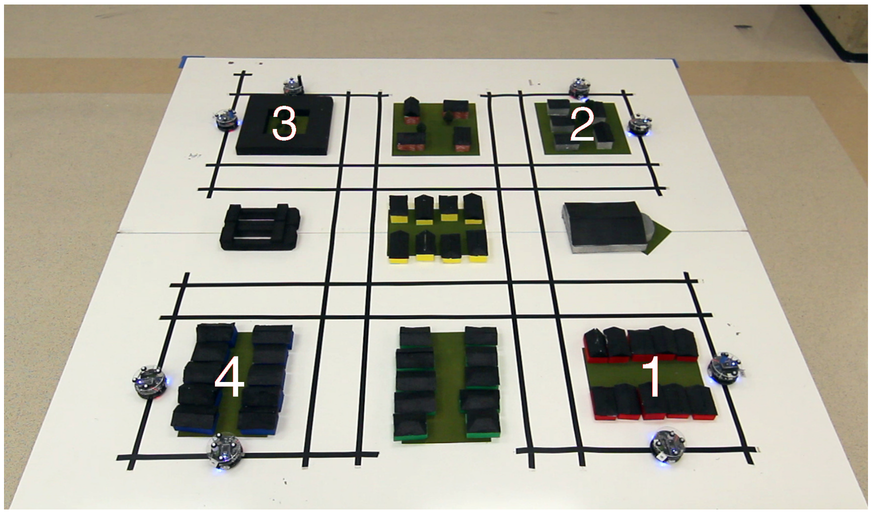
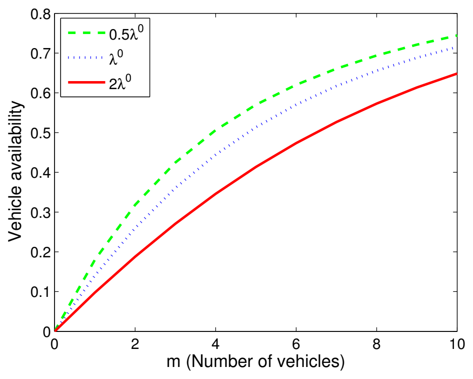
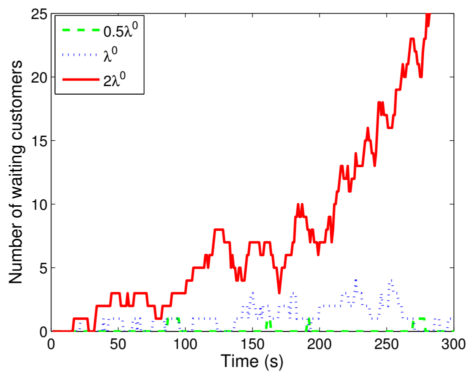


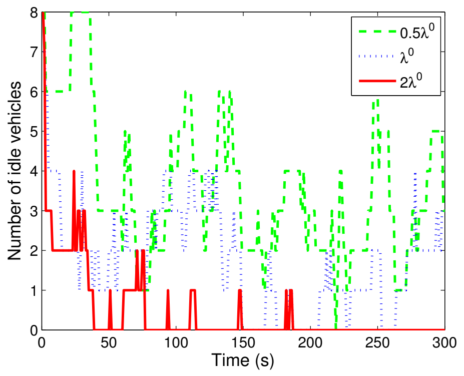
Figure 3.7: Hardware testbed with 4 stations and 8 vehicles.



(a)



(b)



(c)

Figure 3.8: 3.8(a) shows the availability curves for the three test cases. 3.8(b) shows the total number of customers waiting at the four stations for the three test cases. 3.8(c) shows the total number of idle vehicles at the four stations.

3.7 Queueing network modeling of congestion effects

The queueing model described in Section 3.2 does not consider congestion effects (roads are modeled as *infinite* server queues, so the travel time for each vehicle is independent of all other vehicles). However, if too many rebalancing vehicles travel on a route that is already congested, they can cause a traffic jam and decrease throughput in the entire system. Hence, in some scenarios, adding robotic vehicles to improve the quality of service might indeed have the opposite effect. This problem is explored in depth in Chapter 5. Here, we discuss approaches to study congestion effects that specifically leverage our queueing-theoretical setup.

A first approach is to change the infinite-server road queues to queues with a *finite* number of servers, where the number of servers on each road represents the *capacity* of that road. This road congestion model is similar to "vertical queueing" models that have been used in congestion analysis for stop-controlled intersections [67] and for traffic assignment [25]. In classical traffic flow theory [68], the flow rate of traffic increases with the density of vehicles up to a critical value at which point the flow decreases, marking the beginning of a traffic jam. By letting the number of servers represent the critical density of the road, the queueing model becomes a good model for traffic flow up to the point of congestion.

Remarkably, the Jackson network model can be extended to the case where roads are modeled as finite-server queues. Furthermore, the results in Section 2.1.4 are equally valid and an extended version of the MVA algorithm for finite-server queues can be applied [63]. However, the travel times are no longer simply equal to the inverse of the service rates of the road queues, which significantly complicates the formulation of an analogue of optimal rebalancing problem. Another problem associated with this approach is the mapping of from road queues to a physical road network.

The main difficulty in mapping the capacities of the road network into the number of servers of the queueing model (or "virtual" capacities, denoted by q_{ij}) is that trips from different origins and destinations may share the *same* physical road.

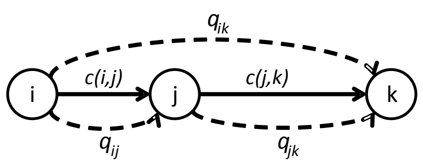


Figure 3.9: A simple 3-station example showing the procedure of mapping physical roads into finite-server road queues.

As a simple example, consider the 3-station network shown in Figure 3.9. Let $c(i, j)$ represent the maximum number of vehicles that can travel on the road between station i and station j without causing significant congestion. Let q_{ij}, the number of servers between i and j, represent the number of vehicles that can travel between i and j before delays occur due to queueing. In the simple network, to go from station i to station k, one must pass through station j. Hence, one has the following consistency constraints

$$q_{ij} + q_{ik} \leq c(i, j), \quad q_{jk} + q_{ik} \leq c(j, k). \tag{3.16}$$

To maximize the overall road usage, we can define a quadratic objective that seeks to minimize the difference between the real road capacities and the sum of the virtual road capacities:

$$\min_{q_{ij}, q_{jk}, q_{ik}} (q_{ij} + q_{ik} - c(i, j))^2 + (q_{jk} + q_{ik} - c(j, k))^2.$$

However, this optimization problem, along with the Constraints (3.16), does not yield a unique solution because nothing is assumed about the relative usage rates of the road queues. If relative road usage is known, the q_{ij}'s can be assigned proportional to the amount of traffic between each pair of stations that use the road. Let π_{ij} be the relative throughput of the road queue between station i and j, consistent with the earlier definition. *Heuristically*, the throughputs $\{\pi_{ij}\}_{ij}$ may be obtained from the arrival rates and travel patterns of passengers or from the analysis of a given rebalancing policy assuming no congestion (according to the procedure discussed in Section 3.3.2). For the simple example, one can write

$$q_{ik} \leq \frac{c(i,j)\pi_{ik}}{\pi_{ik} + \pi_{ij}}, \quad q_{ik} \leq \frac{c(j,k)\pi_{ik}}{\pi_{ik} + \pi_{jk}}. \tag{3.17}$$

Similar constraints can be written for q_{ij} and q_{jk} so that (3.16) is satisfied.

For a general road network $G(\mathcal{V}, \mathcal{E})$, let $\mathcal{P}_{ij}$ be the set of possible non-cyclic paths from station i to j, $i, j \in \mathcal{V}$ (assuming no back tracking). Each path $\varphi_{ij} \in \mathcal{P}_{ij}$ is given by a sequence of road links (u, v), i.e. $\varphi_{ij} = \{(i, u), (u, v), ..., (w, j)\}$ $(u, v \in \mathcal{V})$. The number of possible paths from i to j is given by $|\mathcal{P}_{ij}|$. Let $a_{ij}^{(u,v)}$ denote the fraction of trips from i to j that go through road link (u, v). Denote by $c(u, v)$ the capacity of road link (u, v). For trips going through multiple road segments, the virtual road capacity is determined by the segment with the lowest capacity. One can then consider as virtual road capacities:

$$q_{ij} = \sum_{\varphi_{ij} \in \mathcal{P}_{ij}} \min_{(u,v) \in \varphi_{ij}} \left\{ \frac{c(u,v) a_{ij}^{(u,v)} \pi_{ij}}{\sum_{k,l, \text{such that } (u,v) \in \varphi_{kl}} a_{kl}^{(u,v)} \pi_{kl}} \right\}.$$

However, this formulation can lead to over-conservative results for example if one road segment along one of the paths fills to capacity while the road segments of other paths are relatively unused. Thus, the routing scheme, represented by $a_{ij}^{(u,v)}$, is central to the characterization of the mapping from the road network into the queueing network model.

An alternate approach is to extend the Jackson network model to a more general type of product-form queueing networks known as BCMP networks [69]. The key is to build the queueing network to directly represent the road network, where each queue represents a specific road segment. BCMP networks support multiple classes of agents. In this case, each vehicle traveling from station i to j is said to be in class $\{ij\}$. Once a vehicle reaches its destination, it switches its class to $\{jk\}$ where k is its next destination. The routing distribution at each road queue is class dependent, so different paths can be chosen for each origin-destination pair. The travel time distribution on each road queue is a Coxian distribution [55] (can be used to approximate many distributions) and can be load dependent. The model couples routing with rebalancing through the (class-dependent) routing probabilities in the network, and is a interesting avenue of future research.

3.8 Conclusion

In this chapter we studied a novel model for AMoD systems using the theory of Jackson networks. We showed that an optimal open-loop rebalancing policy for an AMoD system can be readily found by solving a linear program. Based on this policy, we developed a closed-loop, real-time rebalancing policy that can be applied to large scale systems consisting of thousands of vehicles and hundreds of stations. We applied these methods to case studies of New York City and Singapore. In Manhattan, we found that all taxi demands can be met with an AMoD system only around 70% the size of the taxi fleet, and in Singapore, we found that an AMoD system may be able to service all transportation demands with only 40% of the current number of vehicles in the city. The key to our approach in this chapter is that using queueing network theory, we were able to systematically *synthesize* control policies for AMoD systems as well as *analytically* evaluate their performance.

One key limitation to this model in its current form is its inability to be mapped directly to road networks and account for congestion effects. In Chapter 5 we develop an alternative model to study the impact of AMoD systems on urban traffic congestion.

Chapter 4

Queueing Network Analysis and Control of Human-Driven Mobility-on-Demand

4.1 Introduction

In this chapter we extend the Jackson network model of an AMoD system to a car sharing (MoD) system with conventional, human-driven vehicles. To rebalance the MoD system in the absence of self-driving cars, the system operator hires human drivers to drive excess vehicles to stations where they are needed. However, these drivers then themselves become unbalanced. This adds an additional dimension of complexity to the problem. To address the imbalance of drivers, we employ the strategy whereby the drivers are themselves "rebalanced" by driving select customers to their destinations as a taxi service. In this way, the MoD system can be viewed as an one-way customer-driven car sharing service mixed with a taxi service.

Specifically, in this chapter we develop a queueing network framework for the design, analysis, and control of (human-driven) mobility-on-demand systems. We then apply the insights from this queueing framework to develop real-time closed-loop policies to control such systems. From a modeling and analysis perspective, we consider a model similar to the one proposed in [19], which hinges upon the optimization of rebalancing *rates* and is studied under a fluidic approximation (where customers, drivers, and vehicles are modeled as a continuum). The model in [19] offers insights into the minimum number of vehicles and drivers required in an MoD system but does not provide key performance metrics in terms of quality of service (i.e., the availability of vehicles at stations or the customer wait times). The model developed in this chapter can be viewed as an extension of [19] taking into account the stochasticity of the system. From a controls perspective, real-time closed-loop policies for one-way car sharing systems have been studied in [16] and [17] with the objective of maximizing profit, where the rebalancing of vehicles is modeled as a cost. The methods developed in this chapter differ from these works in two key respects: (1) in addition to minimizing cost, our key objective is quality of service for customers in terms of vehicle availabilities and wait times, and (2) we explicitly control the movement of rebalancing drivers which makes the system self-contained (e.g., drivers do not need to rely on public transit to rebalance themselves).

This chapter is organized as follows: Section 4.2 describes in detail the queueing network model of an MoD system. Specifically, the approach is to model an MoD system as two coupled closed Jackson networks with passenger loss. Section 4.3 offers two approaches for the open-loop control of an MoD system. In the first approach, the optimal rebalancing parameters are solved by two decoupled linear programs, and are therefore efficient to compute and easy to scale, but only approximately balance the system. In the second approach, nonlinear optimization techniques are used (with higher computational cost) to balance the system exactly. The rebalancing techniques are then applied to a system sizing example based on taxi data in Manhattan. Section 4.4 introduces a real-time closed-loop control policy useful for practical systems. In Section 4.5 the closed-loop policy is used for a case study of an MoD system in Manhattan. Finally, in Section 4.6 we draw our conclusions.

4.2 Model Description and Problem Formulation

4.2.1 MoD system model

In this section we formally describe the MoD system under consideration and cast it within a queueing network framework by modeling the system as two coupled, closed Jackson networks. We consider N stations with unlimited parking capacity placed in a given geographical area, m_v vehicles that can be rented by customers for one-way trips between stations, and m_d "rebalancing" drivers employed to rebalance the vehicles by driving them to the stations where they are needed. The drivers then "rebalance" themselves by driving customers to their destinations, operating as a taxi service. These assumptions require each driver to *always* have access to a vehicle since the driver's task involves driving a vehicle with or without a customer (A driver left at a station without a vehicle is effectively "stranded"). We therefore pose the constraint $v_i \geq d_i$, where v_i is the number of vehicles at station i and d_i is the number of drivers at station i. With this requirement, we may view the MoD system as two systems operating in parallel – a one-way customer-driven car sharing service with $m_v - m_d$ vehicles and a taxi service with m_d vehicles. It is worth noting that there are other, more elaborated ways of managing an MoD system which is not in the scope of this thesis. For example, in [19], the authors also consider customers potentially riding with multiple drivers. One could also envision a system where drivers can drive other drivers or take public transportation to stations with excess cars. The extension of our model to such cases is an interesting avenue for future research.

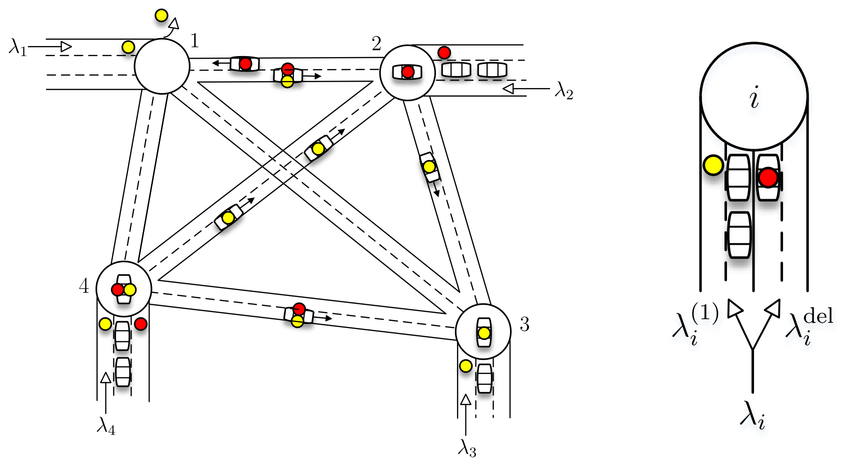


Figure 4.1: Left: MoD system model. Yellow dots represent customers and red dots represent rebalancing drivers. Customers can drive themselves or ride with a rebalancing driver. Customers are lost if no vehicles are available (station 1). Right: each customer arriving at station i is delegated to either System 1 (customer-driven vehicles) or System 2 (taxi system).

Customers arrive to station i according to a Poisson process with parameter λ_i. Upon arrival at station i, the customer selects a destination j with probability p_{ij}, where $p_{ij} \geq 0$, $p_{ii} = 0$, and $\sum_j p_{ij} = 1$. Furthermore, we assume that the probabilities $\{p_{ij}\}_{ij}$ constitute an irreducible Markov chain. The customer can travel to his/her destination in one of two ways: (1) the customer drives a vehicle to his/her destination, or (2) the customer is taken to his/her destination by a rebalancing driver. The travel time from station i to station j is an exponentially distributed random variable with mean $T_{ij} > 0$. As in Chapter 3, a "passenger loss" model is employed, where if a vehicle is not available upon the arrival of a customer, the customer immediately leaves the system. However, due to the additional complexity of our MoD model (a one-way car sharing service and a taxi service in parallel) the passenger loss assumption is more involved. We assume that upon arrival at a station, a customer is delegated to one of the two parallel systems by the MoD service operator (see Figure 4.1). The customer is lost if there are no available vehicles in *the system to which he/she was delegated*. For example, if a customer is delegated to the taxi system and no taxis are immediately available, the customer *cannot* switch over to the other system and drive himself/herself to the desired destination. The modeling consequences of this assumption will be further discussed in the next section. The performance criterion of interest in this case is the probability a customer will find an available vehicle (both empty vehicles and taxis) at each station. In Section 4.4 we will relax the passenger loss assumption and investigate the more realistic scenario where customers form a queue to wait for available vehicles. The performance of the system is then measured by customer wait times.

4.2.2 Jackson network model of an MoD system

We now formally cast the MoD model described in the previous section within a queueing network framework. The key is to construct an abstract queueing network where the stations are modeled as single-server (SS) nodes and the roads as infinite-server (IS) nodes, as done in Chapter 3. Vehicles form a queue at each SS node while waiting for customers and are "serviced" when a customer arrives. The vehicle then moves from the SS node to the IS node connecting the origin to the destination selected by the customer. After spending an exponentially distributed amount of time (with mean T_{ij}) in the IS node, the vehicle moves to the destination SS node. This setup is then a *closed Jackson network with respect to the vehicles.* To capture the idea that the MoD system consists of two systems (customer-driven system and taxi system) operating in parallel, we model the MoD system as two coupled closed Jackson networks. More formally, let System 1 represent the Jackson network of $m_v - m_d$ customer-driven vehicles, and System 2 represent the network of m_d taxis. Let $S^{(k)}$ represent the set of SS nodes and $I^{(k)}$ represent the set of IS nodes in the k^{th} Jackson network, where $k = \{1, 2\}$. For each network, each SS node is connected to every other SS node through an IS node. Thus, each network consists of $N + N(N-1) = N^2$ nodes (the IS node from station i to itself is not represented since $p_{ii} = 0$). For each IS node $i \in I^{(k)}$, let Parent(i) and Child(i) be the origin and destination of i, respectively. The routing matrix $\{r_{ij}^{(k)}\}_{ij}$ in Jackson network k can then be written as

$$r_{ij}^{(k)} = \begin{cases} p_{il}^{(k)} & i \in S^{(k)}, j \in I^{(k)}, i = \text{Parent}(j), l = \text{Child}(j), \\ 1 & i \in I^{(k)}, j \in S^{(k)}, j = \text{Child}(i), \\ 0 & \text{otherwise}, \end{cases}$$

where the first case is the movement from an SS node to an IS node and the second case is from an IS node to its unique destination SS node. The service times at each node are exponentially distributed with mean service rates

$$\mu_i^{(k)}(n) = \begin{cases} \lambda_i^{(k)} & \text{if } i \in S^{(k)}, \\ \frac{n}{T_{jl}} & \text{if } i \in I^{(k)}, j = \text{Parent}(i), l = \text{Child}(i), \end{cases}$$

where n is the number of vehicles in the IS node. With this formulation we have defined two closed Jackson networks of the same form as the one introduced in Chapter 3, which is amenable to analysis.

We now return to the customer arrival process and the loss model assumption. Recall that customers arrive at station i according to a Poisson process with rate λ_i. Upon arrival, and depending on the destination, a customer is first delegated to either System 1 or System 2. This can be seen as a Bernoulli splitting of the customer arrival process into two Poisson processes for each desired destination. Denote by $\lambda_i^{(1)}$ the total rate of customers delegated to System 1, by $p_{ij}^{(1)}$ the routing probabilities associated with System 1 ($p_{ij}^{(1)} \geq 0$, $p_{ii}^{(1)} = 0$, $\sum_j p_{ij}^{(1)} = 1$), by λ_i^{del} the total rate of customers delegated to System 2, and by η_{ij} the routing probabilities associated with System 2. We have the relationship

$$\lambda_i = \lambda_i^{(1)} + \lambda_i^{\mathrm{del}}$$

for each station i. We also define q_i to be the total fraction of customers delegated to System 1 at station i, i.e., $q_i = \lambda_i^{(1)}/\lambda_i$. We can also write $1 - q_i = \lambda_i^{\mathrm{del}}/\lambda_i$. The routing probabilities of the customers can then be expressed as

$$\begin{aligned} p_{ij} &= \mathbb{P}\{i \to j \mid \text{System 1}\}\, p_{ij}^{(1)} + \mathbb{P}\{i \to j \mid \text{System 2}\}\, \eta_{ij} \\ &= q_i\, p_{ij}^{(1)} + (1 - q_i)\, \eta_{ij}. \end{aligned} \tag{4.1}$$

We can equivalently say that the Poisson rate of customers originating at station i and headed for station j is $\lambda_i\, p_{ij}$. The arrival rate of these customers to System 1 is then $\lambda_i^{(1)} p_{ij}^{(1)}$ and the arrival rate to System 2 is $\lambda_i^{\mathrm{del}} \eta_{ij}$. Thus Relation (4.1) can be rewritten as

$$\lambda_i\, p_{ij} = \lambda_i^{(1)} p_{ij}^{(1)} + \lambda_i^{\mathrm{del}} \eta_{ij}. \tag{4.2}$$

If the delegation process is known (i.e., λ_i^{del} and η_{ij} are known), the routing probabilities for System 1 can be solved by rearranging (4.1) as

$$p_{ij}^{(1)} = \frac{1}{q_i}\, p_{ij} - \frac{1 - q_i}{q_i}\, \eta_{ij}. \tag{4.3}$$

In Section 4.2.3 we will describe in detail how to solve for λ_i^{del} and η_{ij}. Arrival rates $\lambda_i^{(1)}$, routing probabilities $p_{ij}^{(1)}$, and mean travel times T_{ij} fully describe the System 1 Jackson network.

Now we consider the second Jackson network, System 2, which models the m_d vehicles operating as a taxi service. This network must not only provide service to customers but also rebalance the MoD system to ensure quality of service. To incorporate the notion of vehicle rebalancing, we use the concept of "virtual" customers as in Chapter 3. Virtual customers are generated at station i according to a Poisson process with parameter ψ_i and routing probabilities ξ_{ij}, independent from the real customer arrival process. Virtual customers are lost upon arrival if a taxi is not immediately available, just like real customers. In this way, virtual customers promote rebalancing while not enforcing a strict rebalancing rate, which is key to retaining tractability in the model. The overall customer arrival rate (real and virtual) at station i for System 2 is

$$\lambda_i^{(2)} = \lambda_i^{\mathrm{del}} + \psi_i.$$

With respect to the vehicles, $\lambda_i^{(2)}$ is the exponentially distributed service rate at SS node i. The routing probabilities for this network can be defined as

$$\begin{aligned} p_{ij}^{(2)} &= \mathbb{P}\{i \to j \mid \text{ virtual}\}\, \xi_{ij} + \mathbb{P}\{i \to j \mid \text{ real}\}\, \eta_{ij} \\ &= \frac{\psi_i}{\lambda_i^{(2)}}\, \xi_{ij} + \frac{\lambda_i^{\text{del}}}{\lambda_i^{(2)}}\, \eta_{ij} \\ &= p_i\, \xi_{ij} + (1 - p_i)\, \eta_{ij}, \end{aligned} \tag{4.4}$$

where $p_i = \frac{\psi_i}{\lambda_i^{(2)}}$.

To summarize our Jackson network model, customers arrive at station i headed for station j according to a Poisson process with rate $\lambda_i\, p_{ij}$. Upon arrival, each customer is delegated to one of two systems, the customer-driven system (System 1) or the taxi system (System 2). The probability of the customer (going from station i to j) being delegated to System 1 is $\frac{\lambda_i^{(1)} p_{ij}^{(1)}}{\lambda_i\, p_{ij}}$ and the probability of the customer delegated to System 2 is $\frac{\lambda_i^{\text{del}} \eta_{ij}}{\lambda_i\, p_{ij}}$ (from (4.3)). Once the customer has been delegated, if he/she finds the station empty of vehicles, the customer immediately leaves the system. Once delegated, a customer cannot switch from System 1 to System 2 or vice versa. We note that in the same way that ψ_i represents the rebalancing-promoting rate of vehicles in the MoD system, λ_i^{del} represents the rebalancing-promoting rate of the drivers. Together, the parameters ψ_i, ξ_{ij}, λ_i^{del}, and η_{ij} constitute the open-loop controls for our model of an MoD system. The open-loop control problem is formalized and solved in Section 4.3.

4.2.3 Performance criteria

Our task to control the MoD system involves optimizing the parameters λ_i^{del} (rebalancing the drivers) and ψ_i (rebalancing the vehicles) as well as the routing probabilities η_{ij} and ξ_{ij}. The key performance metric is the availability of vehicles (the probability that a customer will find an available vehicle), given by (2.10). As stated in the previous chapter, a natural notion of rebalancing is to ensure that $A_i(m) = A_j(m)$ for all $i, j \in S$ (or equivalently $\gamma_i = \gamma_j$ for all $i, j \in S$, as implied by (2.10)). The relative utilizations for each Jackson network are defined as follows

$$\begin{aligned} \gamma_i^{(1)} &= \frac{\pi_i^{(1)}}{\mu_i^{(1)}} = \frac{\pi_i^{(1)}}{\lambda_i - \lambda_i^{\text{del}}} \quad \text{for all } i \in S^{(1)}, \\ \gamma_i^{(2)} &= \frac{\pi_i^{(2)}}{\mu_i^{(2)}} = \frac{\pi_i^{(2)}}{\lambda_i^{\text{del}} + \psi_i} \quad \text{for all } i \in S^{(2)}, \end{aligned}$$

where $\pi_i^{(k)}, i \in S^{(k)}, k = \{1, 2\}$ satisfies (2.7). For autonomous MoD systems, the constraint $\gamma_i = \gamma_j$ embodies two features: (1) fairness, as characterized by equal availability across all stations, and (2) performance, since the availability at each station approaches 100% as the number of vehicles increases. We will apply this constraint to both Jackson networks in our MoD system as first it is a direct generalization of the approach used for autonomous MoD systems and second it yields a linear optimization problem (Section 4.3.1) that is easy to compute and scale to large systems. However, as we will discuss in Section 4.3.2, such approach only approximately balances the system (even though the approximation is often remarkably good). We then introduce an exact approach in Section 4.3.3 that relies on nonlinear optimization, which does ensure fairness while maintaining system performance, but incurs a higher computational cost. Collectively, the open-loop control approaches of Section 4.3 are useful for analysis and design tasks such as system sizing (Section 4.3.4) and drive the development of closed-loop policies (Section 4.4).

4.3 Analysis and Design of MoD Systems

4.3.1 Approximate MoD rebalancing

In this section we formulate a linear optimization approach to (approximately) rebalance an MoD system. Specifically, we would like to manipulate our control variables λ_i^{del}, ψ_i, η_{ij}, and ξ_{ij} such that $\gamma_i^{(1)} = \gamma_j^{(1)}$ for all $i, j \in S^{(1)}$ and $\gamma_i^{(2)} = \gamma_j^{(2)}$ for all $i, j \in S^{(2)}$. To minimize the cost of MoD service, we would like to simultaneously minimize the mean number of rebalancing vehicles on the road (minimize energy use and possibly congestion), given by $\sum_{i,j} T_{ij}\, \xi_{ij}\, \psi_i$, as well as the number of rebalancing drivers needed, given by $\sum_{i,j} T_{ij}\, (\xi_{ij}\, \psi_i + \eta_{ij}\, \lambda_i^{\text{del}})$. We can state this multi-objective problem as follows:

MoD Rebalancing Problem (MRP): Given an MoD system modeled as 2 closed Jackson networks, solve

$$
\begin{aligned}
\underset{\lambda_i^{\text{del}},\psi_i,\eta_{ij},\xi_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\, \xi_{ij}\, \psi_i \text{ and } \sum_{i,j} T_{ij}\, (\xi_{ij}\, \psi_i + \eta_{ij}\, \lambda_i^{\text{del}}) \\
\text{subject to} \quad & \gamma_i^{(k)} = \gamma_j^{(k)} \quad i, j \in S^{(k)}, k = 1, 2 \\
& \sum_j \eta_{ij} = 1, \\
& \sum_j \xi_{ij} = 1, \\
& \eta_{ij} \geq 0, \;\; \xi_{ij} \geq 0, \;\; \lambda_i^{\text{del}} \geq 0, \;\; \psi_i \geq 0 \\
& \lambda_i^{\text{del}} \eta_{ij} \leq \lambda_i p_{ij} \quad i, j \in \{1, \dots, N\}
\end{aligned}
\tag{4.5}
$$

Note that the last constraint in the MRP ensures that the rate of customers delegated to System 2 is less than the total arrival rate of customers. This ensures that System 1 always has non-negative customer arrival rates.

Remarkably, the MRP can be solved as two decoupled linear optimization problems with the same form as in [19] (which uses a deterministic, fluidic model). This result, stated in Theorem 4.3.5, constitutes the main contribution of this section. By decoupling the constraints, we can show that the two objectives are indeed aligned, i.e., minimizing the second objective will minimize the first as well. We begin by presenting supporting lemmas that are used in the proof of Theorem 4.3.5. The first two lemmas establish some structural properties of the model and are very similar to Lemmas 3.3.1 and 3.3.2; their proofs are virtually identical and are thus omitted. The first lemma allows the balance equations of the Jackson network to be solved by considering only the SS nodes.

Lemma 4.3.1 (Folding of balance equations)**.** *Consider either System 1 or System 2 from Section 4.2.2. The relative throughputs π's for the SS nodes can be found by solving the reduced balance equations*

$$\pi_i^{(k)} = \sum_{j \in S^{(k)}} \pi_j^{(k)} p_{ji}^{(k)} \quad \text{for all } i \in S^{(k)},\ k = \{1, 2\}, \tag{4.6}$$

where SS nodes are considered in isolation. The π's for the IS nodes are then given by

$$\pi_i^{(k)} = \pi_{\text{Parent}(i)}^{(k)} p_{\text{Parent}(i)\text{Child}(i)}^{(k)} \text{ for all } i \in I^{(k)}, k = \{1, 2\}. \tag{4.7}$$

Lemma 4.3.2. *For any rebalancing policy $\{\psi_i\}_i$ and $\{\xi_{ij}\}_{ij}$, it holds for all $i \in S^{(2)}$*

1. $\gamma_i^{(2)} > 0$,
2. $(\lambda_i^{\text{del}} + \psi_i)\,\gamma_i^{(2)} = \sum_{j \in S^{(2)}} \gamma_j^{(2)}\,(\psi_j\,\xi_{ji} + \lambda_j^{\text{del}}\,\eta_{ji})$.

Similarly, for System 1,

1. $\gamma_i^{(1)} > 0$,
2. $(\lambda_i - \lambda_i^{\text{del}})\,\gamma_i^{(1)} = \sum_{j \in S^{(1)}} \gamma_j^{(1)}\,(\lambda_j\,p_{ji} - \lambda_j^{\text{del}}\,\eta_{ji})$.

In the next two lemmas, we introduce new optimization variables $\{\alpha_{ij}\}_{ij}$ and $\{\beta_{ij}\}_{ij}$ and show that the constraints $\gamma_i = \gamma_j$ in the MRP are equivalent to linear constraints in these new variables. The proofs are similar to the proof of Theorem 3.3.3.

Lemma 4.3.3 (Constraint equivalence for System 1)**.** *Assume that β_{ij} is given. Set $\lambda_i^{\text{del}} = \sum_{j \neq i} \beta_{ij}$, $\eta_{ii} = 0$, and for $j \neq i$,*

$$\eta_{ij} = \begin{cases} \beta_{ij}/\lambda_i^{\text{del}} & \textit{if } \lambda_i^{\text{del}} > 0, \\ 1/(N-1) & \textit{otherwise.} \end{cases}$$

With this definition, the constraint

$$\sum_{j \in S^{(1)}, j \neq i} (\beta_{ij} - \beta_{ji}) = \lambda_i - \sum_{j \in S^{(1)}, j \neq i} \lambda_j\,p_{ji} \tag{4.8}$$

is equivalent to the constraint

$$\gamma_i^{(1)} = \gamma_j^{(1)}, \quad i, j \in S^{(1)}.$$

Proof. First, rewrite (4.8) in terms of λ_i^{del} and η_{ij}. We then have

$$\lambda_i - \lambda_i^{\text{del}} = \sum_{j \neq i} (\lambda_j \, p_{ji} - \lambda_j^{\text{del}} \, \eta_{ji}).$$

Substituting this expression into the last statement of Lemma 4.3.2, we have

$$\left(\sum_{j \neq i} (\lambda_j \, p_{ji} - \lambda_j^{\text{del}} \, \eta_{ji}) \right) \gamma_i^{(1)} = \sum_{j \neq i} \gamma_j^{(1)} \, (\lambda_j \, p_{ji} - \lambda_j^{\text{del}} \, \eta_{ji}). \tag{4.9}$$

Let $\varphi_{ij} := \lambda_j \, p_{ji} - \lambda_j^{\text{del}} \, \eta_{ji}$ and $\zeta_{ij} := \varphi_{ij} / \sum_j \varphi_{ij}$. Note that $\sum_j \varphi_{ij} = \lambda_i - \lambda_i^{\text{del}} = \lambda_i^{(1)} > 0$ by assumption. The variables ζ_{ij} can be considered transition probabilities of an irreducible Markov chain, and (4.9) can be rewritten in matrix form as $Z\gamma^{(1)} = \gamma^{(1)}$. Matrix Z is an irreducible, row stochastic matrix, so by the Perron-Frobenius theorem [62], the eigenspace associated with the eigenvalue 1 is one-dimensional. Therefore the unique solution to $Z\gamma^{(1)} = \gamma^{(1)}$ (up to a scaling factor) is the vector $(1, ..., 1)^T$, so $\gamma_i^{(1)} = \gamma_j^{(1)}$ for all i, j. □

Lemma 4.3.4 (Constraint equivalence for System 2)**.** *Assume that α_{ij} is given. Set $\psi_i = \sum_{j \neq i} \alpha_{ij}$, $\xi_{ii} = 0$, and for $j \neq i$,*

$$\xi_{ij} = \begin{cases} \alpha_{ij}/\psi_i & \text{if } \psi_i > 0, \\ 1/(N-1) & \text{otherwise.} \end{cases}$$

With this definition, the constraint

$$\sum_{j \neq i} (\alpha_{ij} - \alpha_{ji}) = \sum_{j \neq i} (\beta_{ji} - \beta_{ij}) \tag{4.10}$$

is equivalent to the constraint

$$\gamma_i^{(2)} = \gamma_j^{(2)}, \quad i, j \in S^{(2)}.$$

The proof is essentially identical to the proof of Lemma 4.3.3 and is omitted. Furthermore, we can substitute (4.8) into (4.10) and rewrite (4.10) as

$$\sum_{j \neq i} (\alpha_{ij} - \alpha_{ji}) = -\lambda_i + \sum_{j \neq i} \lambda_j p_{ji}. \tag{4.11}$$

With this substitution, we have decoupled the original MRP constraints to those associated with System 1 (λ_i^{del} and η_{ij}) and those associated with System 2 (ψ_i and ξ_{ij}). Using Lemmas 4.3.3 and 4.3.4, one can also show that minimizing the second objective in the MRP also minimizes the first objective. We now state the main result of this section.

Theorem 4.3.5 (Solution to MRP)**.** *Consider the following two decoupled linear optimization problems*

$$\begin{aligned} \underset{\beta_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\, \beta_{ij} \\ \text{subject to} \quad & \sum_{j \neq i} (\beta_{ij} - \beta_{ji}) = \lambda_i - \sum_{j\neq i} \lambda_j\, p_{ji} \\ & 0 \leq \beta_{ij} \leq \lambda_i\, p_{ij} \end{aligned} \tag{4.12}$$

$$\begin{aligned} \underset{\alpha_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\, \alpha_{ij} \\ \text{subject to} \quad & \sum_{j \neq i} (\alpha_{ij} - \alpha_{ji}) = -\lambda_i + \sum_{j\neq i} \lambda_j\, p_{ji} \\ & 0 \leq \alpha_{ij} \end{aligned} \tag{4.13}$$

These problems are always feasible. Let β^*_{ij} *and* α^*_{ij} *be optimal solutions to problems* (4.12) *and* (4.13) *respectively. By making the following substitutions*

$$\begin{aligned} \lambda_i^{\text{del}} &= \sum_{j\neq i} \beta^*_{ij}, \\ \psi_i &= \sum_{j\neq i} \alpha^*_{ij}, \\ \eta_{ij} &= \begin{cases} 0 & \text{if } i = j, \\ \beta^*_{ij}/\lambda_i^{\text{del}} & \text{if } \lambda_i^{\text{del}} > 0, i \neq j, \\ 1/(N-1) & \text{otherwise,} \end{cases} \\ \xi_{ij} &= \begin{cases} 0 & \text{if } i = j, \\ \alpha^*_{ij}/\psi_i & \text{if } \psi_i > 0, i \neq j, \\ 1/(N-1) & \text{otherwise,} \end{cases} \end{aligned}$$

one obtains the optimal solution to the MRP.

Proof. Problem (4.13) is an uncapacitated minimum cost flow problem and is always feasible. The upper bound constraint in Problem (4.12) constitutes a standard condition for the existence of a feasible solution in a minimum cost flow problem [19, 70, p. 220]. The main task of the proof is showing that the constraints $\gamma_i^{(k)} = \gamma_j^{(k)}$ are equivalent to the constraints in (4.12) and (4.13), which is shown in Lemmas 4.3.3 and 4.3.4. □

This result allows us to compute the open-loop control very efficiently and can be applied to very large systems comprising hundreds of stations. We apply this technique in the next section to compute the availability of vehicles at each station and in Section 4.3.4 to the problem of "sizing" an MoD system (i.e., determining the optimal fleet size and number of drivers).

4.3.2 Availability of vehicles for real passengers

In general, the availability of vehicles at each station in the customer-driven system is different from the taxi system. The approach in the previous section calculates the availability of the two systems separately, but the availability of vehicles in the taxi system applies not only to real customers, but to virtual customers as well. To calculate the availability for all (real) passengers, we must consider both systems concurrently. First, we note that the total throughput of both real and virtual customers for both networks is given by

$$\Lambda_i^{\text{tot}}(m_v, m_d) = \Lambda_i^{(1)}(m_v - m_d) + \Lambda_i^{(2)}(m_d).$$

The throughput of only real passengers is given by

$$\Lambda_i^{\text{pass}}(m_v, m_d) = \Lambda_i^{(1)}(m_v - m_d) + \frac{\lambda_i^{\text{del}}}{\lambda_i^{\text{del}} + \psi_i}\Lambda_i^{(2)}(m_d),$$

where the second term on the right hand side reflects the fraction of real passengers in the taxi network. Thus, the vehicle availability for real passengers is given by

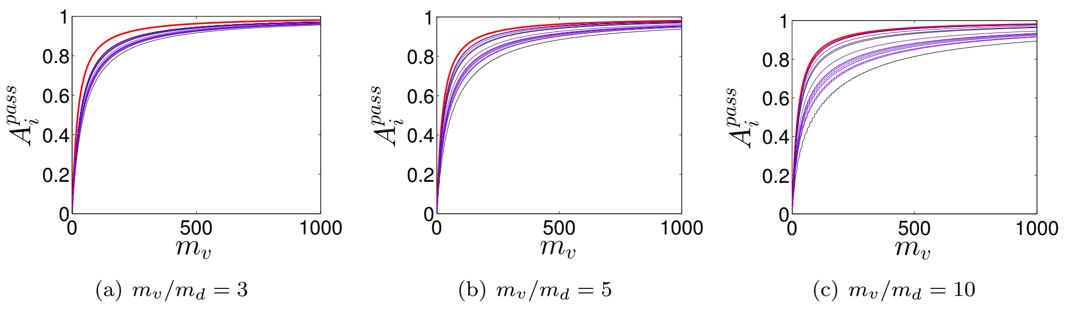
$$A_i^{\text{pass}}(m_v, m_d) = \frac{\Lambda_i^{\text{pass}}(m_v, m_d)}{\lambda_i}.$$

With some algebraic manipulations, $A_i^{\text{pass}}(m_v, m_d)$ can be rewritten as

$$A_i^{\text{pass}}(m_v, m_d) = A_i^{(1)}(m_v - m_d)q_i + A_i^{(2)}(m_d)(1 - q_i). \tag{4.14}$$

Since q_i is in general not the same for all i, the availability of vehicles for real passengers will *not* be the same for every station. Figure 4.2 shows that the rebalancing technique described in the previous section will produce unbalanced vehicle availabilities for real passengers. Furthermore, the degree of system imbalance grows with the vehicle-to-driver ratio, which intuitively makes sense since there are fewer drivers to rebalance the system when the vehicle-to-driver ratio is high. However, it is important to note that even though the availabilities at each station are not the same, as $m_v \to \infty$ and $m_d \to \infty$, the availabilities approach one for *all* stations.

The red line in Figure 4.2 shows the availability of the system if there were m_v drivers and m_v vehicles (or equivalently a taxi system or an AMoD system). It is clear that the autonomous MoD system yields better performance both in terms of throughput (high availability) and fairness (same



(a) $m_v/m_d = 3$ (b) $m_v/m_d = 5$ (c) $m_v/m_d = 10$

Figure 4.2: Overall vehicle availability for passengers for a randomly generated system with 20 stations. The red line shows the availability if there were as many drivers as vehicles (or an AMoD system). 4.2(a) shows a vehicle-to-driver ratio of 3, 4.2(b) shows a vehicle-to-driver ratio of 5, and 4.2(c) shows a vehicle-to-driver ratio of 10.

availability at all stations) due to the ability of every vehicle to perform rebalancing trips. This result presents a strong case for the advantages of AMoD systems over current human-driven MoD systems in operation.

To validate these results, simulations are performed using a small 5-station system positioned in a 5×5 grid with vehicles traveling at a constant speed of 0.2 units per time step. In the simulation, customers arrive at each station according to a Poisson process with rate λ_i and report their desired destinations. Based on the destinations, customers are delegated by a Bernoulli random variable to either drive themselves to their destination or be driven to their destination by a driver. If in either case a vehicle and/or driver is unavailable, the customer leaves the system. At each time step, after the customers are delegated, rebalancing is performed by generating "virtual customers" according to a Poisson process with rate ψ_i and assigning the virtual customers to available drivers at each station. The availability for customers at each station is computed by dividing the number of successfully serviced customers by the total number of customer arrivals. Simulations are performed for 5 system sizes (number of vehicles and drivers) keeping the vehicle-to-driver ratio fixed. In order to capture the steady-state behavior of the system, each simulation is performed for 50,000 time steps. Figure 4.3 shows the simulated vehicle availabilities, averaged over 50 simulation runs, compared to the theoretical results computed using (4.14). The simulations show remarkable consistency between the theoretical availability probabilities and experimental results, and that availability for real passengers indeed differ across stations under the control policy presented in Section 4.3.1.

4.3.3 Exact MoD rebalancing

It is clear that applying the rebalancing constraints separately for the two networks as done in (4.5) does not yield a balanced system in terms of vehicle availability for all customers. Indeed, the

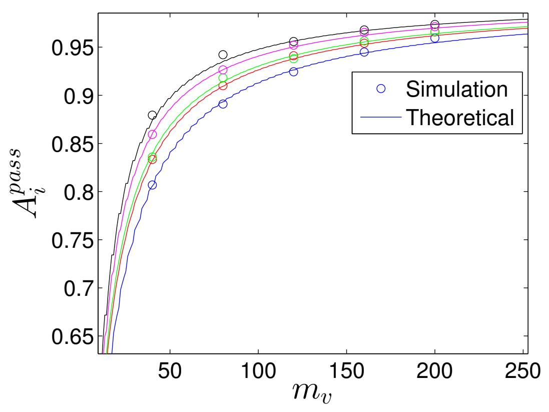


Figure 4.3: Validation of queueing model showing availability for real customers. A 5-station system is simulated with 40, 80, 120, 160, and 200 vehicles and a vehicle-to-driver ratio of 4. The circles represent mean availabilities over 50 simulation runs for each station. Each color represents a different station. The average standard deviation for the simulation results is 0.0189.

constraints needed to balance availability for the passengers is

$$A_i^{\text{pass}}(m_v, m_d) = A_j^{\text{pass}}(m_v, m_d) \quad \text{for all } i, j \in \{1, ..., N\}. \tag{4.15}$$

Note that this constraint is dependent on the number of vehicles and the number of drivers in the system, and thus cannot be reduced to a linear constraint in the decision variables. Taking into account the modified constraints, we reformulate our problem to the following:

Exact MoD Rebalancing Problem (EMRP): Given an MoD system with N stations, m_v vehicles, and m_d drivers modeled as two closed Jackson networks, solve

$$\begin{aligned}
\underset{\lambda_i^{\text{del}}, \psi_i, \eta_{ij}, \xi_{ij}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}\, \xi_{ij}\, \psi_i - c \sum_i A_i^{(2)}(m_v - m_d) \\
\text{subject to} \quad & \gamma_i^{(1)} = \gamma_j^{(1)} \\
& A_i^{\text{pass}}(m_v, m_d) = A_j^{\text{pass}}(m_v, m_d) \\
& \sum_j \eta_{ij} = 1, \\
& \sum_j \xi_{ij} = 1, \\
& \eta_{ij} \geq 0,\ \xi_{ij} \geq 0,\ \lambda_i^{\text{del}} \geq 0,\ \psi_i \geq 0, \\
& \lambda_i^{\text{del}} \eta_{ij} \leq \lambda_i p_{ij}, \quad i, j \in \{1, \ldots, N\}.
\end{aligned} \tag{4.16}$$

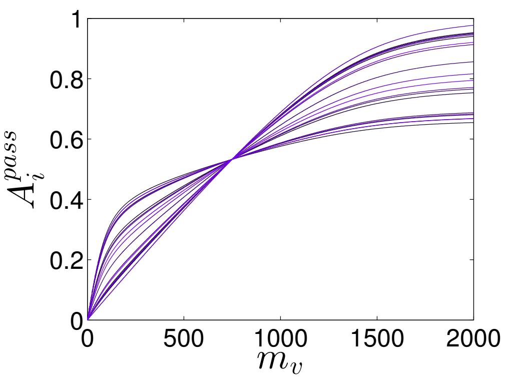
The objective function now trades off two objectives that are not always aligned – minimizing the number of rebalancing trips while maximizing the overall availability (note that the first constraint balances and maximizes the availability of the customer-driven system, so to maximize overall availability, we only need to maximize the availabilities in the taxi system). A weighting factor c is used in this trade-off. The constraint $\gamma_i^{(1)} = \gamma_j^{(1)}$ is used in conjunction with (4.15) to ensure the availability of the customer-driven system remains balanced. The strategy is to use the taxi system to enforce the availability constraint for real customers with the intuition that the system operator has full control over the rebalancing of the taxi system while the rebalancing of the customer-driven system depends on the arrival process of the customers, which is subject to large stochastic fluctuations. If the customer-driven system becomes unbalanced, empty vehicles will accumulate at some stations for extended periods of time, decreasing the effective number of vehicles in the system (see Section 4.4).

The modified availability constraint (4.15) is nonlinear and involves solving for $A_i^{(2)}$ using MVA at each iteration ($A_i^{(1)}$ is also needed, but only needs to be computed once). For systems of reasonably small size ($\sim$ 20 stations and $\sim$ 1000 vehicles), MVA can be carried out quickly ($<$ 1 sec). For larger networks, an approximate MVA technique exists which involves solving a set of nonlinear equations rather than iterating through all values of m [63]. The EMRP can be solved using nonlinear optimization techniques for a given number of vehicles and drivers. We let A^* represent the balanced availability A_i^{pass} obtained by solving the EMRP.

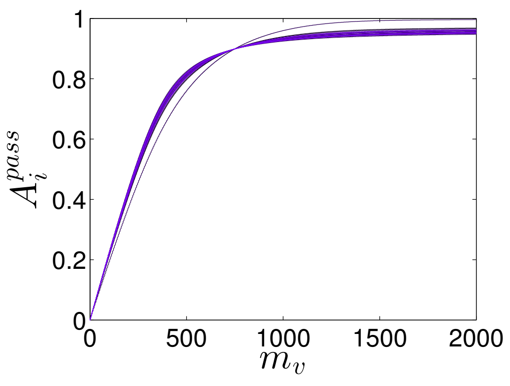
To demonstrate this technique on a realistic system, key system parameters (arrival rates, routing probabilities, and travel times) were extracted from a portion of a data set of New York City taxi trips[1]. Specifically, a 20-station system was created using taxi trips within Lower Manhattan (south of 14th St.) between 10 and 11am on March 1, 2012. The EMRP is solved for this system with 750 vehicles and 150 drivers ($m_v/m_d = 5$). Figure 4.4 shows the resulting availability curves and the trade-off between rebalancing rate and system performance.

Figure 4.4 shows that as the weighting factor c is increased, vehicle availability increases at the cost of an increased number of rebalancing trips up to a point where it levels off (in this case around 90%). This result compares favorably with the linear solution (4.4(d)), where at $m_v = 750$, the availabilities range from 0.84 to 0.94. In general, the linear optimization technique appears suitable for computing a first approximation of key design parameters of the system, and the nonlinear technique can be used to further refine the solution. Finally, compared to an AMoD system with the same number of vehicles (red line in Figure 4.4(d)), the overall availability is 5% lower (90% vs. 95%). This further shows that AMoD systems would achieve higher levels of performance compared to MoD systems.

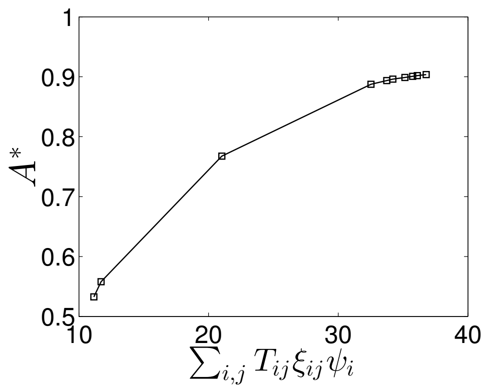
[1]Courtesy of the New York City Taxi & Limousine Commission.



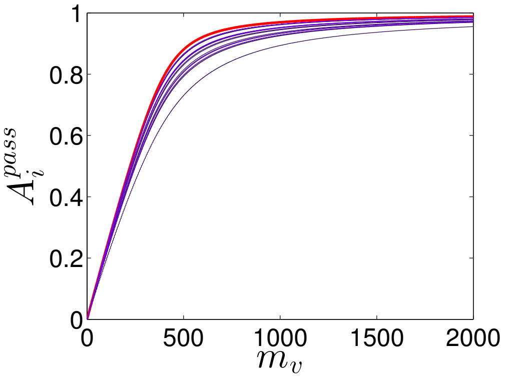
(a) $c = 1, A^* = 0.5327$



(b) $c = 10, A^* = 0.8988$



(c) Pareto optimal curve of EMRP ($c = 1, 2, 3, 4, 5, 6, 10, 15, 20, 50$)



(d) Linear solution.

Figure 4.4: Nonlinear optimization results for a 20 station system based on Lower Manhattan taxi trip data. 4.4(a) shows the optimized availability curves for $c = 1$. 4.4(b) shows the optimized availability curves for $c = 10$. 4.4(c) shows the Pareto optimal curve obtained by increasing c from 1 to 50. The x-axis can be interpreted as the average number of rebalancing vehicles on the road. 4.4(d) shows the linear optimization results for comparison.

4.3.4 Application to system sizing

Though the linear programming approach (Section 4.3.1) does not yield identical availabilities across all stations, it is nonetheless useful for applications such as fleet sizing due to its scalability and efficiency. In this section we provide a simplified example of how to use the MRP approach to gain insight into the optimal vehicle-to-driver ratio (m_v/m_d) of an MoD system. The idea is to find the optimal number of vehicles and drivers that would minimize total cost (or maximize profit) while maintaining an acceptable quality of service. For this simple example, the total cost (normalized by the cost of a vehicle) is

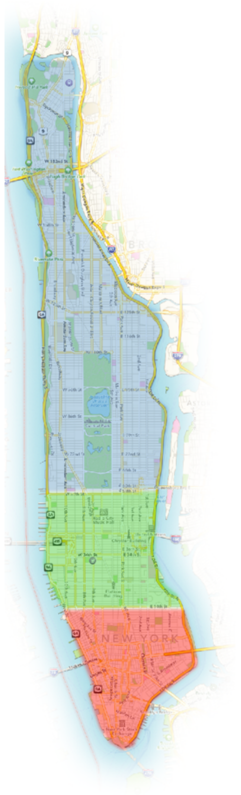
$$c_{\text{total}} = m_v + c_r m_d, \tag{4.17}$$

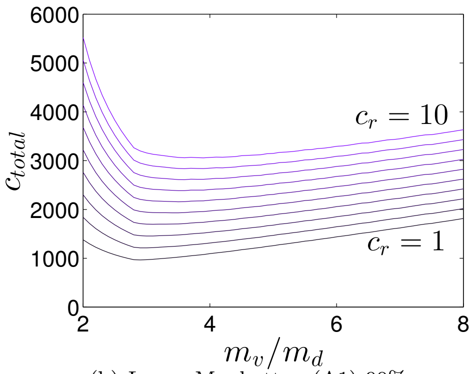
where c_r is the cost ratio between a vehicle and a driver. It is reasonable to assume that the cost of a driver is greater than the cost of a vehicle, so $c_r \geq 1$. Three MoD systems are generated using portions of the New York City taxi data: (1) Lower Manhattan (A1), (2) Midtown Manhattan (A2), and (3) Upper Manhattan (A3). Taxi trips within each region are aggregated and clustered into 20 stations, and the system parameters (λ_i, p_{ij}, and T_{ij}) are estimated. Different travel patterns in the three systems allow us to generalize our insights about the optimal m_v/m_d required to minimize cost. For each system with a fixed m_v/m_d, the MRP is solved and the number of vehicles and drivers needed are found such that the lowest availability across all the stations is greater than the availability threshold. Three availability thresholds are investigated (85%, 90%, and 95%). Figure 4.5(b) shows the total cost as it varies with the vehicle-to-driver ratio and with c_r for Lower Manhattan with 90% availability threshold. The optimal vehicle-to-driver ratio is the minimum point of each line in 4.5(b). Figure 4.5(c) shows the optimal vehicle-to-driver ratios plotted against the cost ratio c_r for all three Manhattan suburbs and all three availability thresholds.

A few insights can be gained from this example. First, the optimal m_v/m_d ratio does not significantly increase with increasing cost ratio. Second, the optimal m_v/m_d ratio decreases as the availability threshold is raised, consistent with the idea that a high quality of service requires more rebalancing, and thus more drivers. Third, the optimal m_v/m_d ratio is clearly different for each of the Manhattan suburbs (which highlights the important system-dependent nature of this value) but stays between 3 and 5 for a wide range of cost ratios. This example shows the applicability of the queueing network approach to the design and analysis of MoD systems. Similar studies can be done with the nonlinear approach, which will yield higher predictive fidelity but at a higher computational cost.

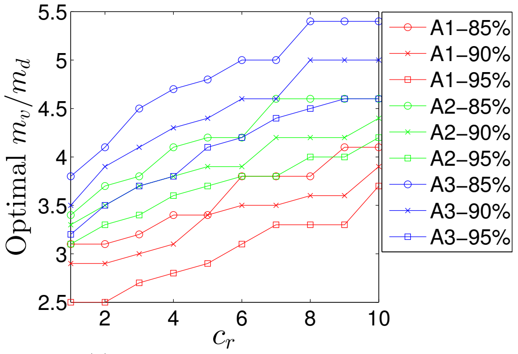
4.4 Closed-loop Control of MoD Systems

In this section we formulate a real-time closed-loop control policy by drawing inspiration from the open-loop counterparts in Section 4.3. Our closed-loop policy relies on receding horizon optimization


(a) Manhattan regions



(b) Lower Manhattan (A1)-90%



(c) Optimal vehicle-to-driver ratio

Figure 4.5: 4.5(a) shows the three suburbs of Manhattan under consideration. 4.5(b) shows the total cost as a function of the vehicle-to-driver ratio for c_r values ranging from 1 to 10. m_v and m_d values at each point in each curve can be solved using (4.17). m_v and m_d satisfies the constraint that the availability at each station is greater than the threshold of 90%. 4.5(c) shows the optimal vehicle-to-driver ratio for the 3 suburbs of Manhattan and 3 availability thresholds (85%, 90%, 95%). The curve A1-90% in 4.5(c) is constructed from the minimum points of each curve in 4.5(b). Other curves in 4.5(c) are constructed using the minima of curves similar to 4.5(b) for other Manhattan neighborhoods and availability thresholds.

and is targeted towards a practical scenario where customers would wait in line for the next available vehicle rather than leave the system. The control policy must perform two tasks: (1) rebalance vehicles throughout the network by issuing instructions to drivers, and (2) assign vehicles (with or without driver) to new customers at each station. For simplicity, as in Section 4.3, we perform these tasks *separately* by designing a vehicle rebalancing policy and a customer-assignment policy. We adapt the vehicle rebalancing policy from 3.3.3 with little modification. The customer-assignment policy is trickier, and we propose a mixed-integer linear program (MILP) to select the best assignment based on the current state of the system. The proposed policy enforces the following operation scenario for the MoD system: customers arriving at each station join a queue of "unassigned" customers. A system-wide optimization problem is solved to try to assign as many customers as possible while keeping the customer-driven vehicles balanced. Once a customer is assigned, he/she moves to the departure queue where he/she will depart with an empty vehicle or with a taxi. The optimization procedure is performed every time a departure queue is empty and there are unassigned customers. The notion of keeping the customer-driven vehicles balanced at each station stems from early studies we performed using simple heuristic policies, where we observed customer-driven vehicles aggregating at a small number of stations unused for long periods of time, effectively decreasing the number of vehicles in the system. This observation inspired the formulation of the EMRP (Section 4.3.3) as well as the real-time policy.

Let n_{ij}^v be the number of customers traveling from station i to j to be assigned to drive themselves. Let n_{ij}^d be the number of customers traveling from station i to j to be assigned to a taxi. Denote by v_i^e the number of excess unassigned customer-driven vehicles at station i, v_{ji}^t the number of customer-driven vehicles traveling from station j to i, and v_{ji}^a the number of customer-driven vehicles at station j assigned to travel to station i but that have not yet left the station. Assuming these quantities are known, the number of customer-driven vehicles at a future time step is

$$v_i^+ = v_i^e + \sum_j (v_{ji}^a + v_{ji}^t + n_{ji}^v - n_{ij}^v).$$

We can define a desired vehicle distribution to be, for example,

$$v_i^{des} = \left\lfloor \frac{(m_v - m_d)\lambda_i}{\sum_j \lambda_j} \right\rfloor .$$

The assignment policy is given by solving the following optimization problem

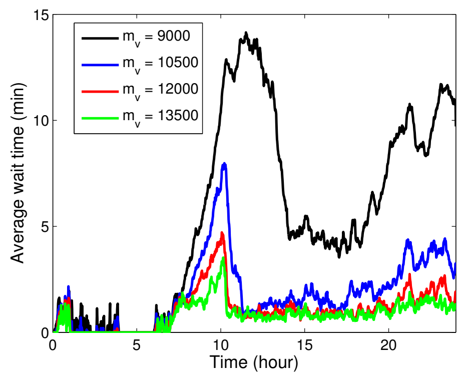
$$\begin{aligned}
\underset{n_{ij}^d, n_{ij}^v}{\text{minimize}} \quad & \sum_i |v_i^+ - v_i^{des}| - w \sum_{i,j} (n_{ij}^d + n_{ij}^v) \\
\text{subject to} \quad & n_{ij}^d + n_{ij}^v \leq c_{ij}^u \\
& \sum_j n_{ij}^v \leq v_i^e, \quad \sum_j n_{ij}^d \leq d_i^u \\
& n_{ij}^v \geq 0,\ n_{ij}^d \geq 0,\ n_{ij}^v \in \mathbb{N},\ n_{ij}^d \in \mathbb{N},
\end{aligned} \tag{4.18}$$

where c_{ij}^u is the number of unassigned customers traveling from i to j, d_i^u is the number of unassigned drivers at station i, and w is a weighting factor. The objective function trades off the relative importance of system balance and customer wait times (increasing w would allow the system to assign more customers and reduce wait times). The constraints ensure that the assignment policy is feasible (there are enough vehicles, drivers, and customers). Problem (4.18) is formulated as a MILP and solved using the IBM CPLEX solver [71].

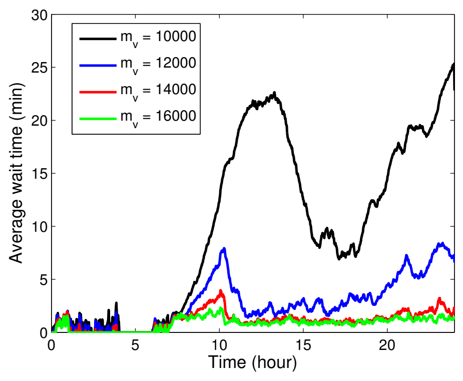
4.5 Case Study: MoD in Manhattan

We assess the performance and the scalability of the real-time control policy through simulation of an MoD system operating over all of Manhattan (as opposed to small regions in Section 4.3.4). We place 100 stations within Manhattan based on k-means clustering of taxi requests. The resulting station positions are such that customer demands are on average less than 300 meters from the nearest station. For each hour of the day, customer arrival rates λ_i and routing probabilities p_{ij} are estimated by counting the number of trips that originate and end at each station, as done in Chapter 3.4. The travel times T_{ij} are estimated from the average vehicle speed and the Manhattan distance between stations. Customer arrivals are generated at each station as a Poisson process with mean λ_i and routing probabilities p_{ij}. In contrast to the analysis in the previous section, simulated customers wait at the station until a vehicle has been assigned to them.

Simulations are performed for vehicle-to-driver ratios of 3 and 4, for 24 hours with a time step of 6 seconds. The results of the simulations are shown in Figure 4.6. For a vehicle-to-driver ratio of 3, a system consisting of 12,000 vehicles and 4,000 drivers yielded a maximum average wait time of under 5 minutes, which is indicative of adequate service. For a vehicle-to-driver ratio of 4, satisfactory quality of service is reached between 12,000 and 14,000 vehicles (3,000 to 3,500 drivers). For comparison, New York City has over 13,000 taxis, and 85% of trips are within Manhattan. This suggests that by operating a fraction of the vehicles as a taxi service to maintain system balance, an MoD system can achieve comparable quality of service to taxi systems with only 1/4 to 1/3 the number of drivers. The driver assignment optimization problem in the simulations was solved in an average of 0.5 seconds. Since the problem size only scales with the number of stations and the constraints consist mostly of bounding hyperplanes, the feasible set is easy to compute and the problem can be solved in real-time for large-scale systems.

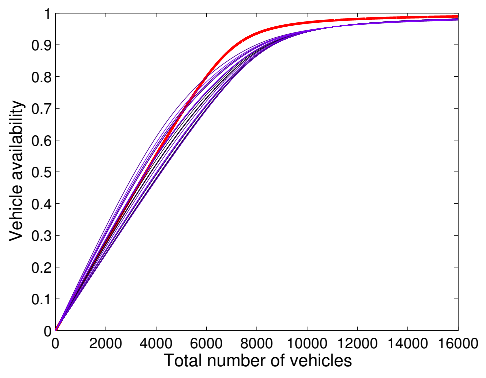


(a) $m_v/m_d = 3$

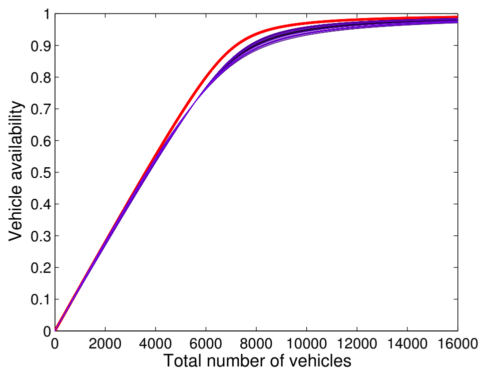


(b) $m_v/m_d = 4$

Figure 4.6: Average customer wait times throughout the day.



(a) $m_v/m_d = 3$



(b) $m_v/m_d = 4$

Figure 4.7: Vehicle availability during peak demand. Availability curves plotted for 20 out of the 100 stations.

We also compare the simulation results to the queueing network analysis in Section 4.3. Using the linear approximate rebalancing analysis presented in Section 4.3.1, we study the availability of vehicles in Manhattan during the peak period from 9 to 10 am. The availability curves are shown in Figure 4.7 for vehicle-to-driver ratios of 3 and 4, and compared to a balanced taxi or autonomous vehicle system (shown in red). According to this analysis, for a MoD system to achieve the same theoretical performance as an AMoD system with 8,000 vehicles (see Chapter 3.4), 9,789 vehicles are needed for a vehicle-to-driver ratio of 3, and 10,267 vehicles are needed for a vehicle-to-driver ratio of 4. This corroborates well with simulation results in Figure 4.6, which suggests that between

11,000 and 12,000 vehicles are needed.

4.6 Conclusion

In this chapter we studied a queueing network model of an MoD system and developed two open-loop control approaches useful for design tasks such as system sizing. We applied such approaches to a system sizing example for three Manhattan neighborhoods, which showed that the optimal vehicle-to-driver ratio is between 3 and 5. Drawing insights from these techniques, we developed a closed-loop real-time control policy for driver assignment that can be applied to large systems. We showed that an MoD system operating in Manhattan can adequately service all taxi demands with the same number of vehicles but require only 1/4 to 1/3 the number of drivers.

Part II

AMoD With Structural and Operational Constraints

Chapter 5

Congestion-aware AMoD

5.1 Introduction

The queueing-theoretical methods studied in Chapters 3 and 4 model travel on a complete graph of origins and destinations (point-to-point travel) rather than on a road network. The roads were assumed to be infinite-server queues which does not consider congestion effects.

While we have seen several potential benefits of AMoD systems, there has been no consensus on whether these systems will ultimately be beneficial or detrimental in terms of traffic congestion. It has been argued that by having faster reaction times, autonomous vehicles may be able to drive faster and follow other vehicles at closer distances without compromising safety, thereby effectively increasing the capacity of a road and reducing congestion. They may also be able to interact with traffic lights to reduce full stops at intersections [72]. On the downside, the process of vehicle rebalancing (empty vehicle trips) increases the total number of vehicles on the road (assuming the number of vehicles with customers stays the same). Indeed, it has been argued that the presence of many rebalancing vehicles may contribute to an *increase* in congestion [73, 74, 75]. These statements, however, do not take into account the fact that in an AMoD system the operator has control over the actions (destination and routes) of the vehicles, and may route vehicles intelligently to avoid increasing congestion or perhaps even decrease it.

Accordingly, the goal of this chapter is twofold. First, on an engineering level, we wish to devise routing and rebalancing algorithms for an autonomous vehicle fleet that seek to minimize congestion. Second, on a socio-economic level, we aim to rigorously address the concern that autonomous cars may lead to increased congestion and thus disrupt currently congested transportation infrastructures. We begin with a brief summary of existing work on congestion.

Traffic congestion has been studied in economics and transportation for nearly a century. The first congestion models [24, 76, 77] sought to formalize the relationship between vehicle speed, density, and flow. Since then, approaches to modeling congestion have included empirical [68], simulation-based [27, 26, 78], queueing-theoretical [79], and optimization [80, 81]. While there have been many high fidelity congestion models that can accurately predict traffic patterns, the primary goal of congestion modeling has been the *analysis* of traffic behavior. Efforts to *control* traffic have been limited to the control of intersections [82, 83] and freeway on-ramps [84] because human drivers behave non-cooperatively. The problem of cooperative, system-wide routing (a key benefit of AMoD systems) is similar to the dynamic traffic assignment problem (DTA) [81]. The key difference is that DTA approaches only optimize routes for passenger vehicles while we seek to optimize the routes of *both* passenger vehicles *and* empty rebalancing vehicles. Since congestion is a highly complex and nonlinear phenomenon, mathematical models of congestion tend to be quite involved. Instead of explicitly modeling the flow of traffic in congestion, we seek conditions and algorithmic approaches that would result in a congestion-free solution in the road network.

Specifically the contributions of this chapter are as follows. First, we model an AMoD system within a network flow framework, whereby customer-carrying and empty rebalancing vehicles are represented as flows over a *capacitated* road network (in such model, when the flow of vehicles along a road reaches a critical capacity value, congestion effects occur). Within this model, we provide a cut condition for the road graph that needs to be satisfied for congestion-free customer and rebalancing flows to exist. Most importantly, under the assumption of a *symmetric* road network, we investigate an existential result that leads to two key conclusions: (1) rebalancing does not increase congestion, and (2) for certain cost functions, the problems of finding customer and rebalancing flows can be decoupled. Second, leveraging the theoretical insights, we propose a computationally-efficient algorithm for congestion-aware routing and rebalancing of an AMoD system that is broadly applicable to time-varying, possibly asymmetric road networks. Third, through numerical studies on real-world traffic data, we validate our assumptions and show that the proposed real-time routing and rebalancing algorithm outperforms point-to-point rebalancing algorithms in terms of lower customer wait times by avoiding excess congestion on the road.

This chapter is organized as follows. In Section 5.2 we perform a simple numerical study based on the theory developed in Chapter 3 to empirically assess the impact of rebalancing on congestion. In Section 5.3 we introduce a network flow model of an AMoD system on a capacitated road network and formulate the simultaneous routing and rebalancing problem. In Section 5.4 we present key structural properties of the model including fundamental limitations of performance and conditions for the existence of feasible (in particular, congestion-free) solutions. The insights from Section 5.4 are used to develop a practical real-time routing and rebalancing algorithm in Section 5.5. Numerical studies and simulation results for a simplified road network of Manhattan are presented in Section 5.6.

5.2 Numerical study of congestion effects

Leveraging ideas from the queueing analysis in Chapter 3, we make the key observation that the throughput of each station (rate of vehicles leaving each station) is the total arrival rate of customers multiplied by the availability, so in systems with a high quality of service, the actual throughput Λ_i can be closely approximated by the customer arrival rates $\tilde{\lambda}_i$. In this section, we use a simple 9-station road network (shown in Figure 5.1) along with this insight to illustrate the impact of rebalancing vehicles on road network congestion.

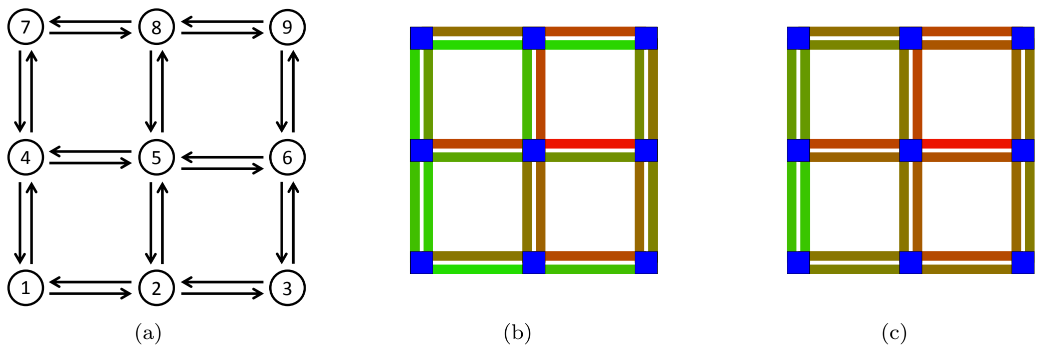


Figure 5.1: Left: Layout of the 9-station road network. Each road segment has a capacity of 40 vehicles in each direction. Center: A randomly generated system on the 9-station road network without rebalancing. The shade on each road segment indicates the level of congestion, where green is no congestion, and red is heavy congestion. Right: The same road network with rebalancing vehicles.

The stations are placed on a square grid, and joined by 2-way road segments each of which is 0.5 km long. Each road consists of a single lane, with a critical density of 80 vehicles/km.[1] This means that the capacity of each road segment (u, v) is $c(u, v) = 40$ vehicles. Each vehicle travels at 30 km/h (8.33 m/s) in free flow, which means the travel time along each road segment is 1 minute in free flow.

To gain insight into the general system behavior, a variety of systems with different levels of imbalance must be studied. First, arrival rates and routing distributions are randomly generated and rebalancing rates are computed using (3.9). In steady state, the fraction of vehicles in each road queue ij is given by $\pi_i \tilde{p}_{ij}$ (Lemma 3.3.1). If we assume 100% availability ($A_i = 1$), the expected rate of vehicles entering each road queue is given by $\Lambda_{ij} = \lambda_i p_{ij}$. Using Little's theorem, the expected number of vehicles on each road queue is given by $L_{ij} = \Lambda_{ij} T_{ij}$. The availability assumption can be justified by the fact that a real system would operate within the regime of high availability and that the number of vehicles on the road gets very close to L_{ij} as availability increases. Similarly, the expected number of rebalancing vehicles on each road queue is given by $L_{ij}^{\text{reb}} = \beta_{ij} T_{ij}$ (where β_{ij} is

[1]If each vehicle is 5 m, this critical density represents a vehicle-to-vehicle separation of 1.5 car-lengths.

defined in Chapter 3.3).

To map the queueing network onto the road network, we adopt a similar procedure as the one used to estimate q_{ij} in Section 3.7. Recall that $\mathcal{P}_{ij}$ is the set of paths from station i to station j. We adopt the routing strategy that *uniformly* distributes vehicles from i to j along each path $\varphi_{ij} \in \mathcal{P}_{ij}$. The number of vehicles that go through each road segment, $L^{\text{road}}(u,v)$, is then the sum of the number of vehicles from each station to every other station that pass through the road segment, given by

$$L^{\text{road}}(u,v) = \sum_{i,j} \sum_{\varphi_{ij} \in \mathcal{P}_{ij} \text{ s.t. } (u,v) \in \varphi_{ij}} \frac{L_{ij}}{|\mathcal{P}_{ij}|}.$$

Note that for stability, $L^{\text{road}}(u,v) < c(u,v)$. The road utilization is given by

$$\rho^{\text{road}}(u,v) = \frac{L^{\text{road}}(u,v)}{c(u,v)}.$$

Figure 5.2(a) plots the vehicle and road utilization increases due to rebalancing for 500 randomly generated systems. The x-axis shows the ratio of rebalancing vehicles to passenger vehicles on the road, which represents the inherent imbalance in the system. The red dots represent the increase in average road utilization due to rebalancing and the blue x's represent the utilization increase in the most congested road segment due to rebalancing. It is no surprise that the average road utilization rate is a linear function of the number of rebalancing vehicles. However, remarkably, the maximum congestion increases are much lower than the average, and are in most cases, zero. This means that while rebalancing generally increases the number of vehicles on the road, rebalancing vehicles mostly travel along less congested routes and rarely increase the maximum congestion in the system. This can be seen in Figure 5.1 right, where rebalancing clearly increases the number of vehicles on many roads but not on the most congested road segment (from Station 6 to Station 5).

In roughly 10% of the systems simulated, the maximum road utilization in the system increased from rebalancing (Figure 5.2(a)). This may cause heavy congestion in systems where congestion is already prevalent. Though different routing strategies may help decrease the maximum road utilization, we can also adjust the rebalancing rates by using a "corrected" travel time in the rebalancing optimization (3.9) that takes into account the congestion in the system. For this example, the travel time is calculated using the Bureau of Public Roads model, a simple relation between road utilization and travel time [85]. The corrected travel time along road segment (u,v) is

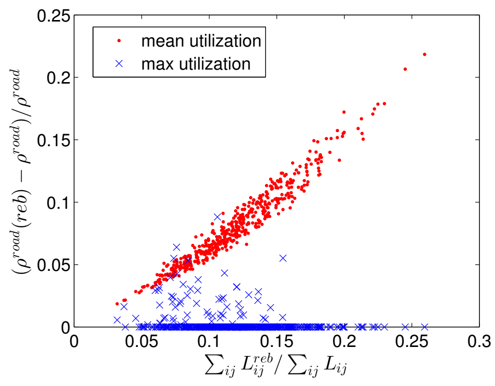
$$T^d(u,v) = T(u,v)(1 + 0.15\bar{\rho}^4(u,v)),$$

where $T(u,v)$ is the free flow travel time, $\bar{\rho}(u,v) = L^{\text{road}}(u,v)/\bar{L}^{\text{road}}(u,v)$, and $\bar{L}^{\text{road}}(u,v)$ is the mean number of vehicles on each road segment. The intuition here is to penalize rebalancing trips on routes with higher-than-average utilization. The corrected mean travel time from station i to j

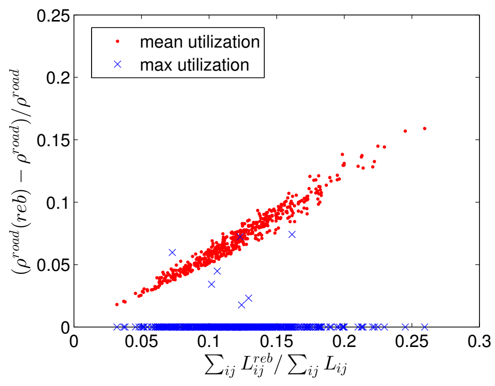
is then

$$T^d_{ij} = \sum_{\varphi_{ij} \in \mathcal{P}_{ij}} a^{\varphi_{ij}}_{ij} \sum_{(u,v)\in\varphi_{ij}} T^d(u,v),$$

where $a^{\varphi_{ij}}_{ij}$ is the fraction of trips from i to j taking path φ_{ij}. In this example, since trips are divided evenly between all paths, $a^{\varphi_{ij}}_{ij} = 1/|\mathcal{P}_{ij}|$. The rebalancing rates β_{ij} are then recalculated from (3.9) using T^d_{ij} instead of T_{ij}.



(a)



(b)

Figure 5.2: 5.2(a): The effects of rebalancing on congestion on a 9-station road network illustrated using 500 randomly generated systems with different arrival rates and routing distributions. The x-axis is the ratio of rebalancing vehicles to passenger vehicles on the road. The y-axis is the fractional increase in road utilization due to rebalancing. The dots show the mean increase in road utilization. The crosses show the increase in utilization of the most congested road segment. 5.2(b): The same 500 systems with rebalancing rates calculated using a "corrected" travel time that takes into account road utilization.

Figure 5.2(b) shows the impact of rebalancing on road utilization with the rebalancing rates adjusted using the method described above. Comparing with Figure 5.2(a), we see that the new rebalancing scheme reduced the number of systems with increased maximum road utilization. Of the 500 simulated systems, only 7 (1.4%) saw an increase in the maximum road utilization. Remarkably, the mean road utilization increase due to rebalancing was reduced as well. This shows that by solely adjusting the rebalancing parameters β_{ij} we can almost always prevent additional congestion in the most congested parts of the system. Additional numerical results strengthening this fact can be found in Appendix A.

In the few rare cases where maximum road utilization does increase, an intelligent routing strategy becomes crucial. While uniform routing along different paths helps distribute vehicles throughout the road network, a better routing strategy would actively route vehicles away from congested roads, limit rebalancing when it may cause further delays, and perhaps even stagger passenger trips to *reduce* congestion. This motivates the study of the combined routing and rebalancing problem on a congested road network. We now proceed to develop a rigorous model for this problem.

5.3 Network Flow Model

In this section we formulate a network flow model for an AMoD system operating over a capacitated road network. The model allows us to derive key structural insights into the vehicle routing and rebalancing problem, and motivates the design of real-time, congestion-aware algorithms for coordinating the autonomous vehicles. We start in Section 5.3.1 with a discussion of our congestion model; then, in Section 5.3.2 we provide a detailed description of the overall AMoD system model.

5.3.1 Congestion Model

We use a simplified congestion model consistent with classical traffic flow theory [24]. In classical traffic flow theory, at low vehicle densities on a road link, vehicles travel at the free flow speed of the road (imposed by the speed limit). This is referred to as the free flow phase of traffic. In this phase, the free flow speed is approximately constant [86]. The flow, or flow rate, is the number of vehicles passing through the link per unit time, and is given by the product of the speed and density of vehicles. When the flow of vehicles reaches an empirically observed critical value, the flow reaches its maximum. Beyond the critical flow rate, vehicle speeds are dramatically reduced and the flow decreases, signaling the beginning of traffic congestion. The maximum stationary flow rate is called the *capacity* of the road link in the literature. In our approach, road capacities are modeled as *constraints on the flow of vehicles*. In this way, the model captures the behavior of vehicles up to the onset of congestion.

This simplified congestion model is adequate for our purposes because the goal is not to analyze the behavior of vehicles in congested networks, but to control vehicles in order to avoid the onset of congestion. We also do not explicitly model delays at intersections, spillback behavior due to congestion, or bottleneck behavior due to the reduction of the number of lanes on a road link. An extension to this model that accommodates (limited) congestion on links is presented in Section 5.6.1.

5.3.2 Network Flow Model of AMoD system

We consider a road network modeled as a directed graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set. Figure 5.3 shows one such network. The nodes v in $\mathcal{V}$ represent intersections and locations for trip origins/destinations, and the edges (u, v) in $\mathcal{E}$ represent road links. As discussed in Section 5.3.1, congestion is modeled by imposing capacity constraints on the road links: each constraint represents the capacity of the road upon the onset of congestion. Specifically, for each road link $(u, v) \in \mathcal{E}$, we denote by $c(u, v) : \mathcal{E} \mapsto \mathbb{N}_{>0}$ the capacity of that link.

When the flow rate on a road link is less than the capacity of the link, all vehicles are assumed to travel at the free flow speed, or the speed limit of the link. For each road link $(u, v) \in \mathcal{E}$, we denote by $T(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}$ the corresponding free flow time required to traverse road link (u, v). Conversely, when the flow rate on a road link is larger than the capacity of the link, the traversal time is assumed equal to ∞ (we reiterate that the focus in this section is on avoiding the onset of congestion).

We assume that the road network is *capacity-symmetric* (or symmetric for short): for any cut[2] $(\mathcal{S}, \bar{\mathcal{S}})$ of $G(\mathcal{V}, \mathcal{E})$, the overall capacity of the edges connecting nodes in $\mathcal{S}$ to nodes in $\bar{\mathcal{S}}$ equals the overall capacity of the edges connecting nodes in $\bar{\mathcal{S}}$ to nodes in $\mathcal{S}$, that is

$$\sum_{(u,v)\in\mathcal{E}:\ u\in\mathcal{S},\, v\in\bar{\mathcal{S}}} c(u, v) = \sum_{(v,u)\in\mathcal{E}:\ u\in\mathcal{S},\, v\in\bar{\mathcal{S}}} c(v, u)$$

It is easy to verify that a network is capacity-symmetric if and only if the overall capacity entering each *node* equals the capacity exiting each node., i.e.

$$\sum_{u\in\mathcal{V}:(u,v)\in\mathcal{E}} c(u, v) = \sum_{w\in\mathcal{V}:(v,w)\in\mathcal{E}} c(v, w).$$

If all *edges* have symmetrical capacity, i.e., for all $(u, v) \in \mathcal{E}$, $c(u, v) = c(v, u)$, then the network is capacity-symmetric. The converse statement, however, is not true in general.

Transportation requests are described by the tuple (s, t, λ), where $s \in \mathcal{V}$ is the origin of the requests, $t \in \mathcal{V}$ is the destination, and $\lambda \in \mathbb{R}_{>0}$ is the rate of requests, in customers per unit time. Transportation requests are assumed to be stationary and deterministic, i.e., the rate of requests does not change with time and is a deterministic quantity. The set of transportation requests is denoted by $\mathcal{K} = \{(s_k, t_k, \lambda_k)\}_k$, and its cardinality is denoted by K.

Single-occupancy vehicles travel within the network while servicing the transportation requests. We denote $f_k(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}$, $k = \{1, \ldots, K\}$, as the *customer flow* for requests k on edge (u, v), i.e., the amount of flow from origin s_k to destination t_k that uses link (u, v). We also denote $f_R(u, v) : \mathcal{E} \mapsto \mathbb{R}_{\geq 0}$ as the *rebalancing flow* on edge (u, v), i.e., the amount of rebalancing flow traversing edge (u, v) needed to realign the vehicles with the asymmetric distribution of transportation requests.

5.3.3 The Routing Problem

The goal is to compute flows for the autonomous vehicles that (1) transfer customers to their desired destinations in minimum time (customer-carrying trips) and (2) rebalance vehicles throughout the network to realign the vehicle fleet with transportation demand (customer-empty trips). Specifically, the *Congestion-free Routing and Rebalancing Problem (CRRP)* is formally defined as follows. Given

[2]For any subset of nodes $\mathcal{S} \subseteq \mathcal{V}$, we define a *cut* $(\mathcal{S}, \bar{\mathcal{S}}) \subseteq \mathcal{E}$ as the set of edges whose origin lies in $\mathcal{S}$ and whose destination lies in $\bar{\mathcal{S}} = \{\mathcal{V} \setminus \mathcal{S}\}$. Formally, $(\mathcal{S}, \bar{\mathcal{S}}) := \{(u, v) \in \mathcal{E} : u \in \mathcal{S}, v \in \bar{\mathcal{S}}\}$.

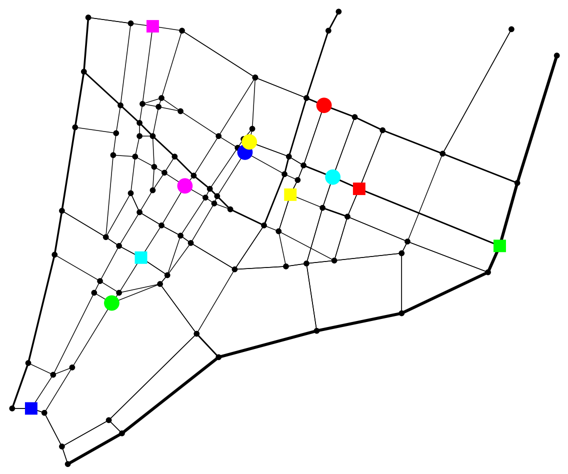


Figure 5.3: A road network modeling Lower Manhattan and the Financial District. Nodes (denoted by small black dots) model intersections; select nodes, denoted by colored circular and square markers, model passenger trips' origins and destinations. Different trip requests are denoted by different colors. Roads are modeled as edges; line thickness is proportional to road capacity.

a capacitated, symmetric network $G(\mathcal{V},\mathcal{E})$, a set of transportation requests $\mathcal{K} = \{(s_k, t_k, \lambda_k)\}_k$, and a weight factor $\rho > 0$, solve

$$\underset{f_k(\cdot,\cdot),f_R(\cdot,\cdot)}{\text{minimize}} \quad \sum_{k\in\mathcal{K}}\sum_{(u,v)\in\mathcal{E}} T(u,v)f_k(u,v) + \rho \sum_{(u,v)\in\mathcal{E}} T(u,v)f_R(u,v) \tag{5.1}$$

$$\text{subject to} \quad \sum_{u\in\mathcal{V}} f_k(u,s_k) + \lambda_k = \sum_{w\in\mathcal{V}} f_k(s_k,w) \quad \text{for all } k\in\mathcal{K} \tag{5.2}$$

$$\sum_{u\in\mathcal{V}} f_k(u,t_k) = \lambda_k + \sum_{w\in\mathcal{V}} f_k(t_k,w) \quad \text{for all } k\in\mathcal{K} \tag{5.3}$$

$$\sum_{u\in\mathcal{V}} f_k(u,v) = \sum_{w\in\mathcal{V}} f_k(v,w) \quad \text{for all } k\in\mathcal{K}, v\in\mathcal{V}\setminus\{s_k,t_k\} \tag{5.4}$$

$$\sum_{u\in\mathcal{V}} f_R(u,v) + \sum_{k\in\mathcal{K}} 1_{v=t_k}\lambda_k = \sum_{w\in\mathcal{V}} f_R(v,w) + \sum_{k\in\mathcal{K}} 1_{v=s_k}\lambda_k \quad \text{for all } v\in\mathcal{V} \tag{5.5}$$

$$f_R(u,v) + \sum_{k\in\mathcal{K}} f_k(u,v) \le c(u,v) \quad \text{for all } (u,v)\in\mathcal{E} \tag{5.6}$$

The cost function (5.1) is a weighted sum (with weight ρ) of the overall duration of all passenger trips and the duration of rebalancing trips. Constraints (5.2), (5.3) and (5.4) enforce continuity of each trip (i.e., flow conservation) across nodes. Constraint (5.5) ensures that vehicles are rebalanced throughout the road network to re-align vehicle distribution with transportation requests, i.e. to ensure that every outbound customer flow is matched by an inbound flow of rebalancing vehicles and vice versa. Finally, Constraint (5.6) enforces the capacity constraint on each link (function 1_x denotes the indicator function of the Boolean variable $x = \{\text{true, false}\}$, that is 1_x equals one if x is true, and equals zero if x is false). Note that the CRRP is a linear program and, in particular, a special instance of the fractional multi-commodity flow problem [59] (see Chapter 2.2).

We denote a customer flow $\{f_k(u,v)\}_{(u,v),k}$ that satisfies Equations (5.2), (5.3), (5.4) and (5.6) as a *feasible customer flow.* For a given set of feasible customer flows $\{f_k(u,v)\}_{(u,v),k}$, we denote a flow $\{f_R(u,v)\}_{(u,v)}$ that satisfies Equation (5.5) and such that the combined flows $\{f_k(u,v), f_R(u,v)\}_{(u,v),k}$ satisfy Equation (5.6) as a *feasible rebalancing flow.* We remark that a rebalancing flow that is feasible with respect to a set of customer flows may be infeasible for a different collection of customer flows.

For a given set of optimal flows $\{f_k^*(u,v)\}_{(u,v),k}$ and $\{f_R^*(u,v)\}_{(u,v)}$, the minimum number of vehicles needed to implement them is given by

$$V_{\min} = \left\lceil \sum_{k\in\mathcal{K}} \sum_{(u,v)\in\mathcal{E}} T(u,v)\Big(f_k^*(u,v) + f_R^*(u,v)\Big) \right\rceil .$$

This follows from a similar analysis done in [21] for point-to-point networks. Hence, the cost function (5.1) is aligned with the desire of minimizing the number of vehicles needed to operate an AMoD system.

5.3.4 Discussion

A few comments are in order. First, we assume that transportation requests are time invariant. This assumption is valid when transportation requests change slowly with respect to the average duration of a customer's trip, which is often the case in dense urban environments [87]. Additionally, in Section 5.5 we will present algorithmic tools that allow one to extend the insights gained from the time-invariant case to the time-varying counterpart. Second, the assumption of single-occupancy for the vehicles models most of the existing (human) one-way vehicle sharing systems (where the driver is considered "part" of the vehicle), and chiefly disallows the provision of ride-sharing or carpooling service (this is an aspect left for future research). Third, as also discussed in Section 5.3.1, our congestion model is simpler and less accurate than typical congestion models used in the transportation community. However, our model lends itself to efficient real-time optimization and thus it is well-suited to the *control* of fleets of autonomous vehicles. Existing high-fidelity congestion models should be regarded as complementary and could be used offline to identify the congestion thresholds used in our model. Fourth, while we have defined the CRRP in terms of fractional flows, an integer-valued counterpart can be defined and (approximately) solved to find optimal routes for each *individual* customer and vehicle. Algorithmic aspects will be investigated in depth in Section 5.5, with the goal of devising practical, real-time routing and rebalancing algorithms. Fifth, trip requests are assumed to be known. In practice, trip requests can be reserved in advance, estimated from historical data, or estimated in real time. Finally, the assumption of capacity-symmetric road networks indeed appears reasonable for a number of major U.S. metropolitan areas. In Appendix C, by using OpenStreetMap data [88], we provide a rigorous characterization in terms of capacity symmetry of the road networks of New York City, Chicago, Los Angeles and other major U.S. cities. The results consistently show that urban road networks are usually symmetric to a *very high* degree. Additionally, several of the theoretical and algorithmic results extend to the case where this assumption is lifted, as it will be highlighted throughout this chapter.

5.4 Structural Properties of the Network Flow Model

In this section we provide two key structural results for the network flow model presented in Section 5.3.2. First, we provide a cut condition that needs to be satisfied for feasible customer and rebalancing flows to exist. In other words, this condition provides a fundamental limitation of performance for congestion-free AMoD service in a given road network. Second, we investigate an existential result (our main theoretical result) that is germane to two key conclusions: (1) rebalancing does not increase congestion in symmetric road networks, and (2) for certain cost functions, the problems of finding customer and rebalancing flows can be *decoupled* – an insight that will be heavily exploited in subsequent sections.

5.4.1 Fundamental Limitations

We start with a few definitions. For a given set of feasible customer flows $\{f_k(u,v)\}_{(u,v),k}$, we denote by $F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}})$ the overall flow exiting a cut $(\mathcal{S},\bar{\mathcal{S}})$, i.e., $F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) := \sum_{k\in\mathcal{K}}\sum_{u\in\mathcal{S},v\in\bar{\mathcal{S}}} f_k(u,v)$. Similarly, we denote by $C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}})$ the capacity of the network exiting $\mathcal{S}$, i.e., $C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) := \sum_{u\in S,v\in\bar{\mathcal{S}}} c(u,v)$. Analogously, $F_{\text{in}}(\mathcal{S},\bar{\mathcal{S}})$ denotes the overall flow entering $\mathcal{S}$ from $\bar{\mathcal{S}}$, i.e., $F_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) := F_{\text{out}}(\bar{\mathcal{S}},\mathcal{S})$, and $C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}})$ denotes the capacity entering $\mathcal{S}$ from $\bar{\mathcal{S}}$, i.e., $C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) := C_{\text{out}}(\bar{\mathcal{S}},\mathcal{S})$. We highlight that the arguments leading to the main result of this subsection (Theorem 5.4.4) do not require the assumption of capacity symmetry; hence, Theorem 5.4.4 holds for *asymmetric* road networks as well.

The next technical lemma shows that the net flow leaving set $\mathcal{S}$ equals the difference between the flow originating from the origins s_k in $\mathcal{S}$ and the flow exiting through the destinations t_k in $\mathcal{S}$, that is,

Lemma 5.4.1 (Net flow across a cut). *Consider a set of feasible customer flows $\{f_k(u,v)\}_{(u,v),k}$. Then, for every cut $(\mathcal{S},\bar{\mathcal{S}})$, the net flow leaving set $\mathcal{S}$ satisfies*

$$F_{out}(\mathcal{S},\bar{\mathcal{S}}) - F_{in}(\mathcal{S},\bar{\mathcal{S}}) = \sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}}\lambda_k - \sum_{k\in\mathcal{K}} 1_{t_k\in\mathcal{S}}\lambda_k.$$

Proof. We compute the sum over all customer flows $k \in \mathcal{K}$ and over all nodes $v \in \mathcal{V}$ of the node balance equation for flow k at node v (Equation (5.3) if node v is the source of k, Equation (5.4) if node v is the sink of k, or Equation (5.2) otherwise). We obtain

$$\sum_{v\in\mathcal{S}}\sum_{k\in\mathcal{K}}\left(\sum_{u\in\mathcal{V}} f_k(u,v) + 1_{v=s_k}\lambda_k\right) = \sum_{v\in\mathcal{S}}\sum_{k\in\mathcal{K}}\left(\sum_{w\in\mathcal{V}} f_k(v,w) + 1_{v=t_k}\lambda_k\right).$$

For any edge (u,v) such that $u,v \in \mathcal{S}$, the customer flow $f_k(u,v)$ appears on both sides of the equation. Thus the equation above simplifies to

$$\sum_{v\in\mathcal{S}}\sum_{k\in\mathcal{K}}\left(\sum_{u\in\bar{\mathcal{S}}} f_k(u,v) + 1_{v=s_k}\lambda_k\right) = \sum_{v\in\mathcal{S}}\sum_{k\in\mathcal{K}}\left(\sum_{w\in\bar{\mathcal{S}}} f_k(v,w) + 1_{v=t_k}\lambda_k\right),$$

which leads to the claim of the lemma

$$F_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) + \sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}}\lambda_k = F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) + \sum_{k\in\mathcal{K}} 1_{t_k\in\mathcal{S}}\lambda_k.$$

□

We now state two additional lemmas providing, respectively, lower and upper bounds for the outflows $F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}})$.

Lemma 5.4.2 (Lower bound for outflow)**.** *Consider a set of feasible customer flows $\{f_k(u,v)\}_{(u,v),k}$. Then, for any cut $(\mathcal{S},\bar{\mathcal{S}})$, the overall flow $F_{out}(\mathcal{S},\bar{\mathcal{S}})$ exiting cut $(\mathcal{S},\bar{\mathcal{S}})$ is lower bounded according to*

$$\sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S},t_k\in\bar{\mathcal{S}}}\lambda_k \le F_{out}(\mathcal{S},\bar{\mathcal{S}}).$$

Proof. Adding Equations (5.2), (5.3) and (5.4) over all nodes in $\mathcal{S}$ and over all flows whose origin is in $\mathcal{S}$ and whose destination is in $\bar{\mathcal{S}}$, one obtains

$$\sum_{k:s_k\in\mathcal{S},t_k\in\bar{\mathcal{S}}}\ \sum_{v\in\mathcal{S}}\left(\sum_{u\in\mathcal{V}} f_k(u,v) + 1_{v=s_k}\lambda_k\right) = \sum_{k:s_k\in\mathcal{S},t_k\in\bar{\mathcal{S}}}\ \sum_{v\in\mathcal{S}}\left(\sum_{w\in\mathcal{V}} f_k(v,w)\right).$$

Flows $f_k(u,v)$ such that both u and v are in $\mathcal{S}$ appear on both sides of the equation. Simplifying, one obtains

$$\sum_{k:s_k\in\mathcal{S},t_k\in\bar{\mathcal{S}}}\lambda_k = \sum_{k:s_k\in\mathcal{S},t_k\in\bar{\mathcal{S}}}\left(\sum_{v\in\mathcal{S},w\in\bar{\mathcal{S}}} f_k(v,w) - \sum_{v\in\mathcal{S},u\in\bar{\mathcal{S}}} f_k(u,v)\right)$$

The first term on the right-hand side represents a lower bound for $F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}})$, since

$$\begin{aligned} F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) &= \sum_{k\in\mathcal{K}} \sum_{v\in\mathcal{S}, w\in\bar{\mathcal{S}}} f_k(v,w) \\ &\geq \sum_{k:s_k\in\mathcal{S}, t_k\in\bar{\mathcal{S}}} \sum_{v\in\mathcal{S}, w\in\bar{\mathcal{S}}} f_k(v,w). \end{aligned}$$

Furthermore, the second term on the right-hand side is upper-bounded by zero. The lemma follows. □

Lemma 5.4.3 (Upper bound for outflow)**.** *Assume there exists a set of* feasible *customer and rebalancing flows* $\{f_k(u,v),\ f_R(u,v)\}_{(u,v),k}$. *Then, for every cut* $(\mathcal{S}, \bar{\mathcal{S}})$*,*

1. $F_{out}(\mathcal{S}, \bar{\mathcal{S}}) \leq C_{out}(\mathcal{S}, \bar{\mathcal{S}})$*, and*

2. $F_{out}(\mathcal{S}, \bar{\mathcal{S}}) \leq C_{in}(\mathcal{S}, \bar{\mathcal{S}})$*.*

Proof. The first condition follows trivially from Equation (5.6). As for the second condition, consider a cut $(\mathcal{S}, \bar{\mathcal{S}})$. Analogously as for the definitions of $F_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})$ and $F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}})$, let $F^{\text{reb}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})$ and $F^{\text{reb}}_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}})$ denote, respectively, the overall rebalancing flow entering (exiting) cut $(\mathcal{S}, \bar{\mathcal{S}})$. Summing Equation (5.5) over all nodes in S, one easily obtains

$$F^{\text{reb}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) - F^{\text{reb}}_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) = \sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}}\lambda_k - \sum_{k\in\mathcal{K}} 1_{t_k\in\mathcal{S}}\lambda_k.$$

Combining the above equation with Lemma 5.4.1, one obtains

$$F^{\text{reb}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) - F^{\text{reb}}_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) = F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) - F_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}),$$

in other words, rebalancing flows should make up the difference between the customer inflows and outflows across cut $(\mathcal{S}, \bar{\mathcal{S}})$. Accordingly, the total inflow of vehicles across $(\mathcal{S}, \bar{\mathcal{S}})$, $F^{\text{tot}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})$, satisfies the inequality

$$\begin{aligned} F^{\text{tot}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) :&= F_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) + F^{\text{reb}}_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) \\ &= F_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) + F^{\text{reb}}_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) + F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) - F_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}}) \\ &\geq F_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}). \end{aligned}$$

Since the customer and rebalancing flows $\{f_k(u,v),\ f_R(u,v)\}_{(u,v),k}$ are feasible, then, by Equation (5.6), $F^{\text{tot}}_{\text{in}}(S, \bar{S}) \leq C_{\text{in}}(S, \bar{S})$. Collecting the results, one obtains the second condition. □

We are now in a position to present a *structural* (i.e., flow-independent) necessary condition for the existence of feasible customer and rebalancing flows.

Theorem 5.4.4 (Necessary condition for feasible flows)**.** *A necessary condition for the existence of a set of* feasible *customer and rebalancing flows* $\{f_k(u,v),\ f_R(u,v)\}_{(u,v),k}$, *is that, for every cut* $(\mathcal{S}, \bar{\mathcal{S}})$,

1. $\sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}, t_k\in\bar{\mathcal{S}}}\lambda_k \leq C_{out}(\mathcal{S}, \bar{\mathcal{S}})$, *and*
2. $\sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}, t_k\in\bar{\mathcal{S}}}\lambda_k \leq C_{in}(\mathcal{S}, \bar{\mathcal{S}})$.

Proof. The theorem is a trivial consequence of Lemmas 5.4.2 and 5.4.3. □

Theorem 5.4.4 essentially provides a structural fundamental limitation of performance for a given road network: if the cut conditions in Theorem 5.4.4 are not met, then there is no hope of finding congestion-free customer and rebalancing flows. We reiterate that Theorem 5.4.4 holds for both symmetric and asymmetric networks (for a symmetric network, claim (2) in Lemma 5.4.3 and condition (2) in Theorem 5.4.4 are redundant).

5.4.2 Existence of Congestion-Free Flows

In this section we address the following question: assuming there exists a feasible customer flow, is it always possible to find a feasible rebalancing flow? As we will see, the answer to this question is affirmative and has both conceptual and algorithmic implications.

Theorem 5.4.5 (Existence of feasible rebalancing)**.** *Assume there exists a set of feasible customer flows* $\{f_k(u,v)\}_{(u,v),k}$. *Then, it is* always *possible to find a set of feasible rebalancing flows* $\{f_R(u,v)\}_{(u,v)}$.

Proof. We prove the theorem for the special case where no node $v \in \mathcal{V}$ is associated with both an origin and a destination for the transportation requests in $\mathcal{M}$. This is without loss of generality, as the general case where a node v has both an origin and a destination assigned can be reduced to this special case, by associating with node v a "shadow" node so that (1) all destinations are assigned to the shadow node and (2) node v and its shadow node are mutually connected via an infinite-capacity, zero-travel-time edge.

We start the proof by defining the concepts of *partial rebalancing flows* and *defective origins and destinations.* Specifically, a partial rebalancing flow, denoted as $\{\hat{f}_R(u,v)\}_{(u,v)}$, is a set of mappings from $\mathcal{E}$ to $\mathbb{R}_{\geq 0}$ obeying the following properties:

1. It satisfies Constraint (5.5) at every node that is not an origin nor a destination, that is for all $v \in \{\mathcal{V} \setminus \{\{s_k\}_k \cup \{t_k\}_k\}\}$,

$$\sum_{u\in\mathcal{V}} \hat{f}_R(u,v) = \sum_{w\in\mathcal{V}} \hat{f}_R(v,w).$$

2. It violates Constraint (5.5) in the "$\leq$ direction" at every node that is an origin, that is for all $v \in \mathcal{V}$ such that $\exists k \in \mathcal{K} : v = s_k$,

$$\sum_{u \in \mathcal{V}} \hat{f}_R(u, v) \leq \sum_{w \in \mathcal{V}} \hat{f}_R(v, w) + \sum_{k \in \mathcal{K}} 1_{v=s_k} \lambda_k.$$

3. It violates Constraint (5.5) in the "$\geq$ direction" at every node that is a destination, that is for all $v \in \mathcal{V}$ such that $\exists k \in \mathcal{K} : v = t_k$,

$$\sum_{u \in \mathcal{V}} \hat{f}_R(u, v) + \sum_{k \in \mathcal{K}} 1_{v=t_k} \lambda_k \geq \sum_{w \in \mathcal{V}} \hat{f}_R(v, w).$$

4. The combined customer and partial rebalancing flows $\{f_k(u, v), \hat{f}_R(u, v)\}_{(u,v),k}$ satisfy Equation (5.6) for every edge $(u, v) \in \mathcal{E}$.

Note that the trivial zero flow, that is $\hat{f}_R(u, v) = 0$ for all $(u, v) \in \mathcal{E}$, is a partial rebalancing flow (in other words, the set of partial rebalancing flows in not empty). Clearly a feasible rebalancing flow is also a partial rebalancing flow, but the opposite is not necessarily true.

For a given partial rebalancing flow, we denote an origin node, that is a node $v \in \mathcal{V}$ such that $v = s_k$ for some $k = 1, \ldots, K$, as a *defective* origin if Equation (5.5) is not satisfied at $v = s_k$ (in other words, the strict inequality $<$ holds). Analogously, we denote a destination node, that is a node $v \in \mathcal{V}$ such that $v = t_k$ for some $k = 1, \ldots, K$, as a *defective* destination if Equation (5.5) is not satisfied at $v = t_k$ (in other words, the strict inequality $>$ holds). The next lemma links the concepts of partial rebalancing flows and defective origins/destinations.

Lemma 5.4.6 (Co-existence of defective origins/destinations). *For every partial rebalancing flow that is not a feasible rebalancing flow, there exists at least one node $u \in \mathcal{V}$ that is a defective origin,* and *one node $v \in \mathcal{V}$ that is a defective destination.*

Proof. By contradiction. Since the flow $\{\hat{f}_R(u, v)\}_{(u,v)}$ is not a feasible rebalancing flow, there exists at least one defective origin or a defective destination. Assume that there exists at least one defective destination, say a node $\hat{t}_j$ where Equation (5.5) is violated:

$$\sum_{u \in \mathcal{V}} \hat{f}_R(u, \hat{t}_j) + \sum_{k \in \mathcal{K}} 1_{\hat{t}_j=t_k} \lambda_k > \sum_{w \in \mathcal{V}} \hat{f}_R(\hat{t}_j, w),$$

Now, assume that there does not exist any defective origin. By summing Equation (5.5) over all nodes $v \in \mathcal{V}$ and simplifying all flows $\hat{f}_R(u, v)$ (as they appear on both sides of the resulting equation), one obtains

$$\sum_{v \in \mathcal{V}} \sum_{k \in \mathcal{K}} 1_{v=t_k} \lambda_k > \sum_{v \in \mathcal{V}} \sum_{k \in \mathcal{K}} 1_{v=s_k} \lambda_k,$$

that is $\sum_{k\in\mathcal{K}} \lambda_k > \sum_{k\in\mathcal{K}} \lambda_k$, which is a contradiction. The alternative case where we assume that there exists at least one defective origin leads to an analogous contradiction. The lemma follows. □

For a given set of customer flows $\{f_k(u,v)\}_{(u,v),k}$ and partial rebalancing flows $\{\hat{f}_R(u,v)\}_{(u,v)}$, we call an edge $(u,v) \in \mathcal{E}$ *saturated* if Equation (5.6) holds with equality for that edge. We call a path *saturated* if at least one of the edges along the path is saturated. We now prove the existence of a special partial rebalancing flow where defective destinations and defective origins are separated by a graph cut formed exclusively by saturated edges (this result, and its consequences, are illustrated in Figure 5.4).

Lemma 5.4.7. *Assume there exists a set of* feasible *customer flows $\{f_k(u,v)\}_{(u,v),k}$, but there does not exist a set of feasible rebalancing flows $\{f_R(u,v)\}_{(u,v)}$. Then, there exists a partial rebalancing flow $\{\hat{f}_R(u,v)\}_{(u,v)}$ that induces a graph cut $(\mathcal{S},\bar{\mathcal{S}})$ with the following properties: (1) all defective destinations are in $\mathcal{S}$, (2) all defective origins are in $\bar{\mathcal{S}}$, and (3) all edges in $(S,\bar{\mathcal{S}})$ are saturated.*

Proof. The proof is constructive and constructs the desired partial rebalancing flow by starting with the trivial zero flow $\hat{f}_R(u,v) = 0$ for all $(u,v) \in \mathcal{E}$. Let $\mathcal{V}_{\text{or, def}} := \{\hat{s}_1, \dots, \hat{s}_{|\mathcal{V}_{\text{or, def}}|}\}$ and $\mathcal{V}_{\text{dest, def}} := \{\hat{t}_1, \dots, \hat{t}_{|\mathcal{V}_{\text{dest, def}}|}\}$ be the set of defective origins and destinations, respectively, under such flow. Then, the zero flow is iteratively updated according to the following procedure:

1. Look for a path between a node in $\mathcal{V}_{\text{dest, def}}$ and a node in $\mathcal{V}_{\text{or, def}}$ that is not saturated (note that for rebalancing flows, paths go from customer destinations to customer origins). If no such path exists, quit. Otherwise, go to Step 2.

2. Add the same amount of flow on all edges along the path until either (1) one of the edges becomes saturated or (2) Constraint (5.5) is fulfilled either at the defective origin or at the defective destination. Note that the resulting flow remains a partial rebalancing flow.

3. Update sets $\mathcal{V}_{\text{or, def}}$ and $\mathcal{V}_{\text{dest, def}}$ for the new partial rebalancing flow and go to Step 1.

The algorithm terminates. To show this, we prove the invariant that if a node is no longer defective for the updated partial rebalancing flow (in other words, Step 2 ends due to condition (2)), it will not become defective at a later stage. Consider a defective destination node v that becomes non-defective under the updated partial rebalancing flow (the proof for defective origins is analogous). Then, at the subsequent stage it cannot be considered as a destination in Step 1 (as it is no longer in set $\mathcal{V}_{\text{dest, def}}$). If a path that does not contain v is selected, then v stays non-defective. Otherwise, if a path that contains v is selected, then, after Step 2, both the inbound flow (that is the flow into v) and the outbound flow (that is the flow out of v) will be increased by the same quantity, and the node will stay non-defective. An induction on the stages then proves the claim. As the number of paths is finite, and sets $\mathcal{V}_{\text{or, def}}$ and $\mathcal{V}_{\text{dest, def}}$ cannot have any nodes added, the algorithm terminates after a finite number of stages.

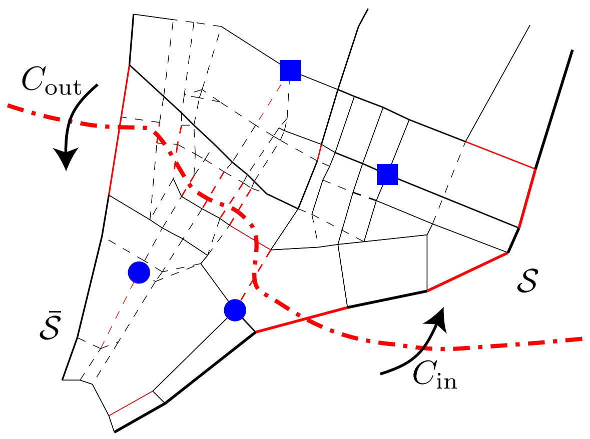


Figure 5.4: A graphical representation of Lemma 5.4.7. If there exists a set of feasible customer flows but there does not exist a set of feasible rebalancing flows, one can find a partial rebalancing flow where all the defective origins, represented as blue circles, are separated from all the defective destinations, represented as blue squares, by a cut of saturated edges (shown in red). In the proof of Theorem 5.4.5 we show that the capacity of such a cut $(\mathcal{S}, \bar{\mathcal{S}})$ is asymmetric, i.e., $C_{\text{out}} < C_{\text{in}}$ – a contradiction that leads to the claim of Theorem 5.4.5.

The output of the algorithm (denoted as $\{\hat{f}_R(u,v)\}_{(u,v)}$) is a partial rebalancing flow that is not feasible (as, by assumption, there does not exist a set of feasible rebalancing flows). Therefore, by Lemma 5.4.6, such partial rebalancing flow has at least one defective origin and at least one defective destination. Let us define $\mathcal{E}_{ns} := \mathcal{E} \setminus \{(u,v) : (u,v) \text{ is saturated}\}$ as the collection of non-saturated edges under the flows $\{f_k(u,v)\}_{(u,v),k}$ and $\{\hat{f}_R(u,v)\}_{(u,v)}$. For any defective destination and any defective origin, all paths connecting them contain at least one saturated edge (due to the exit condition in Step 1). Therefore, the graph $G_{ns}(\mathcal{V}, \mathcal{E}_{ns})$ has two properties: (1) it is disconnected (that is, it is not possible to find a direct path between every pair of nodes in $\mathcal{V}$ by using edges in $\mathcal{E}_{ns}$), and (2) a defective origin and a defective destination can not be in the same strongly connected component (hence, graph $G_{ns}(\mathcal{V}, \mathcal{E}_{ns})$ can be partitioned into at least two strongly connected components).

We now find the cut $(\mathcal{S}, \bar{\mathcal{S}})$ as follows. If a strongly connected component of G_{ns} contains defective destinations, we assign its nodes to set $\mathcal{S}$. If a strongly connected component contains defective origins, we assign its nodes to set $\bar{\mathcal{S}}$. If a strongly connected component contains neither defective origins nor destinations, we assign its nodes to $\mathcal{S}$ (one could also assign its nodes to $\bar{\mathcal{S}}$, but such choice is immaterial for our purposes). By construction, $(\mathcal{S}, \bar{\mathcal{S}})$ is a cut, and its edges are all saturated. Furthermore, set $\mathcal{S}$ only contains destination nodes, and set $\bar{\mathcal{S}}$ only contains origin nodes, which concludes the proof. □

We are now in a position to prove Theorem 5.4.5. The proof is by contradiction. Assume that a set of feasible rebalancing flows $\{f_R(u,v)\}_{(u,v)}$ does not exist. Then Lemma 5.4.7 shows that there exists a partial rebalancing flow $\{\hat{f}_R(u,v)\}_{(u,v)}$ and a cut $(\mathcal{S},\bar{\mathcal{S}})$ such that all defective destinations under $\{\hat{f}_R(u,v)\}_{(u,v)}$ belong to $\mathcal{S}$ and all defective origins belong to $\bar{\mathcal{S}}$. Let us denote the sum of all partial rebalancing flows across cut $(\mathcal{S},\bar{\mathcal{S}})$ as

$$\hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) := \sum_{u\in\mathcal{S}, v\in\bar{\mathcal{S}}} \hat{f}_R(u,v),$$

and, analogously, define $\hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) := \hat{F}^{\text{reb}}_{\text{out}}(\bar{\mathcal{S}},\mathcal{S})$. Since all edges in the cut $(\mathcal{S},\bar{\mathcal{S}})$ are saturated under $\{\hat{f}_R(u,v)\}_{(u,v)}$, one has, due to Equation (5.6), the equality

$$C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) = F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) + \hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}).$$

Additionally, again due to Equation (5.6), one has the inequality

$$F_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) + \hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) \leq C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}).$$

Combining the above equations, one obtains

$$F_{\text{in}}(S,\bar{S}) + \hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) - \hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) \leq C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}).$$

To compute $\hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - \hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}})$, we follow a procedure similar to the one used in Lemma 5.4.1. Summing Equation (5.5) over all nodes in $\mathcal{S}$, one obtains,

$$\sum_{v\in\mathcal{S}}\left[\sum_{u\in\mathcal{V}} \hat{f}_R(u,v) + \sum_{k\in\mathcal{K}} 1_{v=t_k}\lambda_k\right] > \sum_{v\in\mathcal{S}}\left[\sum_{w\in\mathcal{V}} \hat{f}_R(v,w) + \sum_{k\in\mathcal{K}} 1_{v=s_k}\lambda_k\right].$$

The strict inequality is due to the fact that for a partial rebalancing flow that is not feasible there exists at least one defective destination (Lemma 5.4.6), which, by construction, must belong to $\mathcal{S}$. Simplifying those flows $\hat{f}_R(u,v)$ for which both u and v are in $\mathcal{S}$ (as such flows appear on both sides of the above inequality), one obtains

$$\hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - \hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) > \sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}}\lambda_k - \sum_{k\in\mathcal{K}} 1_{t_k\in\mathcal{S}}\lambda_k.$$

Also, by Lemma 5.4.1,

$$F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) - F_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) = \sum_{k\in\mathcal{K}} 1_{s_k\in\mathcal{S}}\lambda_k - \sum_{k\in\mathcal{K}} 1_{t_k\in\mathcal{S}}\lambda_k.$$

Collecting all the results so far, we conclude that

$$\begin{aligned}0 &< F_{\text{in}}(S,\bar{S}) + \hat{F}^{\text{reb}}_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - F_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) - \hat{F}^{\text{reb}}_{\text{out}}(\mathcal{S},\bar{\mathcal{S}})\\ &< C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}).\end{aligned}$$

Hence, we reached the conclusion that $C_{\text{in}}(\mathcal{S},\bar{\mathcal{S}}) - C_{\text{out}}(\mathcal{S},\bar{\mathcal{S}}) > 0$, or, in other words, the capacity of graph $G(\mathcal{V},\mathcal{E})$ across cut $(\mathcal{S},\bar{\mathcal{S}})$ is *not* symmetric. This contradicts the assumption that graph $G(\mathcal{V},\mathcal{E})$ is capacity-symmetric, and the claim follows. □

The importance of Theorem 5.4.5 is twofold. First, perhaps surprisingly, it shows that for symmetric road networks it is *always* possible to rebalance the autonomous vehicles *without* increasing congestion – in other words, the rebalancing of autonomous vehicles in a symmetric road network does *not* lead to an increase in congestion. Second, from an algorithmic standpoint, if the cost function in the CRRP only depends on the customer flows (that is, $\rho = 0$ and the goal is to minimize the customers' travel times), then the CRRP problem can be *decoupled* and the customers and rebalancing flows can be solved separately without loss of optimality. This insight will be instrumental in Section 5.5 to the design of real-time algorithms for routing and rebalancing.

We conclude this section by noticing that the CRRP, from a computational standpoint, can be reduced to an instance of the Minimum-Cost Multi-Commodity Flow problem (Min-MCF), a classic problem in network flow theory [59]. The problem can be efficiently solved either via linear programming (the size of the linear program is $|\mathcal{E}|(M+1)$), or via specialized combinatorial algorithms [89, 90, 91]. However, the solution to the CRRP provides *static fractional* flows, which are not directly implementable for the operation of actual AMoD systems. Practical algorithms (inspired by the theoretical CRRP model) are presented in the next section.

5.5 Real-time Congestion-Aware Routing and Rebalancing

A natural approach to routing and rebalancing would be to periodically re-solve the CRRP within a receding-horizon, batch-processing scheme (a common scheme for the control of transportation networks [37, 21, 51]). This approach, however, is not directly implementable as the solution to the CRRP provides *fractional* flows (as opposed to routes for the *individual* vehicles). This shortcoming can be addressed by considering an integral version of the CRRP (dubbed integral CRRP), whereby the flows are *integer*-valued and can be thus easily translated into routes for the individual vehicles, e.g. through a flow decomposition algorithm [57] (see Algorithm 2). The integral CRRP, however, is an instance of the integral Minimum-Cost Multi-Commodity Flow problem, which is known to be NP-hard [58, 92]. Naïve rounding techniques are inapplicable: rounding a solution for the (non-integral) CRRP does not yield, in general, feasible integral flows, and hence feasible routes. For example, continuity of vehicles and customers can not be guaranteed, and vehicles may appear and disappear along a route. In general, to the best of our knowledge, there are no polynomial-time approximation schemes for the integral Minimum-Cost Multi-Commodity Flow problem.

On the positive side, the integral CRRP admits a decoupling result akin to Theorem 5.4.5: given a set of feasible, *integral* customer flows, one can always find a set of feasible, *integral* rebalancing flows. (In fact, the proof of Theorem 5.4.5 does not exploit anywhere the property that the flows are fractional, and thus the proof extends virtually unchanged to the case where the flows are integer-valued). Our approach is to leverage this insight (and more in general the theoretical results from Section 5.4) to design a heuristic, yet efficient approximation to the integral CRRP that (1) scales to large-scale systems, and (2) is general, in the sense that can be broadly applied to time-varying, asymmetric networks.

Specifically, we consider as objective the minimization of the customers' travel times, which, from Section 5.4 and the aforementioned discussion about the generalization of Theorem 5.4.5 to integral flows, *suggests* that customer routing can be decoupled from vehicle rebalancing (strictly speaking, this statement is only valid for static and symmetric networks – its generalization beyond these assumptions will be addressed numerically in Section 5.6). Accordingly, to emulate the real-world operation of an AMoD system, we divide a given city into geographic regions (or stations, as in Chapters 3 and 4), and each arriving customer is assigned the closest vehicle *within that region* (vehicle imbalance across regions is handled separately by the vehicle rebalancing algorithm, discussed below). We apply a greedy, yet computationally-efficient and congestion-aware approach for customer routing where customers are routed to their destinations using the shortest-time path as computed by an A^* algorithm [93]. The travel time along each edge is computed using a heuristic delay function that is related to the current volume of traffic on each edge. Similar to Section 5.2, for each edge $(u, v) \in \mathcal{E}$ we use the Bureau of Public Roads (BPR) delay model [85]

$$T^d(u,v) := T(u,v)\left(1+\alpha\left(\frac{f(u,v)}{c(u,v)}\right)^{\beta}\right),$$

where $f(u,v) := \sum_{k=1}^{K} f_k(u,v) + f_R(u,v)$ is the total flow on edge (u, v), and α and β are usually set to 0.15 and 4 respectively. Note that customer routing is *event-based*, i.e, a routing choice is made as soon as a customer arrives.

Separately from customer routing, vehicle rebalancing from one region to another is performed every $t_{\text{hor}} > 0$ time units as a batch process (unlike customer routing, which is an event-based process). Denote by $v_i(t)$ the number of vehicles in region i at time t, and by $v_{ji}(t)$ the number of vehicles traveling from region j to i that will arrive in the next t_{vicinity} time units. Let $v_i^{\text{own}}(t) := v_i(t) + \sum_j v_{ji}(t)$ be the number of vehicles currently "owned" by region i (i.e., in the vicinity of such region). Denote by $v_i^e(t)$ the number of excess vehicles in region i, or the number of vehicles left after servicing the customers waiting within region i. From its definition, $v_i^e(t)$ is given by $v_i^e(t) = v_i^{\text{own}}(t) - c_i(t)$, where $c_i(t)$ is the number of customers within region i. Finally, denote by

$v_i^d(t)$ the desired number of vehicles within region i. For example, for an even distribution of excess vehicles, $v_i^d(t) \propto \sum_i v_i^e(t)/N$, where N is the number regions. Note that the $v_i^d(t)$'s are rounded so they take on integer values. The set of origin regions (i.e., regions that should send out vehicles), S_R, and destination regions (i.e., regions that should receive vehicles), T_R, for the rebalancing vehicles are then determined by comparing $v_i^e(t)$ and $v_i^d(t)$, specifically,

$$\begin{aligned} &\text{if } v_i^e(t) > v_i^d(t), \quad \text{region } i \in S_R \\ &\text{if } v_i^e(t) < v_i^d(t), \quad \text{region } i \in T_R. \end{aligned}$$

We assume the residual capacity $c_R(u,v)$ of an edge (u,v), defined as the difference between its overall capacity $c(u,v)$ and the current number of vehicles along that edge, is known and remains approximately constant over the rebalancing time horizon. In case the overall rebalancing problem is not feasible (i.e. it is not possible to move all excess vehicles to regions that have a deficit of vehicles while satisfying the congestion constraints), we define slack variables with cost C that allow the optimizer to select a subset of vehicles and rebalancing routes of maximum cardinality such that each link does not become congested. The slack variables are denoted as ds_i for each $i \in S_R$, and dt_j for each $j \in T_R$.

Every t_{hor} time units, the rebalancing vehicle routes are computed by solving the following integer linear program

$$\begin{aligned} \underset{f_R(\cdot,\cdot),\{ds_i\},\{dt_j\}}{\text{minimize}} \quad & \sum_{(u,v)\in\mathcal{E}} T(u,v)\, f_R(u,v) \; + \sum_{i\in S_R} C ds_i + \sum_{i\in T_R} C dt_i \\ \text{subject to} \quad & \sum_{u\in\mathcal{V}} f_R(u,v) + 1_{v\in S_R}(v_v^e(t) - v_v^d(t) - ds_v) \\ & = \sum_{w\in\mathcal{V}} f_R(v,w) + 1_{v\in T_R}(v_v^d(t) - v_v^e(t) - dt_v), \quad \text{for all } v \in \mathcal{V} \\ & f_R(u,v) \le c_R(u,v), \quad \text{for all } (u,v) \in \mathcal{E} \\ & f_R(u,v) \in \mathbb{N}, \quad \text{for all } (u,v) \in \mathcal{E} \\ & ds_i, dt_j \in \mathbb{N}, \quad \text{for all } i \in S_R, j \in T_R \end{aligned}$$

The set of (integral) rebalancing flows $\{f_R(u,v)\}_{(u,v)}$ is then decomposed into a set of rebalancing paths via Ford-Fulkerson's flow decomposition algorithm (Algorithm 2). Each rebalancing path connects one origin region with one destination region: thus, rebalancing paths represent the set of routes that excess vehicles should follow to rebalance to regions with a deficit of vehicles.

The rebalancing optimization problem is an instance of the Minimum Cost Flow problem. If all edge capacities are integral, the linear relaxation of the Minimum Cost Flow problem enjoys a

Algorithm 2 Flow decomposition algorithm [57].

```
Input: A network flow ${f_R(u,v)}_{(u,v)}$
        A set $S_R$ of origin nodes $s \in S_R$
        A set $T_R$ of destination nodes $t \in T_R$
Output: A list of paths $\{Path_i\}_i$. Each entry $Path_i$ is a collection of consecutive edges $\{(s,u),(u,v),\dots(w,t)\}$.
        Number of vehicles to traverse each path $i$, $\{n_i^v\}_i$
procedure FlowDecomposition($\{f_R(u,v)\}_{(u,v)}$)
    $i = 1$
    $\{f_R^{\text{res}}(u,v)\}_{(u,v)} = \{f_R(u,v)\}_{(u,v)}$
    while $\sum_{s\in S_R}\sum_{v\in\mathcal{V}} f_R^{\text{res}}(s,v) > 0$ do
        $Path_i \leftarrow$ a path from some $s \in S_R$ to some $t \in T_R$ containing only edges $(u,v)$ with $f_R^{\text{res}}(u,v) > 0$
        $n_i^v = \min_{(u,v)\in Path_i} f_R^{\text{res}}(u,v)$
        for all $(u,v) \in Path_i$ do
            $f_R^{\text{res}}(u,v) = f_R^{\text{res}}(u,v) - n_i^v$
        $i = i + 1$
    return the path flows $\{\{Path_i\},\{n_i^v\}\}_i$
```

totally unimodular constraint matrix [59] (see Chapter 2.2.1). Hence, the linear relaxation will necessarily have an integer optimal solution, which will be a fortiori an optimal solution to the original Minimum Cost Flow problem. It follows that an integer-valued solution to the rebalancing optimization problem can be computed efficiently, namely in polynomial time, e.g., via linear programming. Several efficient combinatorial algorithms [59] are also available, whose computational performance is typically significantly better.

The favorable computational properties of the routing and rebalancing algorithm presented in this section enable application to large-scale systems, as described next.

5.6 Numerical Experiments

In this section, we characterize the effect of rebalancing on congestion in asymmetric networks and explore the performance of the algorithm presented in Section 5.5 on real-world road topologies with real customer demands.

5.6.1 Characterization of Congestion due to Rebalancing in Asymmetric Networks

The theoretical results in Section 5.4 are proven for capacity-symmetric networks, which are in general a reasonable model for typical urban road networks (see Appendix C for an analysis of capacity symmetry for major U.S. cities). Nevertheless, it is of interest to characterize the applicability of our theoretical results (chiefly, the existential result in Theorem 5.4.5) to road networks that significantly violate the capacity-symmetry property. In other words, we study to what degree rebalancing might lead to an increase in congestion if the network is asymmetric.

To this purpose, we compute solutions to the CRRP for road networks with varying degrees of capacity asymmetry and we compare corresponding travel times to those obtained by computing optimal routes in the absence of rebalancing (as it would be the case, e.g., if the vehicles were privately owned). We focus on the road network portrayed in Figure 5.5(a), which captures all major streets and avenues in Manhattan. Transportation requests are based on actual taxi rides in New York City on March 1, 2012 from 6 to 8 p.m.[3]. We randomly selected about one third of the trips that occurred in that time frame (roughly 17,000 trips) and we adjusted the capacities of the roads such that the flows induced by these trips would approach the threshold of congestion. The roads considered all have similar speed limits and comparable number of lanes and thus we assign to each edge in the network the same capacity, specifically, one vehicle every 23.6 seconds. This capacity is consistent with the observations that (1) the customer flow is only 30% of the real one (so road capacity is reduced accordingly) and (2) taxis only contribute to a fraction of the overall traffic in Manhattan. Nevertheless, we stress that the capacity was selected specifically to ensure that the flow induced by the trips would approach the threshold of congestion before any asymmetry is induced. To investigate the effects of network asymmetry, we introduce an *artificial capacity asymmetry* into the baseline Manhattan road network by progressively reducing the capacity of all northbound avenues.

In order to gain a *quantitative* understanding of the effect of rebalancing on congestion and travel times, we introduce slack variables $\delta_C(u, v)$, associated with a cost $c_c(u, v)$, to each congestion constraint (5.6). The cost $c_c(u, v)$ is selected so that the optimization algorithm will select a congestion-free solution whenever one is available. Once a solution is found, the actual travel time on each (possibly congested) link is computed with the heuristic BPR delay model [85] presented in Section 5.5. This approach maintains feasibility even in the congested traffic regime, and hence allows us to assess the impact of rebalancing on congestion in asymmetric networks.

Figure 5.5(b) summarizes the results of our simulations. In the baseline case, no artificial capacity asymmetry is introduced, i.e., the fractional capacity reduction of northbound avenues is equal to 0%. In this case, the customer routing problem with no rebalancing (essentially, the CRRP problem with the rebalancing flows constrained to be equal to zero) admits a congestion-free solution. On the other hand, the CRRP requires a (very small) relaxation of the congestion constraints. Overall, the difference between the travel times in the two cases is very small and approximately equal to 2.12%, in line with the fact that New York City's road graph has largely symmetric capacity, as discussed in Section 5.3 and shown in Appendix C. Interestingly, even with a massive 50% reduction in northbound capacity, travel times when rebalancing vehicles are present are within 4.12% of those obtained assuming no rebalancing is performed. Collectively, these results show that the

[3]courtesy of the New York Taxi and Limousine Commission

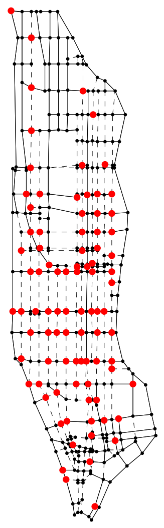
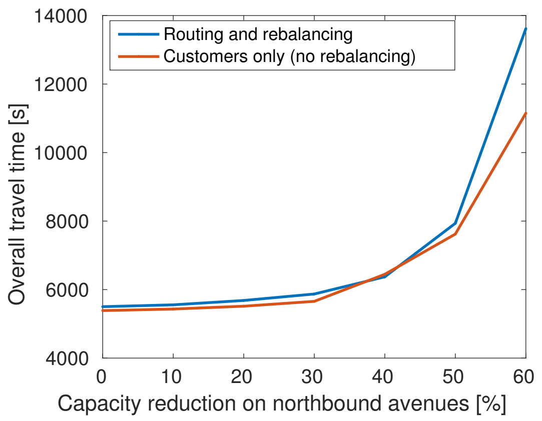



Figure 5.5: Left: Manhattan road network. One-way roads are represented as dashed lines. Centers of rebalancing regions are represented in red. Right: Customer travel times with and without rebalancing for different levels of network asymmetry.

existential result in Theorem 5.4.5, proven under the assumption of a symmetric network, appears to extend (even though approximately) to asymmetric networks. In particular, it appears that vehicle rebalancing does not lead to an appreciable increase in congestion under very general conditions.

We conclude this section by noticing that for a 40% reduction in capacity, the travel times with vehicle rebalancing dip slightly lower than those without. This effect is due to our use of the BPR link delay model: while in our theoretical model the time required to traverse a link is constant so long as a link is uncongested, the link delay in the BPR model varies by as much as 15% between free-flow and the onset of congestion.

5.6.2 Congestion-Aware Real-time Rebalancing

In this section we evaluate the performance of the real-time routing and rebalancing algorithm presented in Section 5.5 against a baseline approach (the real-time rebalancing algorithm in Chapter 3.3.3) that does not explicitly take congestion into account. We simulate $7,000$ vehicles providing service to actual taxi requests on March 1, 2012, for two hours between 6 and 8 p.m., using the same Manhattan road network as in the previous section (see Figure 5.5(a)). Taxi requests are clustered into 88 regions corresponding to a subset of nodes in the road network. Road capacities are reduced to account for exogenous vehicles on the roads to the point that congestion occurs along some routes during the simulation. The free flow speed of the vehicles is set to 25 mph (11 m/s) and approximately 55,000 trip requests (from the taxi data set discussed before) are simulated using a time step of 6 seconds. The simulated speed of the vehicles on each link depend on the number of vehicles in the link, and is calculated using the BPR model. Other delay factors such as traffic signals, turning times, and pedestrian blocking are not simulated.

Three simulations are performed, namely (1) assuming every customer has access to a private vehicle with no rebalancing, (2) using the congestion-aware routing and rebalancing algorithm presented in Section 5.5, and (3) using a baseline rebalancing algorithm, presented in Chapter 3.3.3. The baseline approach is a point-to-point algorithm that computes rebalancing origins and destinations without considering the underlying road network. In the baseline approach, customer routes are computed in the same way as in Section 5.5. For rebalancing, the origins and destinations are first solved using the algorithm presented in Chapter 3.3.3, then the routes are computed using the A^* algorithm much like the customer routes. In simulations (2) and (3), rebalancing is performed every 2 minutes.

Table 5.1 presents a summary of the performance results for simulations (2) and (3). Note that the service time is the total time a customer spends in the system (waiting + traveling). Only data

Table 5.1: Results of the real-time simulations

Performance metric	Congestion-aware	Baseline
# of trips completed	49,585	42,219
mean wait time (all trips)	163.57 s	406.03 s
mean travel time (completed trips)	265.13 s	275.19 s
mean service time (completed trips)	286.96 s	324 s
% with wait time > 5 minutes	5.4%	20%
mean # of rebalancing vehicles	204	1489

from simulations (2) and (3) are presented in Table 5.1 because the only applicable performance metric in simulation (1) is the mean travel time which was 264.69 s. Comparing our congestion-aware algorithm with (1), we notice that the additional rebalancing vehicles have no significant impact on the travel time. Comparing our algorithm with (3), we notice that the congestion-aware algorithm outperforms the baseline algorithm in every metric: low congestion allows the vehicles to service customers faster, resulting in a reduction in wait times as well as travel times. The baseline algorithm will send rebalancing vehicles to stations with a deficit of vehicles regardless of the level of congestion in the road network. This results in many more empty vehicles dispatched to rebalance the system (see Table 5.1), which causes heavy congestion in the network (see Figure 5.6). With our congestion-aware algorithm, we were able to drastically reduce this effect, resulting in very few congested road links.

However, it is important to note that limiting the number of rebalancing vehicles may create imbalance in the AMoD system and result in substantially longer wait times for some of the customers. A potential solution is to relax the congestion constraints as discussed in Section 5.6.1. This will result in a trade-off between creating more congestion by rebalancing and longer customer wait times. One way to perform the trade is to minimize the total service time of customers. However,

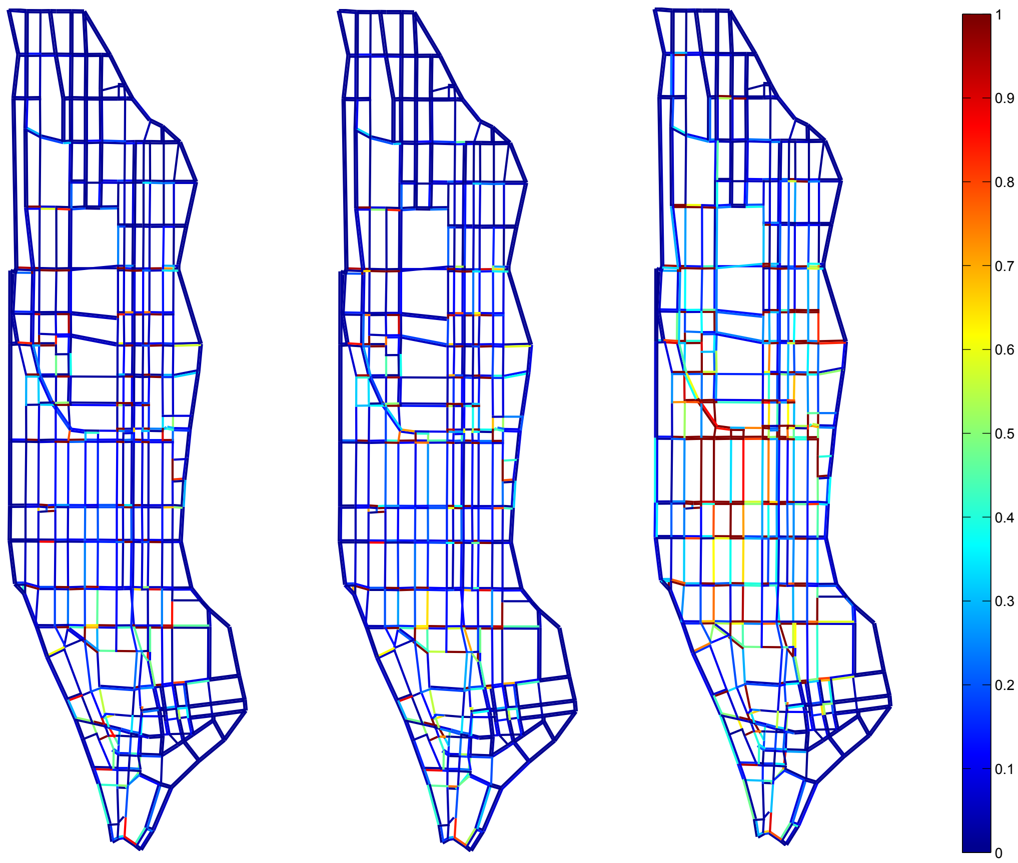


Figure 5.6: A snapshot of the simulation (at the final time step) showing traffic on the road network. Left: No rebalancing. Center: congestion-aware rebalancing. Right: baseline rebalancing. Blue indicates low congestion, red indicates high congestion. A dark red link signals that the vehicle flow is greater than or equal to the road capacity.

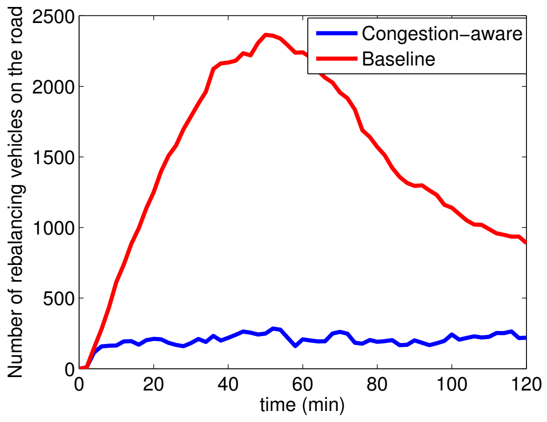


Figure 5.7: Number of rebalancing vehicles on the road in the 2 simulations.

if the customer can be notified of the amount of wait time beforehand, he/she may prefer waiting at the origin rather than spending a similar amount of time stuck in traffic.

5.7 Conclusion

In this chapter we presented a network flow model of an AMoD system on a capacitated road network. We formulated the routing and rebalancing problem and showed that on symmetric road networks, it is always possible to route rebalancing vehicles in a coordinated way that does not increase traffic congestion. Using a model road network of Manhattan, we showed that rebalancing did not increase congestion even for moderate degrees of network asymmetry. We leveraged the theoretical insights to develop a computationally efficient real-time congestion-aware routing and rebalancing algorithm and demonstrated its performance over its point-to-point counterpart through simulation. This highlighted the importance of congestion awareness in the design and implementation of control strategies for a fleet of self-driving vehicles.

The model does however exhibit several limitations. First, the flows in the model are assumed to be time-invariant, which is not representative of real congestion behavior (congestion tends to be a highly time-dependent phenomenon). This assumption is relaxed in the real-time congestion-aware rebalancing algorithm, but traffic behavior is still assumed to be constant in the time between rebalancing horizons. One way to improve upon this algorithm is to consider a time-expanded road network, where traffic predictions can be made by propagating the current traffic conditions. This is an interesting direction of future research.

Chapter 6

Model Predictive Control for AMoD With Charging Constraints

6.1 Introduction

In this chapter we study the problem of controlling AMoD systems in the presence of additional operational constraints, namely charging constraints associated with an electric vehicle fleet. We design a model predictive control (MPC) approach to optimize vehicle scheduling and routing in an AMoD system. Model predictive control (also known as receding horizon control) is a control technique whereby an open-loop optimization problem is solved at each time step to yield a sequence of control actions up to a fixed horizon, and the first control action is executed. Due to its iterative nature, MPC can achieve closed-loop performance, is robust to model errors, and is well suited for complex, constrained systems.

The key feature of this approach is that it is amenable to real-time optimization and receding horizon control while having the flexibility to account for many practical operational complexities such as battery charging constraints, customer priorities, and parking space limitations. Specifically, the contribution of this chapter is threefold. First, we propose a novel discrete-time model of an AMoD system and we show that this formulation allows the easy integration of a number of real-world constraints, with a special focus on electric vehicle charging constraints. Second, leveraging our model, we design a model predictive control algorithm for the optimal coordination of an AMoD system and prove its stability in the sense of Lyapunov. Finally, by using real-world data, we show that the MPC algorithm can be run in real-time for moderately-sized systems and compare its performance to four other AMoD control algorithms and taxi dispatch algorithms in the literature. We show that the MPC algorithm not only outperforms other algorithms in terms of customer wait times, but can also be used as an optimal performance benchmark to evaluate other dispatch algorithms.

This approach draws inspiration from time-space network models such as [94] for optimizing bus routes and [16, 29] for vehicle redistribution policies in car sharing systems. Receding horizon (or model predictive) control techniques have been used in the context of transportation systems [95, 96]. The key difference between our approach and these works (besides many differences in the model), is that we provide a rigorous proof of stability within an MPC framework. In this perspective, our approach is related to the one used in [97] for capacity maximization in battery networks and [98, 99] for cooperative multi-agent systems.

This chapter is organized as follows: In Section 6.2 we present our discrete-time AMoD model and discuss the inclusion of operational constraints, with a particular focus on battery charging. In Section 6.3 we formulate the problem of regulating an AMoD system. In Section 6.4 we present two MPC algorithms to control the AMoD system and prove their stability. Simulation studies are presented in Section 6.5 to assess the performance of the MPC algorithms and characterize the effect of charging constraints on system throughput. In Section 6.6 we conclude by discussing the inclusion of additional operational constraints, e.g., customers' priorities.

6.2 Discrete-time Model

In this section, we first introduce a linear discrete-time model of an AMoD system and then expand the model by introducing charging constraints associated with using electric vehicles. Such constraints are both of practical and theoretical interests. In particular, as we will see, with the addition of charging constraints (which are piecewise-linear), the system is no longer strictly a linear system. However, optimization in the form of a mixed-integer linear program (MILP) can still be performed with these additional constraints. The inclusion of additional operational constraints is discussed in Section 6.6.

6.2.1 Linear AMoD model

We consider a discrete-time system with N stations and m single-occupancy vehicles. Let $\mathcal{N}$ represent the set of stations, $|\mathcal{N}| = N$, and let V represent the set of vehicles, where $|V| = m$. At each time step, customers arrive at each station and wait for vehicles to transport them to their desired destinations. In this model, customers may not be serviced on a first-come-first-serve basis, as the system determines the best ordering for serving the customers. While this idea may at first seem to be at odds with the notion of "fairness," it is a standard strategy for ride-sharing services like SuperShuttle [100], where it is important to cluster customers traveling in the same direction. This strategy is also more natural in an equivalent system where "stations" are geographical regions (rather than physical infrastructure) and customers request transportation via a mobile app. Furthermore, we will show in Section 6.6 that customer priority can be easily integrated into the model.

Before defining the system states, we first define the "control" variables of the AMoD system. A control decision is made at each time step for each vehicle parked at a station. The two possible actions each vehicle can take are (1) transport a customer from one station to another, and (2) rebalance the system by driving itself from one station to another (this is a key advantage of autonomous vehicles). We can encode these actions using binary variables. Let $v_{ij}^k(t) = 1$ if vehicle k is transporting a customer from station i to station j beginning at time t, and arriving at station j at time $t + T_{ij}$. The travel time T_{ij} in this chapter is assumed to be deterministic and known. Similarly, let $w_{ij}^k(t) = 1$ if vehicle k is rebalancing from station i to station j beginning at time t and arriving at time $t + T_{ij}$.

Denote by $d_{ij}(t)$ the number of customers waiting at station i at the start of time period t whose destination is station j. Denote by $c_{ij}(t)$ the number of customers that arrive at station i at time t heading to station j. The dynamics of $d_{ij}(t)$ are propagated as follows:

$$d_{ij}(t+1) = d_{ij}(t) + c_{ij}(t) - \sum_{k \in V} v_{ij}^k(t), \tag{6.1}$$

where the last term represents the number of passenger-carrying vehicles leaving station i at time t. Note that $d_{ij} \geq 0$ for all $i, j \in \mathcal{N}$ and for all time. This means that if $d_{ij}(t) = 0$ and $c_{ij}(t) = 0$, then $v_{ij}^k(t) = 0$ for all $k \in V$. The number of waiting customers plays a significant role in characterizing the performance of the system, hence $d_{ij}(t)$ is modeled as a state variable.

When a vehicle is on the road, it is necessary to keep track of how long it will be traveling before it reaches its destination. We represent this by the binary variables ${}^{T_i}p_i^k(t) \in \{0, 1\}$, where $k \in V$ and $T_i \in \{0, \max_j\{T_{ji}\} - 1\}$ is the number of time steps remaining until the vehicle reaches station i, $i \in \mathcal{N}$. Suppose vehicle k leaves station j destined for station i at time t; then ${}^{T_{ji}-1}p_i^k(t+1)$ is set to one at the next time step to indicate that the vehicle is $T_{ji} - 1$ time steps from station i. In the subsequent time step, ${}^{T_{ji}-2}p_i^k(t+2)$ is set to one to indicate the progress of the vehicle along its path. Eventually ${}^{0}p_i^k(t + T_{ji})$ is set to one to signal that the vehicle has arrived at station i. The propagation of ${}^{T_i}p_i^k(t)$ is formally defined as follows:

$${}^{T_i}p_i^k(t+1) = \begin{cases} {}^{T_i+1}p_i^k(t) + \sum\limits_{j:T_{ji}-1=T_i} (v_{ji}^k(t) + w_{ji}^k(t)) & \text{if } T_i < T_{\max,i} \\ \sum\limits_{j:T_{ji}-1=T_{\max,i}} (v_{ji}^k(t) + w_{ji}^k(t)) & \text{if } T_i = T_{\max,i}, \end{cases} \tag{6.2}$$

where $T_{\max,i} = \max_j T_{ji} - 1$. Note that since each vehicle can only be in one place at one time, the

${}^{T_i}p_i^k$'s are subject to the constraints

$$\sum_{i\in\mathcal{N}}\sum_{T_i} {}^{T_i}p_i^k(t) \leq 1. \tag{6.3}$$

Constraint 6.3 will not be enforced explicitly, and instead will result from constraints on the control variables. The dimension of ${}^{T_i}p_i^k$ depends on the travel time from each station to every other station. For each vehicle and each station i there are $T_{\max,i}$ such variables, so the total dimension of ${}^{T_i}p_i^k$ is $\mathcal{D} = |V| \sum_{i\in\mathcal{N}} T_{\max,i}$.

Finally, let $u_i^k(t)$ be the state variable associated with waiting at a station, that is $u_i^k(t) = 1$ if vehicle k waited at station i from time $t-1$ to time t. The dynamics of $u_i^k(t)$ are modeled as follows

$$u_i^k(t+1) = u_i^k(t) +^0 p_i^k(t) - \sum_{j\in\mathcal{N}}(v_{ij}^k(t) + w_{ij}^k(t)). \tag{6.4}$$

Equation (6.4) ensures that (1) a vehicle can only perform an action (via v_{ij}^k or w_{ij}^k) if it is at a station, and (2) if a vehicle does not perform an action, it waits at a station. In other words, each vehicle must complete a task before starting another. However, a vehicle cannot perform an action if it is already on the road, thus a constraint is needed between $u_i^k(t)$ and ${}^{T_i}p_i^k(t)$ which ensures that a vehicle is either waiting at a station or traveling. This is formalized as

$$\sum_{i\in\mathcal{N}} u_i^k(t) + \sum_{i\in\mathcal{N},T_i} {}^{T_i}p_i^k(t) = 1. \tag{6.5}$$

Finally, the following constraint between u_i^k, v_{ij}^k, and w_{ij}^k ensures that a vehicle only performs one task at a time

$$\sum_{i\in\mathcal{N}}\Big(u_i^k(t+1) + \sum_{j\in\mathcal{N}} v_{ij}^k(t) + \sum_{j\in\mathcal{N}} w_{ij}^k(t)\Big) \leq 1, \tag{6.6}$$

where the sum is zero when vehicle k is traveling (i.e., $\sum_{i\in\mathcal{N},T_i\neq 0} {}^{T_i}p_i^k(t) = 1$).

The variables $d_{ij}(t)$, ${}^{T_i}p_i^k(t)$, and $u_i^k(t)$ make up the state of the system, as they completely define the customer demand and the state of all vehicles. Let $x(t)$ be the state vector, that is, the column vector created by reshaping and concatenating $d_{ij}(t)$, ${}^{T_i}p_i^k(t)$, and $u_i^k(t)$. We define the set of feasible states by $\mathcal{X}$ where

$$\mathcal{X} := \left\{ x = [d_{ij} \; {}^{T_i}p_i^k \; u_i^k]^\intercal \;\middle|\; \begin{array}{l} d_{ij} \in (\mathbb{N} \cup \{0\})^{N^2}, \; d_{ii} = 0 \\ {}^{T_i}p_i^k \in \{0,1\}^{\mathcal{D}}, \; {}^{T_i}p_i^k \text{ satisfies (6.3)} \\ u_i^k \in \{0,1\}^{N|V|}, \; u_i^k \text{ satisfies (6.5)} \end{array} \right\} \tag{6.7}$$

The variables $v_{ij}^k(t)$ and $w_{ij}^k(t)$ make up the control input of the system. Let $u(t)$ be the control vector, that is, the column vector created by concatenating $v_{ij}^k(t)$ and $w_{ij}^k(t)$. Any feasible control v_{ij}^k which sends vehicles to transport customers cannot transport more customers than there are waiting, thus

$$\sum_{k \in V} v_{ij}^k(t) \le d_{ij}(t) + c_{ij}(t). \tag{6.8}$$

We collect our system constraints to form the set of feasible controls, $\mathcal{U}(t)$, where

$$\mathcal{U}(t) := \left\{ u = [v_{ij}^k \; w_{ij}^k]^\intercal \;\middle|\; \begin{array}{l} v_{ij}^k \in \{0,1\}^{|V|N^2}, v_{ii}^k = 0 \\ w_{ij}^k \in \{0,1\}^{|V|N^2}, w_{ii}^k = 0 \\ u \text{ satisfies (6.6) and (6.8)} \end{array} \right\}. \tag{6.9}$$

Note that since (6.6) and (6.8) are time dependent, $\mathcal{U}(t)$ is time dependent. With this formulation, we have modeled an AMoD system (without battery charging or other operational constraints) as a linear system in the form of

$$x(t+1) = Ax(t) + Bu(t) + c(t), \tag{6.10}$$

where $x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}(t)$, $c(t) = [c_{ij}(t) \quad \mathbf{0} \quad \mathbf{0}]^\intercal$, and A and B are the coefficient matrices associated with (6.1), (6.2), and (6.4). The vector $c(t)$ represents new customers that arrive every time step and constitutes an exogenous disturbance for the system.

A few comments are in order. First, one may wonder why information about whether a vehicle is rebalancing or ferrying a passenger is not encoded in the state vector. This is because we have assumed that as soon as a customer boards a vehicle, he/she has been serviced and the vehicle is identical to one that is rebalancing (traveling without a customer). The only thing that matters is the time at which the vehicle arrives at its destination. Second, we have assumed that each station has sufficient parking space for as many vehicles as needed. This may be indeed true if the stations are geographical regions and vehicles are loitering within the region while waiting for customers. However, limited parking spaces are a real concern especially if parking spaces serve as charging stations for electric vehicles. In Section 6.6 we discuss how this AMoD framework can easily be extended to include limited parking capacity. In the next section we outline additional considerations associated with electric vehicles, in particular charging constraints (additional operational constraints are discussed in Section 6.6).

6.2.2 Charging constraints

For AMoD systems using electric vehicles, range is a major concern. To take into account the limited range of each vehicle, we define as an additional state variable the state of charge of each vehicle, $q^k(t) \in [0,1]$. A value $q^k(t) = 1$ means that the batteries are fully charged while $q^k(t) = 0$ means that the batteries are depleted. Vehicles' batteries discharge while driving, and can be charged at the stations while waiting for customers. The capacity of the batteries is limited, so once the batteries are full (i.e., $q^k(t) = 1$), charging stops. Each vehicle's charge evolves according to

$$q^k(t+1) = \min\{q^k(t) + \alpha_c \sum_{i \in \mathcal{N}} u_i^k(t+1), 1\} - \alpha_d \sum_{i \in \mathcal{N}, T_i} {}^{T_i}p_i^k(t+1), \tag{6.11}$$

where $\alpha_c > 0$ is the rate of charge at a charging station and $\alpha_d > 0$ is the rate of discharge while driving.

The charge of the vehicle restricts its range, and in some scenarios, a vehicle may have to wait at its charging station to charge rather than to transport a waiting customer. The charging constraints ensure that each vehicle has enough charge to complete its trip:

$$q^k(t) \geq v_{ij}^k(t)\, \alpha_d\, T_{ij}, \tag{6.12}$$

$$q^k(t) \geq w_{ij}^k(t)\, \alpha_d\, T_{ij}. \tag{6.13}$$

Constraint (6.12) ensures enough charge for a customer trip and (6.13) ensures enough charge for a rebalancing trip.

6.2.3 Objectives

The primary objective is to service all of the waiting customers as quickly as possible. A secondary goal is to ensure that rebalancing is done in an efficient manner and that vehicles do not rebalance when not necessary to avoid adding congestion on the road. Hence, for each time step t we have the cost functions

$$J_x(x(t)) = \sum_{i,j \in \mathcal{N}} d_{ij}(t), \text{ primary objective,} \tag{6.14}$$

$$J_u(u(t)) = \sum_{k \in V} \sum_{i,j \in \mathcal{N}} T_{ij} w_{ij}^k(t), \text{ secondary objective.} \tag{6.15}$$

When charging constraints are considered, we may include the vehicles' final state of charge as an objective to maximize. This allows us to trade off short-term quality-of-service and long-term battery capacity. To this end, we define the cost function

$$J_c(x(t_{\text{hor}})) = \sum_{k \in V} q^k(t_{\text{hor}}). \tag{6.16}$$

6.3 Problem Formulation

The objective of this chapter is to design model predictive control algorithms that are (1) provably stable in the sense of Lyapunov and (2) robust against the exogenous disturbance $c(t)$ (customer arrivals). We can now rigorously formulate the first problem, namely the AMoD regulation problem:

> **AMoD Regulation Problem (ARP)**: Assume $c_{ij}(t) = 0$ for all time $t > 0$. For each time t, select feasible control inputs $u(t) \in \mathcal{U}(t)$ such that as $t \to \infty$, $J_x(x(t)) \to 0$.

In the definition of the ARP it is assumed that the exogenous disturbance $c(t)$ is identically equal to zero for $t > 0$, in other words no new customers arrive after time zero. Hence, the ARP captures the *minimal* requirement that an initial set of customers is eventually transported to the respective destinations, under the assumption of zero future customer arrivals—hence the name "regulation problem."

While the ARP can also be solved by straightforward algorithms such as nearest neighbor dispatch (presented in Section 6.5), it is nevertheless critical to show that our MPC algorithm does not only offer good real-world performance but also guarantees analytical stability. The second objective, robustness against the exogenous disturbance $c(t)$, is analyzed in simulation in Section 6.5.

In the ARP, we note that $J_x(x(t)) \to 0$ will cause $J_u(u(t)) \to 0$ which implies that $u(t) \to \mathbf{0}$ by the definitions of $J_x(x(t))$ and $J_u(u(t))$.

Before presenting our MPC algorithms, we show some important structural properties of our problem setup. We first show that given $x(t) \in \mathcal{X}$, and $u(t) \in \mathcal{U}(t)$, the state of the undisturbed system at the next time step, $x(t+1) = Ax(t) + Bu(t)$, automatically satisfies (6.3) and (6.5), and thus $x(t+1) \in \mathcal{X}$. This property ensures the *persistent feasibility* of the MPC algorithms presented in Section 6.4.

Proposition 6.3.1 (Feasible Sets). *Given $x \in \mathcal{X}$, $u \in \mathcal{U}(t)$, and x^+ given by $x^+ = Ax + Bu$, then $x^+ \in \mathcal{X}$.*

Proof. Let $x^+ = [d_{ij}^+ \; {}^{T_i}p_i^{k+} \; u_i^{k+}]^\intercal$ and $u = [v_{ij}^k \; w_{ij}^k]^\intercal$. First, we note that if v_{ij}^k satisfies (6.8), $d_{ij}^+ \in \{\mathbb{N} \cup 0\}^{N^2}$ and is feasible. Next we show that ${}^{T_i}p_i^{k+} \in \{0,1\}^{\mathcal{D}}$ and satisfies (6.3). To show that ${}^{T_i}p_i^{k+}$ can only take on 0 or 1, first consider $T_i = T_{\max,i}$. In this case, by (6.6), $\sum_{j:T_{ji}-1=T_{\max,i}}(v_{ji}^k + w_{ji}^k) \leq 1$, so ${}^{T_{\max,i}}p_i^{k+} \leq 1$. Now consider $T_i < T_{\max,i}$. Here, we only need to consider the case when ${}^{T_i+1}p_i^k = 1$. When this is the case, by (6.5), $u_i^k = 0$ for all $i \in \mathcal{N}$. Also, by (6.3), ${}^0p_i^k = 0$ since

$T_i + 1 > 0$ ($T_i \geq 0$). Putting these facts into (6.4), we see that $\sum_{j\in\mathcal{N}}(v_{ij}^k + w_{ij}^k) = 0$ which proves that ${}^{T_i}p_i^{k+} \in \{0,1\}^{\mathcal{D}}$. To show Constraint (6.3) is satisfied, take the sum of (6.2):

$$\sum_i \sum_{T_i} {}^{T_i}p_i^{k+} = \sum_i \sum_{T_i=1}^{T_{\max,i}} {}^{T_i}p_i^k + \sum_{i,j}(v_{ji}^k + w_{ji}^k). \tag{6.17}$$

By (6.3) and (6.6), both terms on the right hand side are less than or equal to one. First consider when the second term is equal to one, from (6.4), either $u_i^k = 1$ or ${}^0p_i^k = 1$, and in both cases, $\sum_i \sum_{T_i=1}^{T_{\max,i}} {}^{T_i}p_i^k = 0$. Now consider when the first term on the right hand side of (6.17) is equal to one. In this case, according to (6.5), both u_i^k and ${}^0p_i^k$ must be equal to zero, so by (6.4), $\sum_{i,j}(v_{ji}^k + w_{ji}^k) = 0$. Hence ${}^{T_i}p_i^{k+}$ satisfies (6.3).

To show that u_i^{k+} satisfies (6.5), take the sum of (6.2) and (6.4)

$$\begin{aligned}\sum_i u_i^{k+} + \sum_i \sum_{T_i} {}^{T_i}p_i^{k+} &= \sum_i u_i^k + \sum_i {}^0p_i^k - \sum_{i,j}(v_{ij}^k + w_{ij}^k) + \sum_i \sum_{T_i=1}^{T_{\max,i}} {}^{T_i}p_i^k + \sum_{i,j}(v_{ji}^k + w_{ji}^k) \\ &= \sum_i u_i^k + \sum_i \sum_{T_i} {}^{T_i}p_i^k = 1.\end{aligned}$$

Hence, (6.5) is satisfied and the proposition is proven. □

With this result we can be sure that the reachable space of the system is feasible.

6.4 Model Predictive Control of AMoD

In this section we present two MPC algorithms to optimize vehicle scheduling and routing in an AMoD system. Specifically, the first MPC algorithm addresses the case without charging constraints (Section 6.4.1), while the second MPC algorithm allows the inclusion of charging constraints (Section 6.4.2). We prove that both algorithms solve the ARP (our technical approach is to prove asymptotic stability in the sense of Lyapunov). Note that in general asymptotic Lyapunov stability is a stronger result than simply proving $J_x(x(t)) \to 0$. However, due to the boundedness of the number of customers ($d_{ij}(t)$), asymptotic Lyapunov stability coincides with solving the ARP in this case. We remark that proving asymptotic stability in the sense of Lyapunov does not only guarantee that the number of passengers will decrease to zero (hence, wait times will not grow unbounded) but also implies that, if initial conditions are small (i.e. few passengers are requesting service), wait times will also be small. We numerically characterize the performance of the MPC algorithms in Section 6.5 (in particular, we study their ability to deal with a continuous stream of arriving customers).

6.4.1 MPC without charging constraints

In this section we present the MPC algorithm for solving the AMoD regulation problem without charging constraints. In an MPC algorithm, an optimization problem is solved at each time instant giving a sequence of control actions up to a time horizon t_{hor}. The first step of the control sequence is implemented and the system is re-optimized at the next time instant. Let $u(t+k)_{|t}$ be the control action at time $t+k$, solved at time t, where $k \in \{0, t_{\text{hor}}-1\}$.

Algorithm 1 (MPC without charging constraints)**.** *Given $x(t) \in \mathcal{X}$, at each time instant $t \in \mathbb{N}$ the controls $u(t)_{|t}, u(t+1)_{|t}, \ldots, u(t+t_{hor}-1)_{|t}$ are obtained by solving the optimization problem*

$$\begin{aligned} \underset{u(t),\ldots,u(t+t_{hor}-1)}{\text{minimize}} \quad & \sum_{\tau=t}^{t+t_{hor}-1} J_x(x(\tau+1)) + \rho_1 J_u(u(\tau)) \\ \text{subject to} \quad & x(\tau+1) = Ax(\tau) + Bu(\tau) \\ & x(\tau+1) \in \mathcal{X} \\ & u(\tau) \in \mathcal{U}(\tau) \\ & \tau = t, \ldots, t+t_{hor}-1. \end{aligned}$$

where $\rho_1 > 0$ and $J_x(x(\tau))$ and $J_u(u(\tau))$ are given by (6.14) *and* (6.15)*, respectively. Implement $u(t)_{|t}$ and repeat the optimization at the next time instant.*

Remark 6.4.1. *The purpose of $\rho_1 J_u(u(\tau))$ in the objective is to avoid unnecessary vehicle rebalancing. However, since not enough rebalancing may result in customers not receiving service, this term is secondary to the primary objective of servicing customers so ρ_1 should be set to a small value.*

Before presenting the main result of this section, namely the asymptotic stability of Algorithm 1, we define the notions of N-step and ∞-step reachable sets for the undisturbed system (6.10). These definitions are used to prove Lyapunov stability of the MPC algorithms presented in Theorems 6.4.5 and 6.4.6.

Definition 6.4.2 (N-step Reachable Set)**.** *Given an initial condition $x(0) \in \mathcal{X}$, the N-step reachable set is defined recursively as*

$$\mathcal{R}_{i+1} := \left\{x^+ \in \mathcal{X} \mid \exists x \in \mathcal{R}_i, u \in \mathcal{U}(i) \text{ such that } x^+ = Ax + Bu\right\}, \qquad (6.18)$$

for $i = 0 \ldots N-1$ and $\mathcal{R}_0 = x(0)$.

Definition 6.4.3 (∞-step Reachable Set)**.** *Given $x(0) \in \mathcal{X}$, the ∞-step reachable set of system*

(6.10) *subject to* (6.9) *is*

$$\mathcal{R}_\infty := \limsup_{N\to\infty} \mathcal{R}_N, \tag{6.19}$$

where the above limit is in a set-theoretical sense (i.e., $\limsup_{N\to\infty} \mathcal{R}_N = \cap_{N\geq 1} \cup_{m\geq N} \mathcal{R}_m$*).*

The results in this section also relies on a key theorem of set-valued Lyapunov functions, found in [99, Theorem 4]. We restate the theorem for completeness (its proof can be found in [99]).

Theorem 6.4.4 (Lyapunov stability for set-valued functions)**.** *Let* $\mathcal{R}$ *be a finite dimensional Euclidean space and consider a continuous map* $f : \mathbb{N} \times \mathcal{R} \to \mathcal{R}$ *giving rise to the discrete-time system*

$$x(t+1) = f(t, x(t)). \tag{6.20}$$

Let Ξ *be the collection of equilibrium solutions of* (6.20) *and* $\mathcal{X}^e$ *be the set of equilibrium points corresponding to* Ξ*. Let* $W : \mathcal{R} \rightrightarrows \mathcal{R}$ *be an upper semi-continuous set-valued Lyapunov function satisfying*

1. $x \in W(x)$ *for all* $x \in \mathcal{R}$,
2. $W(x^e) = \{x^e\}$ *for all* $x^e \in \mathcal{X}^e$,
3. $W(x(t+1)) \subseteq W(x(t))$ *for all* $x(t) \in \mathcal{R}$.

Then, system (6.20) *is uniformly stable with respect to* Ξ *in the sense of Lyapunov. If additionally,*

4. *there exists a function* $\mu : Im(W) \to \mathbb{R}_{\geq 0}$*, bounded on bounded sets, such that*

$$\mu\left(W(x(t+1))\right) < \mu\left(W(x(t))\right) \tag{6.21}$$

for all $x(t) \in \mathcal{R} \setminus \mathcal{X}^e$*, then* $x(t) \to x^e \in \mathcal{X}^e$ *as* $t \to \infty$ *and the system is asymptotically stable with respect to* Ξ*.*

We are now ready to present the main result of this section, which shows that Algorithm 1 solves the ARP problem.

Theorem 6.4.5 (Asymptotic stability of Algorithm 1)**.** *Suppose* $t_{hor} \geq 2\max_{i,j\in\mathcal{N}} T_{ij}$*. Then Algorithm 1 solves the AMoD regulation problem.*

Proof. The key idea of the proof is to show that at least one customer is serviced every t_{hor} time steps. We can do this by defining a new linear system equivalent to (6.10) where t_{hor} time steps in (6.10) correspond to one time step in the new system. We can then use Theorem 6.4.4 to prove the asymptotic stability of the system.

To show that Algorithm 1 solves the ARP, we need only to look at the subspace $\mathcal{Y}$ of $\mathcal{X}$, where

$$\mathcal{Y} = \left\{ \tilde{x} = d_{ij} \mid d_{ij} \in (\mathbb{N} \cup \{0\})^{N^2}, \quad d_{ii} = 0 \right\}. \tag{6.22}$$

This is because the objective $J_x(x(t))$ considered in the ARP is only a function of d_{ij}'s. The dynamics of $\tilde{x}$ (referred to as the reduced state) follow (6.1), and the corresponding reduced control becomes $\tilde{u}(t) = v_{ij}^k(t)$ (note that all other constraints in Section 6.2.1 must still be satisfied). Equation (6.1) can then be written as

$$\tilde{x}(t+1) = \tilde{x}(t) + \tilde{B}\tilde{u}(t), \tag{6.23}$$

where $\tilde{B}$ is the matrix realization of the summation in (6.1). We can define the reduced N-step reachable set $\mathcal{Y}_N$ and the ∞-step reachable set $\mathcal{Y}_\infty$ as the appropriate subspaces of $\mathcal{R}_N$ and $\mathcal{R}_\infty$, respectively. For t_{hor} transitions, we can write

$$\begin{aligned} \tilde{x}(t+t_{\text{hor}}) &= \tilde{x}(t+t_{\text{hor}}-1) + \tilde{B}\tilde{u}(t+t_{\text{hor}}-1) \\ &= \tilde{x}(t) + \tilde{B}(\tilde{u}(t) + \ldots + \tilde{u}(t+t_{\text{hor}}-1)) \end{aligned} \tag{6.24}$$

We can further rescale the time variable so that one new time step (denoted by T) is equivalent to t_{hor} old time steps. With this, we can rewrite (6.24) as

$$\tilde{x}(T+1) = \tilde{x}(T) + \tilde{B}\tilde{U}(T) \tag{6.25}$$

where $\tilde{U}(T) = \tilde{u}(t) + \ldots + \tilde{u}(t+t_{\text{hor}}-1)$.

Now, using our reduced system written in the form of (6.25), consider the following set-valued Lyapunov function candidate

$$W(\tilde{x}) := \left\{ \tilde{x} + \tilde{B}\tilde{U} \in \mathcal{Y}_\infty \mid J_x(\tilde{x} + \tilde{B}\tilde{U}) \leq J_x(\tilde{x}) \right\}. \tag{6.26}$$

First, note that the reduced system (6.25) is persistently feasible, since $\tilde{u}(t) = v_{ij}^k(t) = 0$ is always a feasible control input (this is the case where no customers are serviced). Next, we show that $W(\tilde{x})$ is upper semi-continuous. It is necessary and sufficient that the graph of $W(\tilde{x})$ be a closed set [101, p.42]. The graph of W, namely

$$\operatorname{graph}(W) := \left\{ (\tilde{x}, \tilde{x} + \tilde{B}\tilde{U}) \mid \tilde{x} \in \mathcal{Y}_\infty, J_x(\tilde{x} + \tilde{B}\tilde{U}) \leq J_x(\tilde{x}) \right\}, \tag{6.27}$$

is closed because the state space is finite, hence $W(\tilde{x})$ is upper semi-continuous.

The equilibrium point we wish to converge to corresponds to $\arg\min J_x(x(t))$ which is $\tilde{x} = \mathbf{0}$ (the state where all customers have been served). Hence, $\mathcal{X}^e = \{\mathbf{0}\}$. To satisfy the first condition of Theorem 6.4.4, set $v_{ij}^k(\tau) = 0$ for $\tau = t, \ldots, t + t_{\text{hor}} - 1$. This is the same as setting $\tilde{U} = \mathbf{0}$ and hence $\tilde{x} \in W(\tilde{x})$. For the second condition, since $x^e = \mathbf{0}$ and $J_x(x^e) = 0$, we have $W(x^e) = \{x^e\}$. Note that because of Constraint (6.8), $\tilde{U} = \mathbf{0}$ when $\tilde{x} = \mathbf{0}$.

To show that $W(\tilde{x})$ satisfies the third condition, let $z \in W(\tilde{x}(T+1))$. By definition, there exists a sequence of feasible inputs $\tilde{V}$ such that $z = \tilde{x}(T+1) + \tilde{B}\tilde{V}$ and $J_x(z) \le J_x(\tilde{x}(T+1))$. By (6.25), there exists $\tilde{U}$ such that $\tilde{x}(T+1) = \tilde{x}(T) + \tilde{B}\tilde{U}$. Hence, there is a feasible sequence of inputs $\tilde{U} + \tilde{V}$ such that $z = \tilde{x}(T) + \tilde{B}(\tilde{U} + \tilde{V})$, and since $J_x(z) \le J_x(\tilde{x}(T+1)) \le J_x(\tilde{x}(T))$, we have $W(\tilde{x}(T+1)) \subseteq W(\tilde{x}(T))$.

Finally, let

$$\mu(W(\tilde{x}(T))) = \min_{z \in W(\tilde{x}(T))} J_x(z). \tag{6.28}$$

Clearly, $\mu(W(\tilde{x}))$ is bounded. To show that $\mu(W(\tilde{x}(T+1))) < \mu(W(\tilde{x}(T)))$ is equivalent to showing that the number of waiting customers decreases from T to $T+1$ under a sequence of feasible control actions given by the solution of Algorithm 1. According to (6.1), J_x is minimized when $\sum_{i,j \in \mathcal{N}, k \in V} v_{ij}^k$ is maximized, and $\mu(W(\tilde{x}))$ will decrease as long as at least one value in $v_{ij}^k(t), \ldots, v_{ij}^k(t + t_{\text{hor}} - 1)$ is nonzero. Thus, (6.21) will be satisfied if we find an upper bound on t_{hor} that will guarantee at least one of $v_{ij}^k(t), \ldots, v_{ij}^k(t + t_{\text{hor}} - 1)$ will be nonzero. The variables $v_{ij}^k(t)$ and $w_{ij}^k(t)$ are governed by (6.4), which states that either $v_{ij}^k(t)$ or $w_{ij}^k(t)$ can only be nonzero if vehicle k has been waiting or has just arrived at a station. The time it takes for a vehicle to arrive at a station is upper bounded by $\max_{i,j} T_{ij}$. However, the station that the vehicle arrives at may not have customers waiting. In this case, w_{ij}^k is needed to send the vehicle to a station with customers, which takes an amount of time upper bounded by $\max_{i,j} T_{ij}$. Thus, if $t_{\text{hor}} \ge 2\max_{i,j} T_{ij}$, (6.21) is satisfied for all $\tilde{x}(t) \ne \mathbf{0}$. Hence, by Theorem 6.4.4, the MPC algorithm is asymptotically stable and $\tilde{x}(t) \to \mathbf{0}$ as $t \to \infty$, which completes the proof. □

6.4.2 MPC with charging constraints

In this section we extend the results in the previous section to account for range limitations and charging constraints associated with electric vehicles. To do this, we first augment the state vector $x(t)$ with the charge of each vehicle, $q^k(t)$. The new state vector becomes $x' = [d_{ij}\ {}^{T_i}p_i^k\ u_i^k\ q^k]^\intercal$. The state $q^k(t)$ is propagated at each time step according to (6.11). However, (6.11) is piecewise linear (due to the min operator) so the extended system cannot be written in the form of (6.10). As we will see, this will not be an issue for the MPC algorithm. Finally, we add the range constraints (6.12) and (6.13) to the definition of $\mathcal{U}(t)$. To summarize, the augmented feasible states and controls are

$$\mathcal{X}' = \left\{ x' = [x\ q^k]^\intercal \;\middle|\; \begin{array}{l} x \in \mathcal{X} \\ q^k \in \mathbb{R}^V,\ 0 \le q^k \le 1 \end{array} \right\}, \tag{6.29}$$

$$\mathcal{U}'(t) = \left\{ u' = [v_{ij}^k \; w_{ij}^k]^\intercal \;\middle|\; \begin{array}{l} v_{ij}^k \in \{0,1\}^{|V|N^2}, \quad v_{ii}^k = 0 \\ w_{ij}^k \in \{0,1\}^{|V|N^2}, \quad w_{ii}^k = 0 \\ u' \text{ satisfies 6.6, 6.8, 6.12, and 6.13} \end{array} \right\}. \tag{6.30}$$

We turn our attention back to $q^k(t)$ and notice that from (6.11), $q^k(t+1)$ satisfies the following two linear inequalities

$$q^k(t+1) \leq q^k(t) + \alpha_c \sum_{i\in\mathcal{N}} u_i^k(t+1) - \alpha_d \sum_{i\in\mathcal{N},T_i} {}^{T_i}p_i^k(t+1) \tag{6.31}$$

$$q^k(t+1) \leq 1 - \alpha_d \sum_{i\in\mathcal{N},T_i} {}^{T_i}p_i^k(t+1). \tag{6.32}$$

Equation (6.11) can be satisfied in our MPC algorithm by satisfying (6.31) and (6.32) and maximizing q^k.

Algorithm 2 (MPC with charging constraints)**.** *At each time instant $t \in \mathbb{N}$ the controls $u'(t)_{|t}, u'(t+1)_{|t}, ...u'(t+t_{hor}-1)_{|t}$ are obtained by solving the optimization problem*

$$\begin{aligned} \underset{u'(t),\ldots,u'(t+t_{hor}-1)}{\text{minimize}} \quad & \sum_{\tau=t}^{t+t_{hor}-1} \left(J_x(x(\tau+1)) + \rho_1 J_u(u'(\tau)) - \rho_2 \sum_{k\in V} q^k(\tau+1) \right) - \rho_c J_c(x(t_{hor})) \\ \text{subject to} \quad & x(\tau+1) = Ax(\tau) + Bu(\tau) \\ & q^k(\tau+1) \leq q^k(\tau) + \alpha_c \sum_{i\in\mathcal{N}} u_i^k(\tau+1) - \alpha_d \sum_{i\in\mathcal{N},T_i} {}^{T_i}p_i^k(\tau+1) \\ & q^k(\tau+1) \leq 1 - \alpha_d \sum_{i\in\mathcal{N},T_i} {}^{T_i}p_i^k(\tau+1) \\ & x'(\tau+1) \in \mathcal{X}' \\ & u'(\tau) \in \mathcal{U}'(\tau) \\ & \tau = t, \ldots, t+t_{hor}-1 \end{aligned}$$

where $\rho_1 > 0$, $\rho_2 > 0$, and $\rho_c > 0$. Implement $u'(t)_{|t}$ and repeat the optimization at the next time instant.

The next theorem shows that Algorithm 2 solves the ARP with charging constraints.

Theorem 6.4.6 (Asymptotic stability of Algorithm 2)**.** *Suppose $t_{hor} \geq 2(1+\frac{\alpha_d}{\alpha_c})\max_{i,j\in\mathcal{N}} T_{ij}$. Then Algorithm 2 solves the AMoD regulation problem with charging constraints.*

Proof of Theorem 6.4.6. The proof for this theorem follows the same procedure as the proof of Theorem 6.4.5. The main difference is the choice of the time horizon. Consider again the case where vehicle k is enroute from station j to station i. Suppose there are no customers at station i, so once the vehicle arrives, it must travel back to station j for the next pickup. However, once the vehicle arrives at station i, it is completely depleted of charge and needs to charge for $\frac{\alpha_d}{\alpha_c} T_{ij}$ amount of time before departing. Once the vehicle arrives back at station j, it is again depleted of charge and must charge again before servicing the customer. The total time this takes consists of two traveling periods and two charging periods, with the travel time upper bounded by $\max_{i,j \in \mathcal{N}} T_{ij}$ and charging time upper bounded by $\frac{\alpha_d}{\alpha_c} \max_{i,j \in \mathcal{N}} T_{ij}$. We can further observe that fast charging reduces the time horizon needed to maintain stability. This completes the proof. □

Remark 6.4.7. *The time horizon bounds given in Theorem 6.4.5 and 6.4.6 assume the worst case scenario, which would very rarely occur in practice. Thus, in most practical cases, a shorter time horizon (which reduces computational cost) should also reduce d_{ij} to zero. In Section 6.5.1 we will show this is indeed the case.*

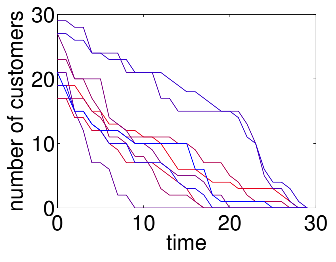
In the next section we show through simulation that the MPC algorithms solve the AMoD regulation problem and we benchmark their performance against other algorithms in the literature.

6.5 Simulation Results

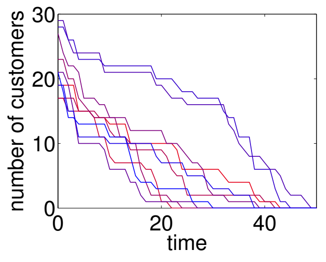
In this section, we present three sets of simulation results that demonstrate the correctness and performance of our MPC approach. First, we show through simulation that the AMoD regulation problem can indeed be solved using Algorithm 1 and Algorithm 2. Second, we show using real taxi data that Algorithm 1 yields real-time performance on small to medium-sized systems and outperforms several state-of-the-art algorithms from the literature in terms of customer wait times. Finally, we study how the charge/discharge rate of batteries affect the performance of an AMoD system with electric vehicles. For all simulations, Algorithm 1 and 2 were implemented using the IBM CPLEX solver for mixed-integer linear programs (MILP) [71].

6.5.1 AMoD regulation

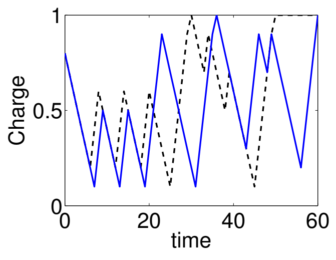
To validate Algorithms 1 and 2, an initial d_{ij} was randomly generated with up to 30 customers at each station while $c_{ij}(t)$ was set to zero for all t. The simulation was performed with 30 vehicles and 10 stations, with 3 vehicles at each station to begin with. The maximum travel time between two stations was 7 time steps. For the system without charging constraints, t_{hor} was set to 10 steps while for the system with charging constraints, t_{hor} was set to 20. The weight of the secondary objectives were set to $\rho_1 = 0.01$ and, for the system with charging constraints, $\rho_2 = 0.001$, $\rho_c = 0$. Figure 6.1(a) shows the number of customers waiting at each of the 10 stations as a function of time. Figure 6.1(b) shows the number of customers vs. time for a system with charging constraints. In this case, the initial charge of all vehicles was set to 0.8 and vehicles could charge twice as fast as they could discharge ($\alpha_c = 0.2$ and $\alpha_d = 0.1$). Figure 6.1(c) shows the charge levels of two of the vehicles and illustrates that when future customer demand is not taken into account, a general strategy for each vehicle is to service customers until its batteries are almost depleted, then charge just enough to service the next customer. The need to recharge after trips results in longer wait times and in this case, a longer total time to service all the customers (50 minutes for the case with charging constraints and 30 minutes without).



(a)



(b)



(c)

Figure 6.1: 6.1(a): Number of customers waiting at each station as a function of time for 10 stations, 30 vehicles without charging constraints. 6.1(b): Number of customers waiting at each station as a function of time with charging constraints. 6.1(c): State of charge for two vehicles as a function of time.

6.5.2 Performance of MPC

To evaluate the performance of our MPC algorithm, we conducted an extensive simulation study comparing Algorithm 1 to several other AMoD and taxi dispatch algorithms found in the literature using real New York taxi data[1]. We show that Algorithm 1 not only outperforms other algorithms in terms of customer wait times, but can be used as an “optimal” baseline to quantitatively evaluate the performance of other taxi dispatch or AMoD algorithms. The simulations are performed with 40 vehicles for 24 hours with a time step of 6 seconds. Taxi trips within New York City’s Financial District area (Lower Manhattan, south of Canal St.) are extracted for nine Mondays in March and April of 2012, resulting in 2300-3500 trips per day. The simulated vehicles move along the Manhattan distance between pickup and drop-off locations, with speeds estimated from the data to account for congestion. The Financial District is divided into 15 regions: the center of each region (a “station”) is computed using k-means clustering on historical customer origin/destination data. In the simulation, customers are *not* required to go to a station to receive service: once a vehicle is assigned to a customer, it drives to the customer’s location and drops him/her off at the requested destination, then drives to the nearest station. The role of stations is to (1) model the availability of parking and charging facilities and (2) provide a discretized model for the vehicle rebalancing

[1]Courtesy of the New York City Taxi & Limousine Commission

problem.

For this simulation study, we implemented six dispatch algorithms, including two versions of Algorithm 1:

1. *Nearest-neighbor dispatch (NN):* Each customer is assigned the nearest free vehicle. If no vehicles are available, the customer request is added to a first-in, first-out queue. Free vehicles move according to a random walk until assigned to a customer.

2. *Collaborative dispatch (CD) [37]:* customer requests are aggregated in a queue (a single queue is used for the entire Financial District) and, when the queue size reaches a threshold x, the same number of free vehicles are dispatched to the customers. Vehicles are matched to customers to minimize the total distance they need to drive empty. The algorithm is modified from [37] in two ways: (1) a time-varying queue size $x(t)$ is employed to account for highly time-varying demand, and (2) the optimization is solved in a centralized fashion.

3. *Markov redistribution (MR) [102]:* Customer demand information is used to rebalance empty vehicles among stations in order to drive the vehicles' distribution towards the distribution of passenger arrivals. The problem is cast as a linear program (LP) and yields a randomized rebalancing strategy for each empty vehicle. The algorithm implemented in our simulation is modified from [102] in two ways: (1) the problem is solved exactly as an LP and (2) in order to accommodate time-varying demand, the algorithm rebalances vehicles based on the sum of (i) estimated future customer demand and (ii) current number of passengers waiting. Since the MR algorithm is randomized, its performance can vary widely between trials: for each day, the results presented are the median of ten executions.

4. *Real-time rebalancing (RR) (see Chapter 3.3.3):* Vehicles are rebalanced based on current waiting customers. At each rebalancing epoch (every 2 minutes) the algorithm computes the number of vehicles at or enroute to each station and solves a linear program to evenly distribute excess free vehicles throughout the system. It uses the same fifteen stations as the MR algorithm.

5. *Algorithm 1, MPC with sampled customer arrivals (MPCS):* The algorithm uses the same 15 stations employed by MR and solves the problem with a time horizon of 15 minutes. Customer arrival rates, computed using historical data, are sampled as a Poisson process and fed into Algorithm 1 as predicted future arrivals (c_{ij}). This sampling is done every 2 minutes to prevent unnecessary rebalancing. The rebalancing weight is set to $\rho_1 = 0.01$. A small term is added to the cost function to promote a uniform distribution of vehicles among the stations at the end of the optimization horizon.

6. *Algorithm 1, MPC with full arrival information (MPCF):* The actual customer arrivals over the time horizon (15 minutes) are fed into Algorithm 1 as c_{ij}. The algorithm optimizes vehicle

assignments with perfect knowledge of the next 15 minutes, and can therefore serve as a baseline for optimal performance, as long as wait times are small compared to the optimization horizon. The rebalancing weight is set to $\rho_1 = 0.01$.

Table 6.1 shows the peak customer wait times for each algorithm over the nine days that were simulated (every Monday from March 5 to April 30, 2012). The algorithm that yielded the best performance for each day (shortest peak wait time) is shown in bold (MPCF was excluded since it is non-causal). We note that in all the days where the peak wait time for MPCF is less than the optimization horizon of 15 minutes (day 3, 5, 6, and 9), MPCF achieves the best performance and can effectively serve as an optimal baseline. When the customer demand is high and the wait time is greater than the optimization horizon, the performance of MPC algorithms suffers: MPCS and MPCF are unable to effectively use information on future arrivals because new passengers are generally not serviced within the optimization horizon. Thus, in this regime, the MPCF algorithm is not a reliable optimal baseline. Nevertheless, MPCS achieves the best performance in 7 of the 9 days simulated. It's also worth noting that over the 4 days with short wait times, MPCS achieved peak wait times that were on average 34% shorter than the next best algorithm, RR. Figure 6.2 shows the simulation results for day 5 (April 2). In addition to achieving a lower peak wait time, the MPC algorithms are also able to recover quickly to service the remaining customers after peak demand. This is illustrated in Table 6.2, which lists the fraction of time spent for each algorithm where the average wait time was at least 50% of the respective peak. In this respect, MPCS again consistently outperforms the other algorithms.

A few observations about the other algorithms are in order. First, performance of the CD algorithm is generally slightly better than the NN algorithm, and is consistent with the results obtained by the authors in [37]. However, the problem of selecting a threshold queue size (i.e. the algorithm's tuning parameter) to maximize performance remains open. As a matter of fact, in three instances the NN algorithm outperforms CD in our simulations. Second, as neither NN nor CD rebalance empty vehicles so as to anticipate future demand, their performance highlights the critical importance of preemptive routing to achieve good quality-of-service. Third, since the MR algorithm is a randomized algorithm, its performance can be (and occasionally is) very suboptimal and, at times, significantly worse than NN. Median performance (over 10 runs), however, is generally better than both NN and CD. Finally, MR is conceived for steady-state systems. Thus, while the algorithm can be extended to time-varying systems, it is unclear whether the performance presented in [102] can be replicated in scenarios with highly variable customer demand.

The median runtime per iteration of Algorithm 1 on a 2.8 GHz Intel Core i7 PC with 16GB of RAM was 5.5 seconds. The other algorithms analyzed were significantly faster: the NN, CD, MR and RR algorithms had a median runtime per iteration of 2.2, 0.3, 18 and 332 ms respectively. Nevertheless, our results show that Algorithm 1 is amenable to a real-time implementation for a

[2]Note that the peak wait time for MPCS is only 3 minutes

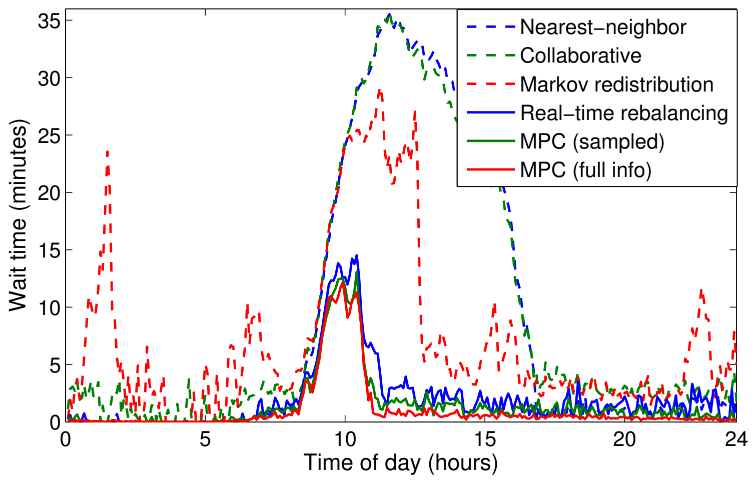


Figure 6.2: Average customer wait times throughout the day (April 2, 2012) for all dispatch algorithms.

Table 6.1: Peak wait time in minutes for each algorithm

Day	1	2	3	4	5	6	7	8	9
NN	36	34	27	56	36	9	38	41	24
CD	35	34	24	56	36	10	39	41	25
MR	37	53	11	31	29	13	32	35	16
RR	**16**	16	7	20	15	7	27	**21**	10
MPCS	17	**15**	**4**	**18**	**13**	**3**	**24**	24	**7**
MPCF*	18	19	3	19	12	2	29	24	6

moderately-sized system.

6.5.3 Effect of charge rate

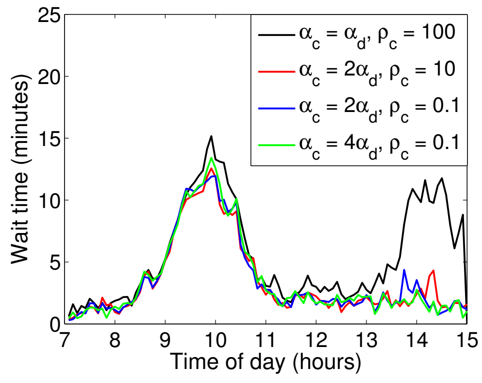
In this section we explore the performance limitations of AMoD systems with electric vehicles. Specifically, we would like to answer the question: how does charging rate affect the ability of an AMoD system to service customer demand? To this end, simulations were performed using 40 vehicles with taxi data from 7 am to 3 pm (period of high demand) on April 2, 2012. The simulations were performed for the MPCS algorithm (see Section 6.5.2) with charging constraints (Algorithm 2). The discharge rate, α_d, was chosen to be 0.0037, which corresponds to an electric vehicle such as a Nissan Leaf or BMW i3 driving at 20 km/h while using 70% of its battery capacity (to avoid over-discharging, which could damage the batteries). Three charging rates were used: $\alpha_c = \alpha_d$, $\alpha_c = 2\alpha_d$, and $\alpha_c = 4\alpha_d$, which correspond to a charging time of 4 hours, 2 hours, and 1 hour, respectively. The charging times represent realistic current electric vehicle charging capabilities. To avoid prematurely depleting the batteries, a higher final charging cost ρ_c was assigned to the

Table 6.2: Fraction of time where the average wait time was at least 50% of peak

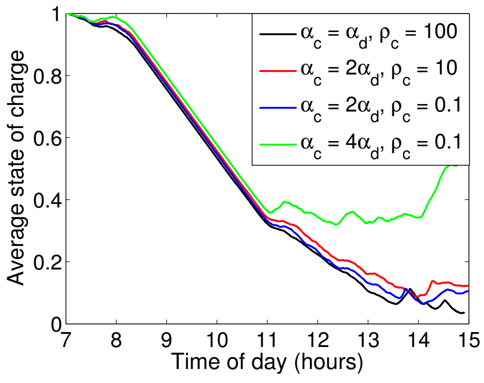
Day	1	2	3	4	5	6	7	8	9
NN	0.15	0.16	0.18	0.34	0.27	0.07	0.17	0.19	0.12
CD	0.15	0.16	0.16	0.35	0.26	**0.07**	0.17	0.19	0.11
MR	0.09	0.06	0.12	0.09	0.13	0.08	0.10	0.08	0.09
RR	0.08	0.07	0.11	0.09	0.07	0.07	**0.07**	0.06	0.05
MPCS	**0.06**	**0.05**	**0.09**	**0.08**	**0.06**	0.17[2]	0.08	**0.06**	**0.04**
MPCF*	0.07	0.04	0.06	0.08	0.06	0.08	0.07	0.06	0.03

cases with slower charge rates. Figure 6.3(a) shows the customer wait times for the four simulations performed, and 6.3(b) shows the average state-of-charge of vehicles over time.

We first note that as long as there is excess battery capacity in the system, the customer wait times for an electric AMoD system are comparable to the AMoD system without charging constraints (see Figure 6.2). In this case, a charge rate of $4\alpha_d$ is stabilizing and is able to support future demand. A charge rate of $2\alpha_d$ is able to service demand without increasing wait time, but the batteries are almost fully depleted by the end of the simulation. A charge rate of α_d is too slow to support customer demand for the entire simulation duration. The large final charging cost $\rho_c = 100$ trades off quality of service with battery capacity, resulting in a slightly higher peak wait time. Even so, batteries become depleted near the end of the simulation period and the algorithm begins to prefer battery charging over servicing customers, resulting in longer wait times.



(a)



(b)

Figure 6.3: 6.3(a): Average customer wait time for April 2, with charging constraints. 6.3(b): Average vehicle charge as a function of time for different charging rates and final charging costs.

6.6 Conclusion and Additional Model Extensions

In this chapter we presented a model predictive control approach to optimize vehicle scheduling and routing in an AMoD system. The approach allows the easy integration of a number of operational constraints – in particular we focused on charging constraints. We presented two MPC algorithms and rigorously showed that they are able to regulate an AMoD system (i.e., drive an initial customer demand to zero assuming no additional customers arrive over time). Algorithm performance for the case of dynamic arrivals was evaluated using real-world data through simulations. Overall, numerical results showed that the proposed MPC algorithms outperformed other control strategies for AMoD systems.

While we have mainly focused on extending the AMoD model for charging constraints, many other real-world constraints can be directly incorporated into our modeling framework with little modification. We briefly touch on some of the possible extensions to highlight the flexibility and potential of our approach.

1. **Limited number of charging stations**. Let $h_i, i \in \mathcal{N}$, represent the number of charging stations at each station. We can assume that $\sum_{i\in\mathcal{N}} h_i \geq |V|$, that is, there is at least one charger for each vehicle in the system. Vehicles can only wait at a charging station, though they can still pick up and drop off passengers at stations with no free charging stations. This adds the constraint $\sum_{k\in V} u_i^k(t) \leq h_i$ for all $i \in \mathcal{N}$. A limited number of charging stations will also promote rebalancing, as it forces vehicles to travel to stations with free charging stations (or likely a deficit of vehicles).

2. **Customer priorities**. Taking into account priority waiting involves simply adding a weighting matrix to the objective function. Rather than minimizing $\sum_{\tau=t}^{t+t_{\text{hor}}-1} J_x(x(\tau+1))$ we minimize $\sum_{\tau=t}^{t+t_{\text{hor}}-1} Q(\tau+1)J_x(x(\tau+1))$. In this way, we can give a higher priority to customers who have been waiting for a longer period of time, or assign the weights based on price incentives. This approach can also be used for customer arrivals with known time windows.

3. **Interaction with the smart grid**. Electric vehicles may act as energy storage devices to enable intermittent renewable energy such as solar and wind [7]. Vehicles can "sell" their energy to the grid during peak hours and charge themselves during off-peak hours. If the charging/discharging schedule is known, the charging rates α_c can be adjusted accordingly to facilitate the energy transfer, at the same time maintaining quality of service in the AMoD system.

The key limitation of this approach is scaling up the MILP formulation of the optimization problem to large-scale systems. Since the computational complexity of the MILP formulation scales exponentially with the number of stations and vehicles, this will likely require the use of parallel architectures and ad hoc approximations. Nonetheless, at a medium scale, the MPC algorithms are useful for benchmarking the performance of other rebalancing algorithms.

Chapter 7

Conclusions

The modeling approaches and algorithms developed in this dissertation collectively provide the first rigorous study of the operational challenges for autonomous mobility-on-demand systems. Our methodology consists of three steps: (1) rigorous theoretical models capturing specific operational or structural challenges we wish to address, (2) practical coordination algorithms suitable for large-scale systems, and (3) evaluation of societal impact via case studies using real-world data.

The theoretical models we have developed can be used to to make strategic decisions, such as determining the number of vehicles needed in a given city and the number and locations of charging stations in the case of an electric vehicle fleet. The coordination algorithms provide insight into how to route a large fleet of autonomous vehicles in a computationally practical way. Our case studies of New York City and Singapore showed that significant performance gains are possible through fleet-level coordination which will translate into reduced costs for system operators and reduced prices for customers. Finally, we showed that empty vehicle trips would not increase traffic congestion in a city if performed in an intelligent way.

The field of planning and coordination for future mobility systems is still in its infancy. As MoD systems continue to gain popularity and AMoD systems begin to emerge, solving the problems of large-scale coordination will become increasingly vital to the creating a sustainable urban transportation network.

To conclude, we summarize the results presented in each chapter, and end with a discussion on future directions of research.

7.1 Summary

In Chapter 3, we studied a queueing-theoretical model of an AMoD system with vehicle rebalancing using the theory of Jackson networks. We showed that an optimal open-loop rebalancing policy can be solved as a linear program and developed a scalable closed-loop rebalancing algorithm suitable for large systems. We demonstrated the performance of the algorithms using case studies of Manhattan and Singapore.

In Chapter 4, we extended the Jackson network model for AMoD systems to conventional MoD systems where rebalancing is performed by human drivers. We formulated optimal open-loop rebalancing policies and showed how they could be used for making strategic decisions such as fleet sizing.

In Chapter 5, we studied the impact of AMoD on traffic congestion. In particular, using a network flow model of AMoD we showed that rebalancing will *not* increase congestion if done in an intelligent fashion. We provided a scalable real-time congestion-aware rebalancing algorithm that limits rebalancing if it could cause additional congestion, and we showed through simulation studies that congestion does not increase under the congestion-aware algorithm.

Finally, in Chapter 6 we studied additional operational complexities and constraints for AMoD systems, with a focus on charging constraints. We developed an MPC framework for routing vehicles in an AMoD system subject to range and charging constraints, and we showed that the MPC algorithm can also serve as a performance benchmark for evaluating rebalancing algorithms.

7.2 Future Directions

The models and algorithmic approaches presented in this dissertation lay the foundations for many avenues of further research. Several future directions of research have been mentioned throughout the dissertation; other promising directions for research are outlined in the following.

Queueing network model with congestion constraints It is of interest to extend the queueing-theoretical model presented in Chapter 3 to take into account congestion constraints. One way to accomplish this is to model a road network using the theory of BCMP networks [69], where each road link is a queue with possibly load-dependent service. A different routing matrix can be defined for vehicles traveling from each origin to each destination, and optimized for quality of service (e.g. vehicle availability) or for congestion (minimizing travel times). This approach would permit the computation of quality of service metrics while simultaneously solving the routing and rebalancing problem.

Demand-aware stations In one possible realization of an AMoD system, customers may request transportation through an app on their mobile devices, and vehicles would drive themselves to the customers' locations to provide service. In such a scenario, a "station" may no longer represent a physical location, but a region from which customer requests arrive. Indeed, the locations of these stations can be dynamically adjusted reflect the spatial distribution of customer arrivals at each point in time. New stations may also be dynamically generated if sufficient demand is concentrated in one location, and old stations may be removed if they no longer generate sufficient demand. A dynamic clustering algorithm such as [65] can be used to determine the number and locations of the stations in the presence of time-varying customer demand.

Demand staggering to reduce congestion One potential way to further reduce congestion using AMoD is to stagger customer demands. In this problem, each customer may be given a time window for pickup or a desired time-of-arrival at the destination. Because traffic congestion is a highly nonlinear phenomenon, the hope is that a small amount of vehicle reduction on the road can translate into large time savings. If this is the case, for many customers, even though the pickup time may be delayed, a shorter travel time resulting from reduced congestion may compensate for delayed pickup resulting in on-time delivery.

Additional performance metrics for evaluation The performance metrics for the algorithms presented in this thesis consisted mainly of mean values (such as mean wait times). In real-life scenarios, however, it is important to also take into account the higher moments of wait time distributions and the maximum wait times. Thus it is of interest to investigate other optimization objectives such as minimizing the maximum customer wait time. This problem may be formulated using techniques from risk-constrained optimization.

Intermodal AMoD An interesting application for AMoD is the so-called "first-mile/last-mile" problem. Because public transit operate on fixed routes, the idea is that current public transit infrastructure can deliver most travelers to within a mile of their destinations, but cannot service the last leg of journey. An AMoD system may be integrated with public transportation to provide local transportation services and thus solve the "first-mile/last-mile" problem. This problem is interesting from both a routing and scheduling perspective. How should the vehicles distribute themselves to minimize wait times with the knowledge that many customers will come from fixed public transit stations? How should a customer decide whether to use AMoD service for the entire trip or just to the nearest bus/train station?

Rebalancing for MoD In Chapter 4 we studied a model for human-driven MoD systems where rebalancing drivers were hired to rebalance empty vehicles. The drivers themselves were rebalanced by riding with customer-carrying vehicles. It is of interest to investigate additional modes of rebalancing these drivers. Each driver can be driven (by a van) to a location where a vehicle requires rebalancing, then be picked up after he/she completes the rebalancing trip. Alternatively, a driver may use a vehicle (in the MoD system) to transport multiple other drivers to locations with excess vehicles.

Evaluation of algorithms through high fidelity traffic simulation The coordination algorithms we presented were evaluated using low-fidelity simulations where vehicles traveled at a constant average speed. Delays such as traffic lights, turning durations, and pedestrians were not taken into account. To gain a better understanding of the performance of these algorithms, they should be tested using a microscopic traffic simulator[103] that models a detailed road network and vehicle-to-vehicle interactions.

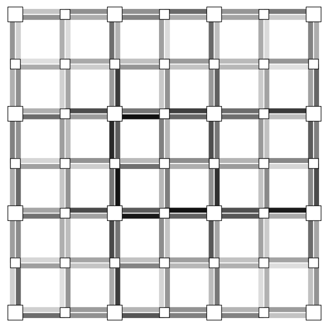
Decentralized architectures for cooperative routing Thus far we have only considered algorithms where decisions are made by a central controller. This centralized architecture may represent a single point of failure vulnerability in the system. Alternatively, vehicles may communicate directly with other vehicles in their vicinity and make routing and coordination decisions cooperatively. Such a decentralized scheme would likely sacrifice some degree of optimality, compared to the centralized approach, but nonetheless has its advantages, such as the ability for vehicles to continue to operate if its connection with the central server is severed. This would likely make the system more robust to failures and attacks.

Appendices

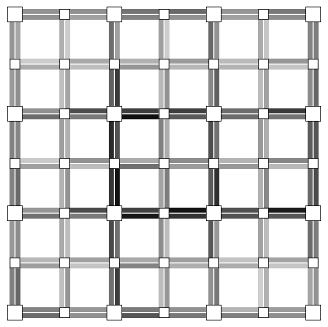
Appendix A

Additional numerical results of congestion on a simple road network

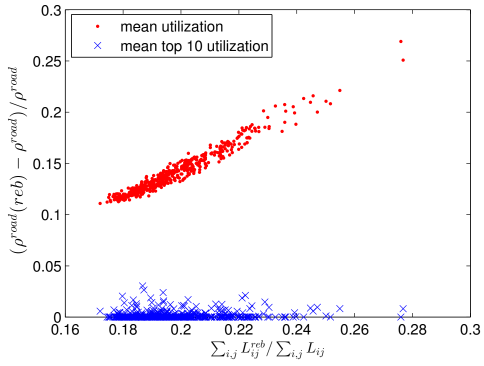
We further validate the numerical study in Chapter 5.2 by considering a larger 7×7 grid road network consisting of 16 stations (one station every 2 blocks). The larger grid permits more routes between stations and is more representative of a real road network than the 3×3 grid 9-station example. The same analysis is carried out for this road network and shown in Figure A.1. Rather than focusing on the most congested road segment as in the 9-station example, we look at the 10 most congested road segments in the network, and whether rebalancing increases the congestion on these segments. From Figure A.1(c), we see that the increases in utilization on the 10 most congested roads is far less than the average utilization increase (an even more promising result than the 9-station case). In Figure A.1(d), the rebalancing correction scheme is applied and the number of systems with increased top 10 road utilization (crosses) is reduced from 143 (28.6%) to 25 (5%). This is achieved at the expense of slightly higher overall rebalancing rates, but with the majority of rebalancing occurring along less utilized routes.



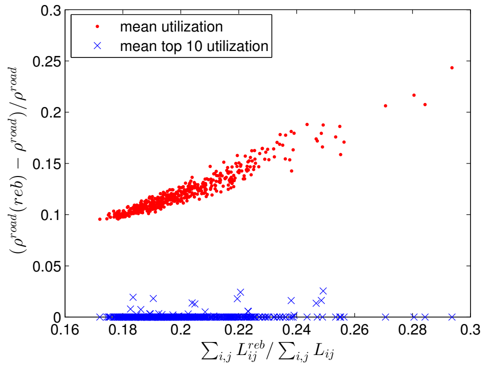
(a)



(b)



(c)



(d)

Figure A.1: A.1(a): A randomly generated system without rebalancing on a 7×7 square grid road network with 16 stations, shown with large squares. Small squares mark the intersections. White represents low congestion and black represents heavy congestion. A.1(b): The same system with rebalancing. A.1(c): 500 randomly generated systems with different arrival rates and routing distributions on the 7×7 grid 16-station road network. The x-axis is the ratio of rebalancing vehicles to passenger vehicles on the road. The y-axis is the fractional increase in road utilization due to rebalancing. The dots show the mean increase in road utilization. The crosses show the mean increase in utilization of the top 10 most congested road segments. A.1(d): The same 500 systems with adjusted rebalancing rates.

Appendix B

Simulation results of MPC Algorithm

This appendix contains the simulation results comparing the MPC algorithm to other mobility-on-demand dispatch algorithms using New York taxi data from March 19, April 9, and April 30, 2012. Description of each algorithm is given in Chapter 6.5.2.

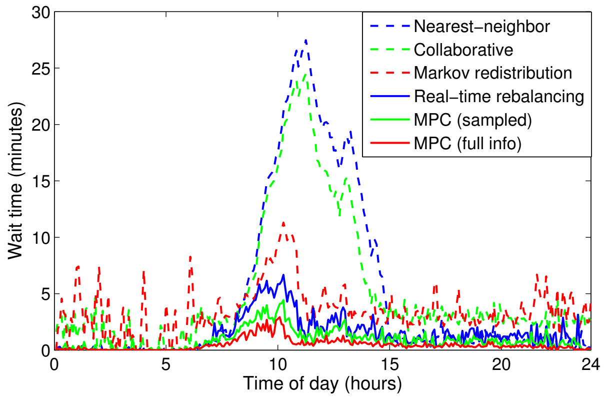


Figure B.1: Average customer wait times throughout the day (March 19, 2012) for all dispatch algorithms.

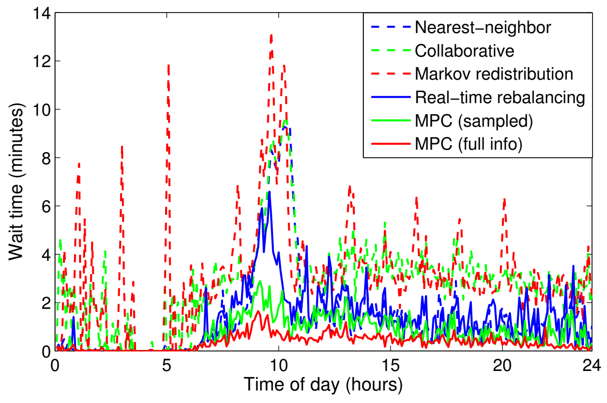


Figure B.2: Average customer wait times throughout the day (April 9, 2012) for all dispatch algorithms.

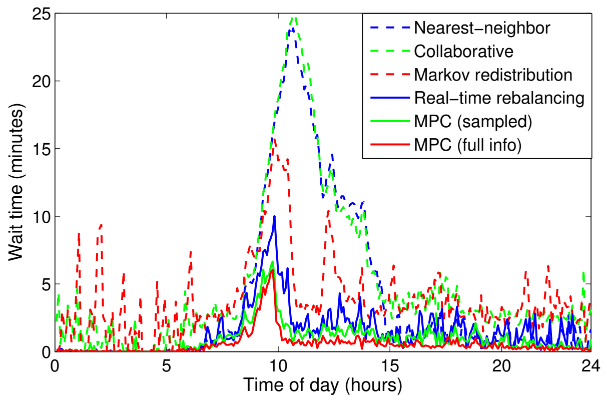


Figure B.3: Average customer wait times throughout the day (April 30, 2012) for all dispatch algorithms.

Appendix C

Capacity Symmetry within Urban Centers in the US

The existential result in Section 5.4, Theorem 5.4.5, relies on the assumption that the road network is capacity-symmetric, i.e., for every cut $(\mathcal{S}, \bar{\mathcal{S}}), C_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) = C_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})$. One may wonder whether this assumption is (approximately) met in practice. From an intuitive standpoint, one might argue that transportation networks within urban centers are indeed *designed* to be capacity symmetric, so as to avoid accumulation of traffic flow in some directions. We corroborate this intuition by computing the imbalance between the outbound capacity (i.e., C_{out}) and the inbound capacity (i.e., C_{in}) for 1000 randomly-selected cuts within several urban centers in the United States. For each edge $(u, v) \in \mathcal{E}$, we approximate its capacity as proportional to the product of the speed limit $v_{\max}(u, v)$ on that edge and the number of lanes $L(u, v)$, that is, $c(u, v) \propto v_{\max}(u, v) \cdot L(u, v)$. The road graph $G(\mathcal{V}, \mathcal{E})$, the speed limits, and the number of lanes are obtained from OpenStreetMap data [88].

For a cut $(S, \bar{S})$, we define its fractional capacity disparity $D(\mathcal{S}, \bar{\mathcal{S}})$ as

$$D(\mathcal{S}, \bar{\mathcal{S}}) := 2\,\frac{\left|C_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) - C_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})\right|}{C_{\text{out}}(\mathcal{S}, \bar{\mathcal{S}}) + C_{\text{in}}(\mathcal{S}, \bar{\mathcal{S}})}.$$

Table C shows the average (over 1000 samples) fractional capacity disparity for several US urban centers. As expected, the road networks for such cities appear to possess a very high degree of capacity-symmetry, which validates the symmetry assumption made in Section 5.4.

Table C.1: Average fractional capacity disparity for several major urban centers in the United States.

Urban center	Avg. frac. capacity disparity	Std. dev.
Chicago, IL	$1.2972 \cdot 10^{-4}$	$1.003 \cdot 10^{-4}$
New York, NY	$1.6556 \cdot 10^{-4}$	$1.304 \cdot 10^{-4}$
Colorado Springs, CO	$3.1772 \cdot 10^{-4}$	$2.308 \cdot 10^{-4}$
Los Angeles, CA	$0.9233 \cdot 10^{-4}$	$0.676 \cdot 10^{-4}$
Mobile, AL	$1.9368 \cdot 10^{-4}$	$1.452 \cdot 10^{-4}$
Portland, OR	$1.0769 \cdot 10^{-4}$	$0.778 \cdot 10^{-4}$

Bibliography

[1] Bureau of Transportation Statistics, "Pocket Guide to Transportation," U.S. Department of Transportation, Tech. Rep., 2016.

[2] U. S. Energy Information Administration, "International Energy Outlook 2013," Tech. Rep., 2013.

[3] United Nations Environment Programme, "The Emissions Gap Report 2013 - UNEP," United Nations, Tech. Rep., 2013.

[4] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "Urban Mobility Scorecard," Texas A&M Transportation Institute and INRIX, Inc, Tech. Rep., 2015.

[5] L. D. Burns, W. C. Jordan, and B. A. Scarborough, "Transforming personal mobility," *Earth Island Institute, Columbia University*, 2013.

[6] UN, "World Urbanization Prospects: The 2014 Revision," United Nations, Tech. Rep., 2014.

[7] W. J. Mitchell, C. E. Borroni Bird, and L. D. Burns, *Reinventing the Automobile: Personal Urban Mobility for the 21st Century.* Cambridge, MA: The MIT Press, 2010.

[8] C. Wald, "Wheels when you need them," *Science*, vol. 345, no. 6199, pp. 862–863, 2014.

[9] A. Schmauss, "Car2go in Ulm, Germany, as an advanced form of car-sharing," *European Local Transport Information Service (ELTIS)*, 2009.

[10] C. Fricker and N. Gast, "Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity," *EURO Journal on Transportation and Logistics*, pp. 1–31, 2012.

[11] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," *Omega*, vol. 45, pp. 7–19, 2014.

[12] T. Raviv, M. Tzur, and I. A. Forma, “Static repositioning in a bike-sharing system: models and solution approaches,” *European Journal of Transportation and Logistics*, vol. 2, pp. 187–229, 2013.

[13] L. Di Gaspero, A. Rendl, and T. Urli, “Constraint-based approaches for balancing bike sharing systems,” in *Principles and Practice of Constraint Programming.* Springer, 2013, pp. 758–773.

[14] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve, “Inventory rebalancing and vehicle routing in bike sharing systems,” Tech. Rep., 2013.

[15] S. Herrmann, F. Schulte, and S. Voß, “Increasing acceptance of free-floating car sharing systems using smart relocation strategies: a survey based study of car2go Hamburg,” in *Computational Logistics.* Springer, 2014, pp. 151–162.

[16] D. Jorge, G. H. A. Correia, and C. Barnhart, “Comparing Optimal Relocation Operations With Simulated Relocation Policies in One-Way Carsharing Systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1667–1675, 2014.

[17] R. Nair and E. Miller Hooks, “Fleet management for vehicle sharing operations,” *Transportation Science*, vol. 45, no. 4, pp. 524–540, 2011.

[18] Clemens Grupp, “The Relocation Planning Problem in Free-Floating Car Sharing: Models and Heuristics,” Master’s thesis, Technische Universitat Munchen, 2015.

[19] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, “Rebalancing the Rebalancers: Optimally Routing Vehicles and Drivers in Mobility-On-Demand Systems,” in *American Control Conference*, Washington, DC, Jun. 2013, pp. 2362–2367.

[20] R. Zhang and M. Pavone, “A Queueing Network Approach to the Analysis and Control of Mobility-On-Demand Systems,” in *American Control Conference*, Chicago, IL, Jul. 2015, pp. 4702–4709.

[21] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, “Robotic Load Balancing for Mobility-On-Demand Systems,” *International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, Jun. 2012.

[22] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone, “Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-On-Demand Systems: A Case Study in Singapore,” in *Lecture Notes in Mobility.* Springer, Jun. 2014, pp. 229–245.

[23] M. Ben Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani, “DynaMIT: a simulation-based system for traffic prediction.” Citeseer, 1998.

[24] J. G. Wardrop, "Some Theoretical Aspects of Road Traffic Research," in *ICE Proceedings: Engineering Divisions*, vol. 1, no. 3. Thomas Telford, 1952, pp. 325–362.

[25] S. Peeta and A. Ziliaskopoulos, "Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future," *Networks and Spatial Economics*, vol. 1, pp. 233–265, 2001.

[26] Q. Yang and H. N. Koutsopoulos, "A microscopic traffic simulator for evaluation of dynamic traffic management systems," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 3, pp. 113–129, 1996.

[27] M. Treiber, A. Hennecke, and D. Helbing, "Microscopic simulation of congested traffic," in *Traffic and Granular Flow 99*. Springer, 2000, pp. 365–376.

[28] M. Treiber and D. Helbing, "Macroscopic simulation of widely scattered synchronized traffic states," *Journal of Physics A: Mathematical and General*, vol. 32, no. 1, p. L17, 1999.

[29] A. G. H. Kek, R. L. Cheu, Q. Meng, and C. H. Fung, "A decision support system for vehicle relocation operations in carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 1, pp. 149–158, 2009.

[30] D. K. George and C. H. Xia, "Fleet-sizing and service availability for a vehicle rental system via closed queueing networks," *European Journal of Operational Research*, vol. 211, no. 1, pp. 198–207, 2011.

[31] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications.* SIAM, 2014, vol. 18.

[32] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[33] S. Anily and R. Hassin, "The swapping problem," *Networks*, vol. 22, no. 4, pp. 419–433, 1992.

[34] K. Treleaven, M. Pavone, and E. Frazzoli, "Asymptotically Optimal Algorithms for One-to-One Pickup and Delivery Problems With Applications to Transportation Systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2261–2276, Sep. 2013.

[35] F. A. T. Montané and R. D. Galvao, "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service," *Computers & Operations Research*, vol. 33, no. 3, pp. 595–619, 2006.

[36] H. N. Psaraftis, "Dynamic Vehicle Routing Problems," in *Vehicle Routing: Methods and Studies*, B. Golden and A. Assad, Eds. Elsevier (North-Holland), 1988, pp. 223–248.

[37] K. T. Seow, N. H. Dang, and D. H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Transactions on Automation Sciences and Engineering*, vol. 7, no. 3, pp. 607–616, 2010.

[38] E. Frazzoli and M. Pavone, "Multi-Vehicle Routing," in *Encyclopedia of Systems and Control.* Springer, Apr. 2014, pp. 1–11.

[39] A. Alshamsi, S. Abdallah, and I. Rahwan, "Multiagent self-organization for a taxi dispatch system," in *8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 21–28.

[40] D. J. Fagnant, K. M. Kockelman, and P. Bansal, "Operations of Shared Autonomous Vehicle Fleet for Austin, Texas, Market," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2536, pp. 98–106, 2015.

[41] P. Jaillet and M. R. Wagner, "Online Routing Problems: Value of Advanced Information and Improved Competitive Ratios," *Transportation Science*, vol. 40, no. 2, pp. 200–210, 2006.

[42] E. Feuerstein and L. Stougie, "On-line single-server dial-a-ride problems," *Theoretical Computer Science*, vol. 268, no. 1, pp. 91–105, 2001.

[43] D. J. Bertsimas and G. J. van Ryzin, "A Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane," *Operations Research*, vol. 39, pp. 601–615, 1991.

[44] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic Vehicle Routing for Robotic Systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, Jul. 2011.

[45] M. Pavone, "Dynamic Vehicle Routing for Robotic Networks," Ph.D. dissertation, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics. Cambridge, MA, Jun. 2010. [Online]. Available: http://18.7.29.232/handle/1721.1/59665

[46] K. Treleaven, M. Pavone, and E. Frazzoli, "Models and Efficient Algorithms for Pickup and Delivery Problems on Roadmaps," in *Proc. IEEE Conf. on Decision and Control*, Maui, HI, Dec. 2012, pp. 5691–5698.

[47] L. Rüschendorf, "The Wasserstein distance and approximation theorems," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 70, no. 1, pp. 117–129, 1985.

[48] R. Zhang and M. Pavone, "Control of Robotic Mobility-On-Demand Systems: A Queueing-Theoretical Perspective," *International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.

[49] R. Zhang, K. Spieser, E. Frazzoli, and M. Pavone, "Models, Algorithms and Evaluation for Autonomous Mobility-On-Demand Systems," in *American Control Conference*, Chicago, IL, Jul. 2015.

[50] R. Zhang, F. Rossi, and M. Pavone, "Routing Autonomous Vehicles in Congested Transportation Networks: Structural Properties and Coordination Algorithms," in *Robotics: Science and Systems*, Ann Arbor, MI, Jun. 2016.

[51] ——, "Model Predictive Control of Autonomous Mobility-on-Demand Systems," in *Proc. IEEE Conf. on Robotics and Automation*, 2016.

[52] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*, Z. . Associates, Ed. Prentice-Hall International, 1992, vol. 2.

[53] E. Cinlar, *Introduction to stochastic processes.* Courier Corporation, 2013.

[54] J. R. Jackson, "Networks of waiting lines," *Operations research*, vol. 5, no. 4, pp. 518–521, 1957.

[55] R. Serfozo, *Introduction to stochastic networks.* Springer, 1999, vol. 44.

[56] S. S. Lavenberg, *Computer performance modeling handbook.* Elsevier, 1983, vol. 4.

[57] Lester Randolph Ford and Delbert Ray Fulkerson, *Flows in Networks.* Princeton University Press, 1962.

[58] R. M. Karp, "On the computational complexity of combinatorial problems," in *Networks, Networks (USA), (Proceedings of the Symposium on Large-Scale Networks, Evanston, IL, USA, 18-19 April 1974.)*, vol. 5, no. 1, Jan. 1975, pp. 45–68.

[59] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, *Network Flows: Theory, Algorithms and Applications.* Upper Saddle River, New Jersey 07458: Prentice Hall, 1993.

[60] A. Waserhole and V. Jost, "Pricing in vehicle sharing systems: Optimization in queuing networks with product forms." OSP. At hal.archives-ouvertes.fr, 2013.

[61] R. C. Larson and A. R. Odoni, *Urban Operations Research.* Prentice Hall, 1981.

[62] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* SIAM, 2001.

[63] M. Reiser and S. S. Lavenberg, "Mean-value analysis of closed multichain queuing networks," *Journal of the Association for Computing Machinery*, vol. 27, no. 2, pp. 313–322, 1980.

[64] B. Kulis and M. I. Jordan, "Revisiting k-means: New Algorithms via Bayesian Nonparametrics," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 513–520.

[65] T. Campbell, M. Liu, B. Kulis, J. P. How, and L. Carin, "Dynamic clustering via asymptotics of the Dependent dirichlet process mixture," in *Advances in Neural Information Processing Systems*, 2013, pp. 449–457.

[66] Land Transport Authority, “Singapore land transit statistics in brief,” 2012.

[67] S. M. Madanat, M. J. Cassidy, and M. Wang, “Probabilistic delay model at stop-controlled intersection,” *Journal of Transportation Engineering*, vol. 120, no. 1, pp. 21–36, 1994.

[68] B. S. Kerner, *Introduction to modern traffic flow theory and control: the long road to three-phase traffic theory.* Springer Science & Business Media, 2009.

[69] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, “Open, Closed, and Mixed Networks of Queues with Different Classes of Customers,” *Journal of the Association for Computing Machinery*, vol. 22, no. 2, pp. 248–260, Apr. 1975.

[70] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed., ser. Algorithmics and Combinatorics. Springer, 2007, vol. 21.

[71] *ILOG CPLEX User's guide*, ILOG, Mountain View, CA, 1999.

[72] J. Pérez, F. Seco, V. Milanés, A. Jiménez, J. C. Díaz, and T. De Pedro, “An RFID-based Intelligent Vehicle Speed Controller Using Active Traffic Signals,” *Sensors*, vol. 10, no. 6, pp. 5872–5887, 2010.

[73] B. Templeton, “Traffic Congestion & Capacity,” 2015, available at http://www.templetons.com/brad/robocars/congestion.html.

[74] M. Barnard, “Autonomous Cars Likely to Increase Congestion,” Jan. 2016, available at http://cleantechnica.com/2016/01/17/autonomous-cars-likely-increase-congestion.

[75] Michael W. Levin, Tianxin Li, Stephen D. Boyles, and Kara M. Kockelman, “A general framework for modeling shared autonomous vehicles,” in *95th Annual Meeting of the Transportation Research Board*, 2016.

[76] M. J. Lighthill and G. B. Whitham, “On kinematic waves. I. Flood movement in long rivers,” in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 229, no. 1178. The Royal Society, 1955, pp. 281–316.

[77] C. F. Daganzo, “The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory,” *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[78] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K. Axhausen, “MATSim-T: Architecture and simulation times,” *Multi-agent systems for traffic and transportation engineering*, pp. 57–78, 2009.

[79] C. Osorio and M. Bierlaire, “An analytic finite capacity queueing network model capturing the propagation of congestion and blocking,” *European Journal of Operational Research*, vol. 196, no. 3, pp. 996–1007, 2009.

[80] S. Peeta and H. S. Mahmassani, “System optimal and user equilibrium time-dependent traffic assignment in congested networks,” *Annals of Operations Research*, vol. 60, no. 1, pp. 81–113, 1995.

[81] B. N. Janson, “Dynamic traffic assignment for urban road networks,” *Transportation Research Part B: Methodological*, vol. 25, no. 2, pp. 143–161, 1991.

[82] T. Le, P. Kovács, N. Walton, H. L. Vu, L. L. H. Andrew, and S. S. P. Hoogendoorn, “Decentralized signal control for urban road networks,” *Transportation Research Part C: Emerging Technologies*, 2015.

[83] N. Xiao, E. Frazzoli, Y. Luo, Y. Li, Y. Wang, and D. Wang, “Throughput optimality of extended back-pressure traffic signal control algorithm,” in *Control and Automation (MED), 2015 23th Mediterranean Conference on.* IEEE, 2015, pp. 1059–1064.

[84] M. Papageorgiou, H. Hadj Salem, and J.-M. Blosseville, “ALINEA: A local feedback control law for on-ramp metering,” *Transportation Research Record*, no. 1320, 1991.

[85] Bureau of Public Roads, “Traffic Assignment Manual,” U.S. Department of Commerce, Urban Planning Division, Washington, D.C (1964), Tech. Rep., 1964.

[86] B. S. Kerner, “Traffic Congestion, Modeling Approaches to,” in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. Springer New York, 2009, pp. 9302–9355.

[87] H. Neuburger, “The economics of heavily congested roads,” *Transportation Research*, vol. 5, no. 4, pp. 283 – 293, 1971.

[88] M. Haklay and P. Weber, “OpenStreetMap: User-Generated Street Maps,” *Pervasive Computing, IEEE*, vol. 7, no. 4, pp. 12–18, Oct 2008.

[89] Andrew V. Goldberg, Eva Tardos, and Robert E. Tarjan, “Network Flow Algorithms,” Cornell University Operations Research and Industrial Engineering, Tech. Rep., 1989.

[90] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas, “Fast approximation algorithms for multicommodity flow problems,” *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 228–243, 1995.

[91] A. V. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein, “An Implementation of a Combinatorial Approximation Algorithm for Minimum-Cost Multicommodity Flow,” in *Integer Programming and Combinatorial Optimization*, ser. Lecture Notes in Computer

Science, R. Bixby, E. Boyd, and R. Ros-Mercado, Eds. Springer Berlin Heidelberg, 1998, vol. 1412, pp. 338–352.

[92] S. Even, A. Itai, and A. Shamir, "On the Complexity of Timetable and Multicommodity Flow Problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[93] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, July 1968.

[94] N. Kliewer, T. Mellouli, and L. Suhl, "A time–space network based exact optimization model for multi-depot bus scheduling," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1616–1627, 2006.

[95] S. Lin, B. De Schutter, Y. Xi, and H. Hellendoorn, "Efficient network-wide model-based predictive control for urban traffic networks," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 122–140, 2012.

[96] R. R. Negenborn, B. De Schutter, and J. Hellendoorn, "Multi-agent model predictive control for transportation networks: Serial versus parallel schemes," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 3, pp. 353–366, 2008.

[97] C. Danielson, F. Borrelli, D. Oliver, D. Anderson, M. Kuang, and T. Phillips, "Balancing of battery networks via constrained optimal control," in *American Control Conference*, 2012, pp. 4293–4298.

[98] E. Franco, L. Magni, T. Parisini, M. M. Polycarpou, and D. M. Raimondo, "Cooperative constrained control of distributed agents with nonlinear dynamics and delayed information exchange: A stabilizing receding-horizon approach," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 324–338, 2008.

[99] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[100] SuperShuttle, "How It Works," 2015.

[101] J. Aubin and H. Frankowska, *Set-valued analysis.* Birkhäuser, 1990.

[102] M. Volkov, J. Aslam, and D. Rus, "Markov-based redistribution policy model for future urban mobility networks," in *IEEE Conference on Intelligent Transportation Systems.* IEEE, 2012, pp. 1906–1911.

[103] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM2002)*, 2002, pp. 183–187.