MODELS AND ALGORITHMS FOR TRANSPORTATION IN THE SHARING ECONOMY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Daniel Freund

August 2018

MODELS AND ALGORITHMS FOR TRANSPORTATION IN THE SHARING ECONOMY

Daniel Freund, Ph.D.

Cornell University 2018

This thesis consist of two parts. The first deals with bike-sharing systems which are now ubiquitous across the U.S.A. We have worked with Motivate, the operator of the systems in, for example, New York City, Chicago, and San Francisco, to innovate a data-driven approach to managing both their day-to-day operations and to provide insight on several central issues in the design of their systems. This work required the development of a number of new optimization models, characterizing their mathematical structure, and using this insight in designing algorithms to solve them. Many of these projects have been fully implemented to improve the design, rebalancing, and maintenance of Motivate's systems across the country.

In the second part, we study a queueing-theoretic model of on-demand transportation systems (e.g., Uber/Lyft, Scoot, etc.) to derive approximately optimal pricing, dispatch, and rebalancing policies. Though the resulting problems are high-dimensional and non-convex, we develop a general approximation framework, based on a novel convex relaxation. Our approach provides efficient algorithms with rigorous approximation guarantees for a wide range of objectives and controls.

BIOGRAPHICAL SKETCH

Daniel Freund grew up in Cologne, Germany, and graduated from the Städtische Apostelgymnasium in 2009. Subsequently, he spent time on Kibbutz Ma'agan Michael in Israel before completing his undergraduate education in Mathematics at the University of Warwick in Coventry (UK). During his undergraduate, he was given the opportunity to participate in the Weizmann Institute's 2012 Kupcinet-Getz International Summer School. After graduating in 2013 he enrolled as a Ph.D. student at the Center for Applied Mathematics at Cornell University in Ithaca, New York. He spent the summer and fall of 2015 as a Data Scientist at Motivate, discovering many of the problems discussed in this thesis. After graduating from Cornell he will join Lyft as post-doctoral research fellow for one year before joining the faculty at MIT as an Assistant Professor of Operations Management in 2019.

Dedicated to those who (nevertheless) persisted.

And a family of elephants…

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor, David B. Shmoys. This thesis would not have been possible if it were not for his guidance, mentorship, and unwavering support and he has become a role model for me in more than one way. I also want to thank David P. Williamson and Jon M. Kleinberg for serving on my committee and giving valuable feedback. Throughout my time at Cornell I had the privilege to work with many other Cornell faculty, most importantly Shane G. Henderson and Siddhartha Banerjee – being infected by their scientific curiosity was perhaps the most important lesson I learned in graduate school!

Before I became interested in the mixture of data and theory, and transportation and economics that is at the core of this thesis, I had great mentors who helped me walk my first research steps: Moni Naor, Ben Fisch, Matthias Poloczek, and Daniel Reichman with whom I wrote my first papers, Bobby Kleinberg and Éva Tardos who taught me most of my theoretical foundations, and John Hopcroft, who encouraged me to explore my interests.

Most of the results in this thesis were the product of collaborations with other graduate students, whose brilliance I benefited from: I am particularly grateful to Thodoris Lykouris and Alice Paul as well as Nanjing Jian and Ashkan Norouzi-Fard. Through Cornell's Citi Bike research group I also had the opportunity to work with a number of talented undergraduate students, including Holly M. Wiberg, Aaron M. Ferber, and Hangil Chung among others; I am looking forward to seeing many more results from them.

The first part of this thesis would not have been possible without a long-lasting collaboration between our group at Cornell and Motivate. This collaboration allowed me to spend 7 productive months at Motivate, which was enabled in large part by Jacob Doctoroff, Emily Gates, Jay Walder, Jules Flynn, and Chris

Lewis – I am greatly indebted to all of them. I am also thankful to Chris Sholley, Ashivni Shekhawat, and Cindy Chiao from whom I learned a great deal while interning at Lyft.

The last 5 years would not have been fun if it were not for the people and relationships that shaped them: Eoin, Mark, Sam, Flora, Jonathan, Molly, Steffen, Aditya, Mischa, and Rahmtin of the extended Lake St. family, Benny, Daniela, Laura, Netta, Maja, Manu, Chen, Anna, Simona, and Limor of my klike, Roni, my MYM teammates, and my JM buddies made these years enjoyable through ups and downs.

Finally, I want to thank my family for always supporting me: Sylvia and Friedhelm for always rooting for me; Neil, Phyl, Ari, and Gabe for your love, your cookies, and your editorial help; David, Genia, and Tommy for always being there for me; and most of all my parents for Pit and Mit, for providing me with endless opportunities, and for challenging me to always give my best.

This work was in part supported by NSF grants CCF-1526067, CMMI-1537394, CCF- 1522054, and CCF-1740822, and written during the Simons special semester on real-time decision making.

Many chapters in this thesis begin with quotes by Aaron Sorkin's characters. They should be attributed to him, but "Good writers borrow from other writers. Great writers steal from them outright." (S. Seaborn/A. Sorkin)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The number of commuting options available to the people of San Francisco has increased significantly in this decade. Aside from private vehicles, the options used to be restricted to BART, Muni, taxis, and cable/street cars (as well as some water transportation). Today, these have been complemented by ride-sharing (Uber/Lyft), dock-based bike-sharing (Ford GoBike), free-floating car sharing (Zipcar), station-based car sharing (Maven), electric scooter sharing with large (Scoot) or small (Bird, Spin, LimeBike) scooters, and electric bicycles (JUMP). We will refer to the collection of these services interchangeably as *shared vehicle* or *on-demand transportation systems*. Though San Francisco may be extraordinary with respect to the adoption rate of these systems, the increasing variety of transportation options is ubiquitous in major cities around the world.

A common denominator of on-demand transportation systems is the users' flexibility to enter and leave the system anywhere and anytime. This gives the systems several characteristics (cf. Table 1.1) that distinguish them from traditional mass transit. Thus, despite the study of mass transportation systems being among the oldest research areas within operations research (OR), the design and operation of shared vehicle systems require specialized models and algorithms. Before summarizing the contributions in this thesis, we now describe the distinctive features of shared vehicle systems and give an overview of the different operational levers present.

1.1 Distinctive Features of Shared Vehicle Systems

Schedules & Routes. Traditional public transportation systems can be characterized by customers entering and leaving vehicles (e.g., buses, trains) at fixed stations, that the vehicles visit in fixed order, at fixed times. The contrast to on-demand transportation systems, or bike-sharing as a special case thereof, is well-described through a quote by Jay Walder [2016], CEO of Motivate, the operator of several bike-sharing systems (BSS) across the world:

> [...] Bike sharing creates a system for personal mobility. It is personalized mass transit. You distance yourself from the idea of stations and routes and schedules.[...]

Walder's description points out the major difference between traditional public transportation and on-demand transportation systems: customers do not have to adhere to fixed schedules and routes. Instead, they engage vehicles at their preferred time and take their preferred route to their destination.

Stations. Beyond the routes and schedules, Walder also mentions the distancing from stations. Though the largest BSSs in the US are station-based (resp., dock-based), these stations are quite different from public transit stations in more than one way. First, given the flexibility with respect to schedules, stations are not meant to be waited at (though customers may have to wait for a stock-out to end, in order to rent/return a bike). Second, given the flexibility with respect to routes, stations really can be thought of as part of a *complete network*, in which any ride can go from any station to any other station. Third, setting up bike-sharing stations happens on a different time-scale than traditional public

transit: as an extreme example, contrast the 48 stations Citi Bike added to the Upper West and Upper East Sides of Manhattan during the summer of 2015 with the 3 new subway stations along the planned Second Avenue Subway that were constructed from 2011 to 2017.[1] More generally speaking, shared vehicle systems have expanded to new cities/service areas much faster than traditional public transit.

Entering and Leaving the System. Just like on-demand transportation, private vehicles also offer the flexibility for users to commute on their preferred schedule and route. However, as pointed out by Chicago bike advocate Julie Sherman [2017], there are two significant differences when it comes to the beginning and the end of a trip:

> I like the flexibility of being able to usually find and grab a bike as I make my way from point A to B, and then point F to G, without worrying that my own bike is sitting out vulnerable to the elements, vandalism, and theft.

Both differences pointed out by Sherman relate to the users' responsibility for the vehicle in shared vehicle systems: users enter and leave the system at the beginning and end of each trip. As such, they hold no responsibility for the vehicle after ending the trip. Also, they thereby gain the flexibility to begin trips at locations other than the end location of the most recent trip.

Data and Real-time Decision-making. Different transportation systems vary both with respect to the data they (can) collect and the extent to which data is used operationally. This is true even for traditional public transportation: in

[1] Remarkably, the subway stations had first been proposed in 1919!

San Francisco, the BART system collects data reflecting the origin-destination pairings of all trips, yet the Muni system does not. This is partially due to billing requirements: BART prices depend on the destination, Muni prices do not. The operational needs also explain the vast pools of data that shared vehicle systems collect:

- Dispatch decisions by ride-sharing systems are made in almost real time and require real-time data of which available drivers are nearby.
- Surge prices are set on a time-scale in the order of minutes and rely on an accurate reflection of the local and global balance between supply and demand.
- Routing decisions for rebalancing trucks in bike-sharing systems are made a few times per hour and are based on an accurate reflection of the current state of the system.

In contrast to the above real-time decisions, planning decisions like the following rely on somewhat coarser data:

- Non-motorized rebalancing decisions in bike-sharing (cf. Section 1.2.2) are made at most on a monthly time-scale.
- System design questions in bike-sharing systems, e.g., what stations to add docks to or where to set up stations, are made no more frequently than on an annual time-scale.

The operational decisions in public transportation systems, e.g., setting routes and schedules, more closely reflect the latter set of decisions. In contrast to the real-time decisions, these tend to not rely on pools of data as vast as those in

shared vehicle systems (cf. Section 1.4 for a summary of the kinds of data we have used in our collaboration with Motivate).

Two-sidedness. Most of the challenges and solutions described in this thesis deal with the need to balance demand and supply in on-demand transportation systems. In today's ride-sharing systems, this crucially requires ensuring that sufficiently many drivers are driving to serve customers with sufficiently low ETAs; in turn, having sufficiently many drivers relies on having enough riders requesting rides at sufficiently high prices (to make driving worthwhile the drivers' time). This is a fundamental difference compared to both traditional public transportation systems and to other shared vehicle systems in which the supply is set explicitly by the platform. As a result, pricing and other decision-making in these platforms must simultaneously consider the supply and demand sides. In this regard, challenges faced by ride-sharing systems are more similar to those in other two-sided markets, like Airbnb, Upwork, or Handy. However, other challenges in ride-sharing are first and foremost based on its nature as an on-demand transportation service. Throughout this thesis, the focus is on those challenges, rather than on the two-sided nature of these systems.

Table 1.1: Summary of Features Present in Different Commuting Options

	Time-scale of Decisions	Data available	Imbalance of Supply	Two-Sidedness	Flexibility of routes/schedules	Flexibility of entering/leaving
Ride-sharing (Uber, Lyft, etc.)	Real-time (Dispatch) to Weekly (Incentives)	Real-time	Alleviated by drivers' strategic behavior	Yes	Yes	Yes
Dock-based bike-sharing (Citi Bike, Hubway, etc.)	Minutes or Hours to Seasonal	Demand censoring due to stock-outs	Alleviated only by platform's rebalancing	No	Yes	Yes
Free-floating car-sharing (Car2Go, Zipcar)	Hours (Rebalancing) to Seasonal (Prices)	Demand censoring due to stock-outs	Alleviated only by platform's rebalancing	No	Yes	Yes
Free-floating bike-/scooter-sharing (JUMP, Bird, etc.)	Hours (Rebalancing) to Seasonal (Prices)	Demand censoring due to stock-outs	Alleviated only by platform's rebalancing	No	Yes	Yes
Traditional Public Transit (BART, Muni, Subway, etc.)	Seasonal (Schedules)	System-dependent	Alleviated by schedules	No	No	Yes
Personal Vehicles (Car, bikes, etc.)	N/A	N/A	N/A	No	Yes	No

1.2 Challenges of Imbalance

Users' flexibility in using on-demand transportation systems poses challenges for platform operators unbeknownst to traditional mass transportation modes. Fundamentally, these challenges are due to imbalance in the system. Though it is true across modes of transportation that travel demand is strongly asymmetric in each rush hour (cf. Figure 1.1 and Figure 1.2), the consequences for users vary across systems. Since buses and subways are scheduled to ride back and forth along their routes, asymmetric demand in these systems only implies that capacity in one direction is more utilized than in the other. Yet, capacity is still available in both directions. In contrast, in bike-sharing and other shared vehicle systems, the means of transportation themselves may not be available. This effect is displayed in Figure 1.3, which shows for several stations the fraction of docks filled with bikes over time; when the fraction of docks filled is 0 (or 1), i.e., when stations are empty (or full), customers suffer out-of-stock events, meaning that they cannot rent (return) bikes at (to) the station. This lack of availability has been cited (Capital Bikeshare [2014]) as one of the major reasons for dissatisfaction among users of bike-sharing systems. Similarly, ride-sharing systems compete on lower pick up times (Banerjee et al. [2018]), which also increase due to imbalance in the system. As such, alleviating the effects of imbalance is a major focus of platform operators. However, the tools available to operators vary, as we describe for different platforms below.

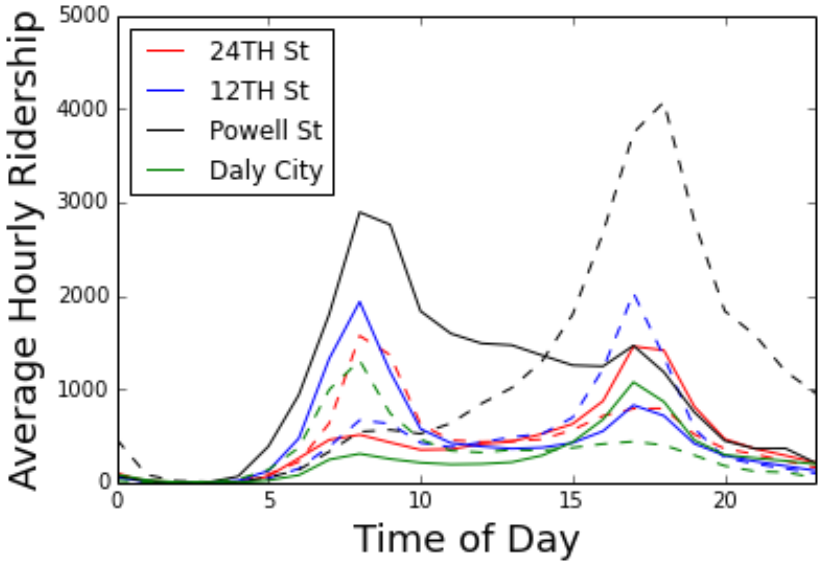


Figure 1.1: Asymmetric demand at 4 stations in the BART system. Solid lines denote arrivals whereas dashed lines denote departures.

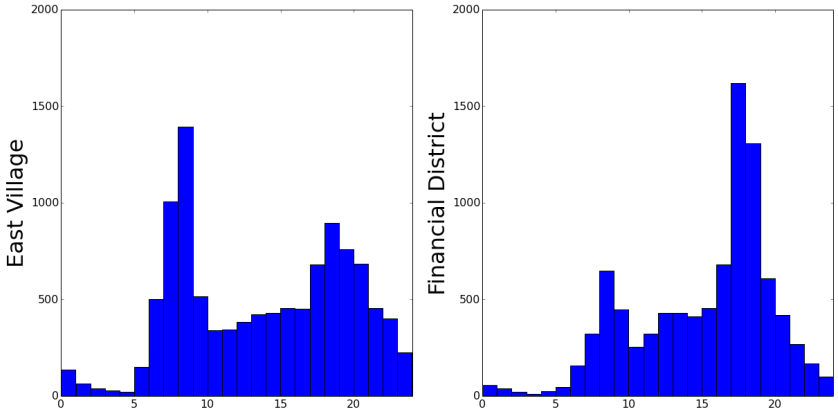


Figure 1.2: Number of Rentals originating at stations in the East Village and the Financial District during a week in July 2017, bucketed into one-hour intervals.

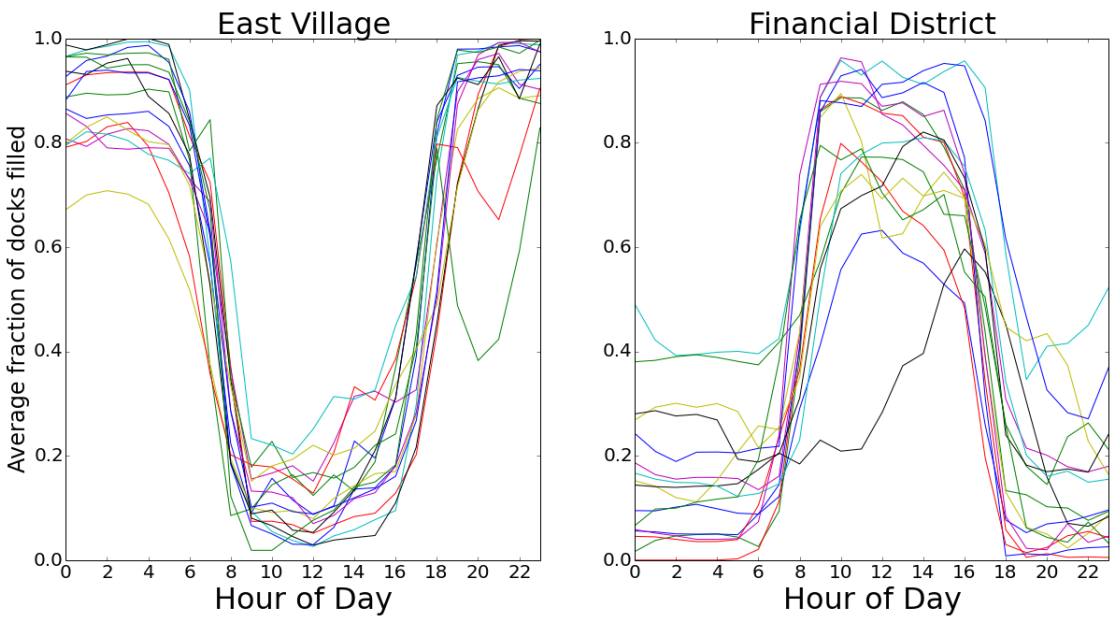


Figure 1.3: Average fraction of docks filled at stations over various times of the day.

1.2.1 Levers to reduce imbalance in ride-sharing

Fundamentally, ride-sharing companies need to match drivers to passengers who request rides and set prices for these rides. Most platforms, like Uber and Lyft, set static prices per mile/minute, which are then adjusted via a surge price that depends on real-time supply-demand imbalance. Dispatch, the decision which driver to match, is of course always dependent on current conditions. Though many trade-offs need to be considered in the context of dispatching (e.g., dispatching drivers from further away increases the odds of either the driver or the passenger canceling the ride), imbalance within the system can in principle be reduced by dispatching from high-supply to low-supply areas.

The effects of surge pricing on the balance between demand and supply happen on at least three different levels: first, in the very near-term, surge prices

decrease demand in areas where they are are charged. Second, over a slightly longer time-scale, they incentivize drivers to drive in areas more likely to have surge prices, which give higher earnings to those drivers; as such, they can increase supply locally over a slightly longer time-scale. Third, surge prices increase average-earnings for drivers and thereby, over a longer time-scale, increase the incentives for drivers to drive at all at a given time. Thus, surge prices allow platforms to modulate the demand-side in the short-term as well as increasing the local and global supply over longer time-scales. Beyond these effects, surge prices can also be applied to change the distribution of drivers within the system: by setting higher prices for rides to low-demand areas and lower prices for rides to high-demand areas, operators may ensure that more supply remains where it is most needed (cf. Section 9.3).

Though not (explicitly) applied by current ride-sharing platforms, it is also conceivable for ride-sharing platforms to reduce imbalance by paying drivers to relocate (cf. Section 10.1). In fact, when considering an autonomous system, in which the platform operates vehicles, this is an appropriate way to think about the cost induced by the relocation of vehicles.

1.2.2 Rebalancing in Bike-Sharing

In contrast to ride-sharing systems, bike-sharing systems have yet to start using pricing to modulate demand. This is partly due to the public-private partnerships via which these systems are set up and that enforce rules banning pricing from being used for demand regulation. Instead, operators have focused on the supply-side, having developed a range of different rebalancing solutions to

handle imbalance.

Most of the rebalancing in bike-sharing systems tends to be done via vans and box-trucks that can move between around 20 and 60 bikes at a time within the system – these are especially efficient when there is little congestion and many bikes can be picked up at once. Beyond box-trucks, platforms also operate so-called *trikes*: trailers pulled by a cyclist that can hold up to 18 bikes at a time. However, given the physical difficulty of pulling the trailer, trikes tend to be used over relatively short distances. Finally, some operators of dock-based systems select specific stations at which the capacity is artificially increased through so-called *corrals*. This is done by placing bikes in between the docks and thus using all available physical space at the stations. To ensure that the bikes do not get stolen, an employee looks after the bikes until, in the next rush hour when the demand goes in the opposite direction, the corrals empty out again.

1.2.3 Free-floating Shared Vehicle Systems

Free-floating car, scooter, or bike-sharing systems all have in common the fact that they do not apply, at the time of this writing, dynamic pricing to modulate demand. However, they do use rebalancing. In comparison to dock-based bike-sharing systems, the operational challenges are somewhat different though. Whereas a single employee with a box-truck can move up to 60 bikes within the system, in a car-sharing system, each vehicle moved requires a single employee. For free-floating bikes or scooters, it is not the limited capacity of vehicles moved per employee, but rather the dispersed locations of vehicles (rather than in a single station) that make it difficult to efficiently operate vans or trucks. Noticeably,

in systems with electric vehicles, an additional requirement for the operator is to ensure, that vehicles are charged, in addition to being rebalanced. At least two of the platforms operating these systems experiment with crowdsourcing to tackle both charging and rebalancing (LimeBike [2018], Bird [2018]).

1.3 Contributions

This thesis treats a range of optimization issues arising in the operation of shared vehicle systems. Each chapter contains the results of a different project. Most chapters include theoretical results and many involve the data analysis that motivated the theory developed. Broadly speaking, Part I of this thesis deals with dock-based bike-sharing systems, Part II with more general shared vehicle systems.

The main methodological difference between the two parts is due to different models of the system: the model in Part I treats the demand for rentals and returns at each station as exogenously given and time-varying. This gives rise to a tractable model for many different applications in practice, even though it sacrifices subtleties due to effects within the network. In particular, treating returns as exogeneously given implies that they need not be triggered by rentals elsewhere. Though this may seem restrictive, we find in Chapter 3 that it captures the most important characteristics of real data-sets. In contrast, in Part II, we analyse queueing-theoretic models that capture such effects, treating both rentals and returns as endogeneous. However, the model in Part II assumes that the system is in a time-invariant steady-state. In contrast to the application-driven focus of Part I, Part II aims to derive strong theoretical guarantees; these are

interesting from a stochastic control perspective, but also provide implementable insights. Yet, for practical considerations the algorithms may rely on too many parameters that would need to be estimated. Given the difference between the two parts with respect to both modeling and the purpose of the results, we separate the related work sections, providing one in each part.

Feasible Permutations. Most chapters in this thesis can be read independently of each other. That being said, most chapters in Part I assume knowledge of Chapter 3. Similarly, most results in Part II rely on Sections 9.2 and 9.3. Below, we summarize the results of each chapter.

1.3.1 Results in Part I

The results in Part I apply an inventory management model introduced in the groundbreaking work of Raviv and Kolka [2013]. The goal of the model is to estimate, over a finite time-horizon, the number of out-of-stock events at a station in a dock-based bike-sharing system as a function of the initial number of bikes. For a fixed sequence of arrivals (rentals and returns), it is easy to compute the updated number of bikes after each arrival as one fewer (rental) or one more (return) than before the arrival; exceptions occur when the station is empty or full, which is exactly the case in which a rental, respectively a return, experiences an out-of-stock event (cf. Chapter 3 for details). Raviv and Kolka [2013] developed this notion and extended it to exogeneously given stochastic arrival processes. This sparked a line of work (cf. Chapter 2) that applies the inventory model in routing models that aim to guide the dispatching of rebalancing trucks in bike-share systems. Throughout Part I, we develop various use cases of variations

of the model beyond the application in routing. We conclude the first part in Chapter 8 with a discussion of real-world implementations of our work.

Capacity Allocation. In Chapter 4 we apply the inventory model to study the sizing of stations within a bike-share system; that is, given the exogeneous arrivals, we ask how many docks should be placed at each location within the system. We model this question as an optimization problem, derive discrete convex properties for it, and use those to develop a discrete gradient-descent algorithm with running time linear in the number of docks in the system. Next, we prove that the algorithm is amenable to scaling techniques to guarantee convergence in time logarithmic in the number of docks. Using rich data-sets from NYC, Boston, and Chicago, we derive suggestions on where within the system docks should be added and taken away. Finally, based on changes to the system implemented by Citi Bike in NYC, we develop a counterfactual analysis to evaluate the improvement in the system due to the docks moved.

Incentive Design. In Chapter 5 we consider a data-set from Citi Bike's *Bike Angel* program. The program incentivizes users to return bikes to stations where bikes are scarce and to rent bikes where empty docks are needed. When originally set up, the program was based on fixed stations in each rush hour at which rentals, respectively returns, were incentivized. We apply the user dissatisfaction functions to study the tradeoffs between this static set of stations and an incentive scheme that dynamically updates the set of incentivized stations based on real-time data. We thereby characterize the improvements possible by switching to a dynamic scheme.

Rebalancing. Chapter 6 contains three different applications of Raviv and Kolka's inventory model. We begin by describing the real-world implementa-

tion of an integer-programing based decision aid to guide trucks for overnight rebalancing. Though the integer program itself is similar to what is known in the literature, to the best of our knowledge, ours was the first attempt to operationalize this methodology in practice. Thereafter, we consider two kinds of rebalancing that are not truck-based: trikes and corrals. For trailers, we show that, under appropriate assumptions, the user dissatisfaction functions at two separate stations can be coupled via the trailer; this gives rise to a maximum-matching formulation to identify the best pairs of fixed stations between which to route trailers. For corrals, we design a maximum coverage integer programming formulation to find the optimal placement of corrals within the system.

Maintenance Scheduling. Beyond rebalancing, service quality in bike-sharing systems is also affected by maintenance decisions. Specifically, the docks in stations sometimes break. In such cases, the capacity of the station is effectively reduced by the number of defective docks. Similarly, each returned bike that is labelled as defective reduces the effective capacity of the station. While bikes are designated as broken when they are returned, defective docks need to be inferred from data (cf. Figure 1.4). Combining the inference and the inventory model, the operator can estimate (at each station) the impact of defective docks on user experience; this in turn may inform the prioritization of locations in a maintenance schedule. In Chapter 7 we model this problem as a budgeted (by the length of a shift) prize-collecting traveling salesman problem where the prize corresponds to the reduction in out-of-stock events. We design a polynomial-time primal-dual algorithm for this problem with an approximation guarantee of 2, improving upon the previously best-known guarantee of $2 + \epsilon$ in time $O(n^{\frac{1}{\epsilon}})$. As a corollary of our analysis, we also obtain an approximation guarantee of 2 for the budgeted prize-collecting minimum spanning tree problem, improving

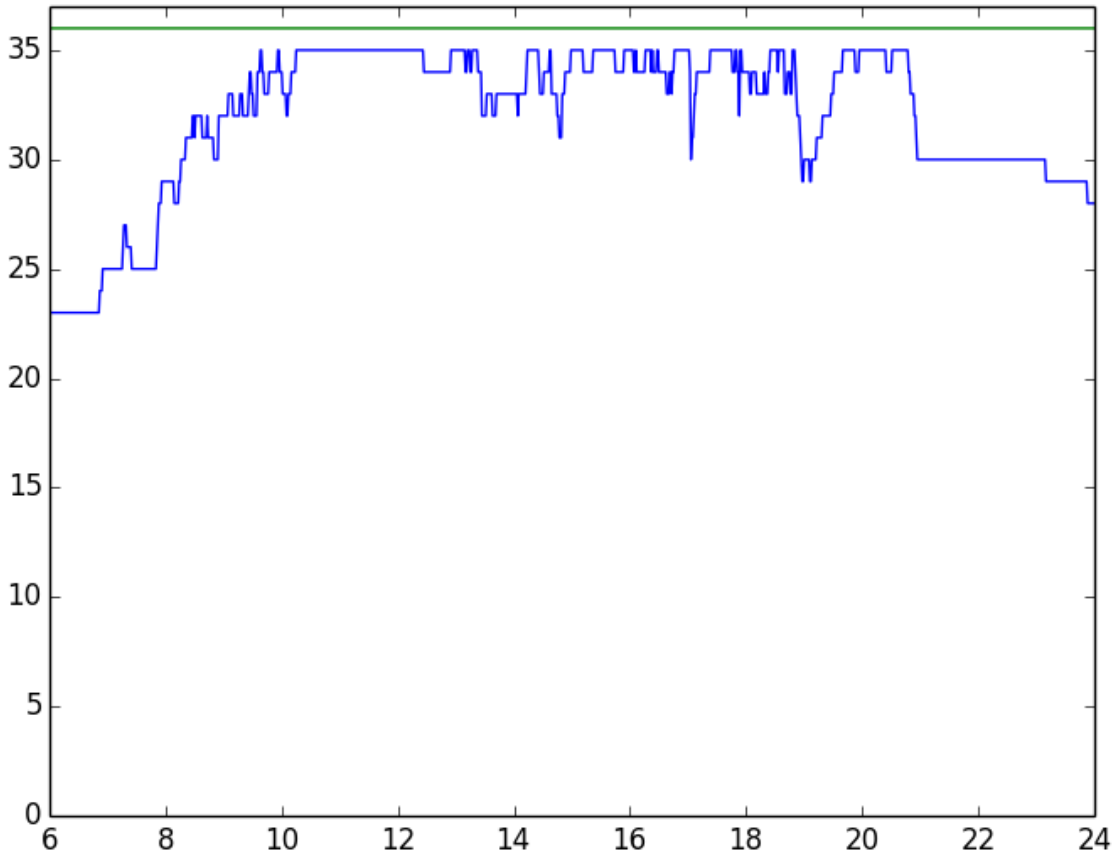


Figure 1.4: Number of bikes (in blue) and capacity (in green) of a particular bike-share station over the course of one day from 6AM to 12AM. Between 10AM and 8PM, a pattern persists wherein the number of bikes continuously fluctuates between one below capacity and at least two below capacity, but it never reaches capacity. This strongly indicates that the last dock at the station was defective at the time.

upon the previously best-known guarantee of 3.

1.3.2 Results in Part II

Throughout Part II we focus on variations of a simple closed queueing network model of a shared vehicle system. In the simplest version of the model, we are

given n locations with pairwise (Poisson) demand-rates and m vehicles. Customers appear to travel from an origin location to a destination; each customer has a value drawn from a known distribution. If no vehicles are present at the origin, the customer disappears; else, we offer the customer a price. If the price is higher than the customer's value, then the customer declines and disappears. Else, the customer accepts the price and instantaneously travels to the destination, i.e., the number of vehicles at the origin decreases by one and increases by one at the destination. In that case, the platform receives a reward. Our goal is to maximize the steady-state rate at which the platform collects the rewards. In this simplest version, rewards are independent of the price offered, i.e., the goal of the platform is to maximize ridership (throughput) and the dynamics are completely characterized by the customers' trips. We formally define this model in Section 9.2.

Vanilla Case. Though the queueing dynamics are easily explained, maximizing steady-state rewards is a non-concave maximization problem. In Section 9.3 we provide an algorithm with strong provable guarantees for regimes of interest; in particular, we show that it is possible to obtain prices that are independent of the current configuration of vehicles, yet have a parametric guarantee that linearly converges to optimality as the number of vehicles grows. Thereafter, we extend our analysis far beyond the simple setting described above.

Objectives. In Section 9.3.3 we first extend the analysis to hold for objectives like social welfare and revenue. Then, we show that our approach naturally applies to multiobjective settings in which one objective (e.g., social welfare) ought to be maximized subject to a lower bound on another (e.g., revenue). For these settings we obtain bicriteria approximation algorithms with the same

guarantees.

Controls. In Section 10.1 we extend the results to controls other than prices. Specifically, we study a dispatch mechanism and a rebalancing mechanism. The former models the decision that a ridesharing platform makes when a ride request is made and the platform decides which driver to dispatch for serving the request. The latter corresponds to, depending on the exact system modeled, incentives given to drivers to relocate, the control of autonomous vehicles, or (in car-sharing or scooter-sharing) an explicit effort to rebalance to different locations upon the end of a trip. We show that in some ways the two controls can actually be viewed as mathematically equivalent and prove that the same guarantees obtained for the basic setting hold, in fact, for any combination of the controls.

Travel-times. In Section 10.2 we extend the model to capture travel-times inbetween stations. Though we prove that our analysis yields asymptotic optimality for this case as well, we also show that in heavy traffic, i.e., when we obtain a solution in which induced demand exactly matches the available supply, one cannot hope to attain the same rate of convergence obtained for the case without travel times: even an optimal policy cannot achieve a rate of convergence faster than the square-root of the number of vehicles. In a slightly lighter regime, however, a linear rate of convergence to optimality is still achievable!

1.4 Impact on Industry

The availability of data and the complexity of the systems make shared vehicle systems an attractive application for operations research. Though this is recognized by the many papers published over the last five years (cf. Sections 2 and

9.1), the impact on practice seems limited. In fact, a study by de Chardon et al. [2016] found that in the space of bike-sharing, operators did not use optimization to support their decision-making. Even worse, the study states that

> [..] New York City is the sole operator we know using custom software forecasting station demand and trip flows [..].

Many of the contributions in Part I overcome this limitation, as they have had impact on the bike-sharing systems operated by Motivate International (incl. New York City, Boston, San Francisco, and others). We summarize these industry implementations at the end of Part I, in Chapter 8.

On availability of data. Most of the data we relied on is publicly available. The demand-rates used to compute the inventory model in Part I are based on historic ridership-data and a live-feed of the number of bikes at each station.[2] The observed real impact in Section 6.4 also rely on the latter. The data underlying the pilots conducted for overnight rebalancing (cf. Section 6.2) as well as the data underlying the *Bike Angels* analysis (cf. Section 5) are proprietary to NYCBS.

[2] `https://www.citibikenyc.com/system-data`

Part I

Inventory Models in Bike-Sharing Systems

CHAPTER 2

RELATED WORK

> "We're meant to keep doing better. We're meant to keep discussing and debating and we're meant to read books by great historical scholars and then talk about them" — J. Breckenridge

Over the past decade bike-sharing has become a prominent research topic within areas such as data mining, machine learning, and optimization. In this chapter, we provide an extensive literature review of this research area. We begin by summarizing the literature on routing for rebalancing, then discuss existing work on forecasting in bike-sharing systems, and finally describe related work on the design of bike-sharing systems.

2.1 Routing

Much of the research on bike-sharing systems has focused on optimizing the routes of rebalancing trucks employed by system operators. A particularly influential paper in this context is Raviv and Kolka [2013] who define a user dissatisfaction function to measure the number of out-of-stock events at an individual station as a function of the number of bikes at the station. Different ways of computing this cost function have been suggested by Schuijbroek et al. [2017], O'Mahony et al. [2016], and Parikh and Ukkusuri [2014]. Subsequent work by Raviv et al. [2013] defined a routing problem based on the user dissatisfaction function: at first, a time bound is given in which trucks can be routed to move bikes within the system; thereafter, no more decisions are made, and the objective

is given by the expected number of out-of-stock events, given the configuration of bikes resulting from the trucks' rebalancing. Such routing problems, and attempts to solve them to optimality for larger and larger instances, were further investigated by Forma et al. [2015], Ho and Szeto [2014], and Szeto et al. [2016], among others. Parts of our work in Section 6 are very much related to those papers. Similarly, a line of work, starting with Rainer-Harbach et al. [2013] and followed by Raidl et al. [2013] and Kloimüllner et al. [2014] investigated greedy strategies for the rebalancing problem, though they considered a slight variation (*i.e.*, a fluid version) of the user dissatisfaction function. The work by Kloimüllner et al. [2014] stands out in that regard in that it also applies to the dynamic case, in which unsatisfied demand also occurs during the rebalancing process. An orthogonal approach to rebalancing has been taken by Shu et al. [2013], O'Mahony et al. [2016], and Jian and Henderson [2015]; all of these papers aim to find the optimal configuration of bikes at the beginning of some period. Shu et al. [2013] assume complete knowledge of the future and solve a flow problem; O'Mahony et al. [2016] employs the user dissatisfaction function; Jian and Henderson [2015] use a simulation-optimization based approach to capture network effects. In these three versions, limited means for rebalancing (and thus, the routing aspect of the problem) are disregarded since the focus is solely on the optimal allocation of bikes. Contardo et al. [2012], Vogel et al. [2014], and Nair et al. [2013] are similar to Shu et al. [2013] in that they solve particular flow problems rather than routing problems. Nair et al. [2013] aims to obtain certain service levels with at least some probability. Vogel et al. [2014] presents an NP-hard flow model that also takes into consideration a rebalancing cost. All of these assume that not only the rate of rentals and returns at each station is known, but also the routing probability of each customer, *i.e.*, the probability

of a customer at a given station having a particular destination. An approach similar to that of Jian and Henderson [2015] was pursued by Datner et al. [2017], in which they also account for the cost of longer travel times due to out-of-stock events rather than minimizing only the number of out-of-stock events.

A disjoint line of work has focused on minimizing the length of the route of a single capacitated truck, or the combined length of routes for a fleet of such trucks, that needs to visit nodes with given demand and supply. The paper by Benchimol et al. [2011] is an early example of such work. They give an approximation algorithm, a hardness result, and a polynomial-time algorithm for instances, wherein the underlying graph is a tree. The same problem has been studied by Chemla et al. [2013] and Dell'Amico et al. [2014] from a mixed-integer programming perspective. Further works in the same spirit have been pursued by Erdoğan et al. [2014], Erdoğan et al. [2015], and Bulhões et al. [2018]. Interestingly, Di Gaspero et al. [2013], in a sense, combines the approaches of maximizing impact and minimizing travel time: given fixed targets for each station, the authors aim to minimize a weighted combination of travel time and absolute value distance (summed over all stations) between the targeted bike allocation and the one resulting from rebalancing.

Some recent papers have taken different approaches based on robust optimization. Ghosh et al. [2016] studies a repositioning appraoch based on an iterative two-player game, in which the environment generates a demand scenario out of feasible demand scenarios; they apply this approach to small systems with 20 stations. They also develop a simulation model, which Lowalekar et al. [2017] uses to demonstrate the benefit of multi-stage stochastic optimization. Ghosh et al. [2017] makes explicit the distinction between *routing* and *repositioning* with

the former being about minimizing travel time and the latter being about finding the best obtainable allocation.

In contrast to the work outlined on rebalancing with trucks, O'Mahony and Shmoys [2015] also investigates the use of trailers in bikesharing systems; later work by Freund et al. [2016] (cf. Section 6) also considers so-called corrals.

2.2 Forecasting

Separate from the literature on rebalancing, there is also a long line of literature related to forecasting. Most of the forecasting relates to prediction of demand based on historical data; examples include Li et al. [2015], Rudloff and Lackner [2014], Salaken et al. [2015], O'Mahony and Shmoys [2015], and Riquelme et al. [2017]. Several other forecasting questions have been studied as well: Kaspi et al. [2016] tries to detect which bikes in a system are broken given the usage data at each station, a question relevant for routing problems such as the budgeted prize-collecting traveling salesman problem studied by Paul et al. [2017] (cf. Section 7); Hsu et al. [2016] uses a discrete choice model to study the behavior of users when faced with out-of-stock events; Zhang et al. [2016] predicts the destination and destination time of customers given the origin, the time and personal information about the user (gender/age); an approach to predicting pairwise demand, rather than incoming/outgoing demand at individual stations, can be found in Singhvi et al. [2015]; Chen et al. [2016] dynamically clusters stations to predict which stations will run out of available bikes/docks.

2.3 System Design

Finally, there is a line of work on the design of such systems. Kabra et al. [2015] applies techniques from econometrics to study the density with which stations should be placed. O'Mahony et al. [2016] defines an integer program to investigate what allocation of docks, given a budget of docks, to existing stations minimizes out-of-stock events (using the local user dissatisfaction function at each station); Freund et al. [2017] (cf. Section 4) extends this question in various ways and provides an efficient algorithm to solve it. Jian et al. [2016], using simulation optimization in the same manner as Jian and Henderson [2015], aims to find the optimal allocation of docks, though they allow for network effects that cause non-convexities. All of these papers are based not on rebalancing but on the question of what the result of optimal rebalancing would look like. A similar approach is used by Saltzman and Bradford [2016], which investigate the augmentation problem, that is, the problem of (optimally) adding docks to existing stations. While Saltzman and Bradford [2016] uses simulation, this question can also be approached using the methodology described in Chapter 4.

CHAPTER 3

USER DISSATISFACTION FUNCTION

Raviv and Kolka [2013] define a *user dissatisfaction function* (UDF) that uses demand information to map the number of bikes at each station at the beginning of a time interval to the expected number of customers over the course of the interval that will not be able to rent/return because of the station being empty/full. We begin this chapter by formally defining these UDFs.

3.1 Definition

We denote a sequence of s customers at a bike-share station by $X = (X_1, \ldots, X_s) \in \{\pm 1\}^s$. The sign of X_t identifies whether customer t arrives to rent or to return a bike, i.e., if $X_t = 1$, then customer t wants to return a bike and if $X_t = -1$, then customer t wants to rent a bike. The truncated sequence $(X_1, \ldots, X_t)$ is written as $X(t)$. We denote throughout by d and b the number of open docks and available bikes at a station before any customer has arrived. Notice that a station with d open docks and b available bikes has $d + b$ docks in total. Whenever a customer arrives to return a bike at a station and there is an open dock, the customer returns the bike, the number of available bikes increases by 1, and the number of open docks decreases by 1. Similarly, a customer arriving to rent a bike when one is available decreases the number of available bikes by 1 and increases the number of open docks by 1. If instead a customer arrives to rent (return) a bike when no bike (open dock) is available, then she disappears with an out-of-stock event. We assume that only customers affect the inventory-level at a station, i.e.,

no rebalancing occurs. It is useful then to write

$$\delta_{X(t)}(d,b) := \max\{0, \min\{d+b, \delta_{X(t-1)} - X_t\}\};\ \delta_{X(0)}(d,b) = d;$$
$$\beta_{X(t)}(d,b) := \max\{0, \min\{d+b, \beta_{X(t-1)} + X_t\}\};\ \beta_{X(0)}(d,b) = b;$$

as a shorthand for the number of open docks and available bikes after the first t customers arrive.

Our objective is based on the number of out-of-stock events. In accordance with the above-described model, customer t experiences an out-of-stock event if and only if $\delta_{X(t)}(d,b) = \delta_{X(t-1)}(d,b)$, that is, out-of-stock events occur if and only if an arriving customer does not change the number of bikes at the station. Since $d + b = \delta_{X(t)}(d,b) + \beta_{X(t)}(d,b)$ for every t, this happens if and only if $\beta_{X(t)}(d,b) = \beta_{X(t-1)}(d,b)$. Since we are interested in the number of out-of-stock events as a function of the initial number of open docks and available bikes, we can write the objective as

$$c^X(d,b) = |\{\tau : X_\tau = 1, \delta_{X(\tau-1)}(d,b) = 0\}| + |\{\tau : X_\tau = -1, \beta_{X(\tau-1)}(d,b) = 0\}|.$$

It is then easy to see that with $c^{X(0)}(d,b) = 0$, $c^{X(t)}(d,b)$ fulfills the recursion

$$c^{X(t)}(d,b) = c^{X(t-1)}(d,b) + \mathbf{1}_{\{\beta_{X(t)}(d,b) = \beta_{X(t-1)}(d,b)\}}.$$

Given, for each station $i \in [n]$, a distribution p_i over possible sequences of arrivals $\{(\pm 1)^s, s \in \mathbb{N}_0\}$, which we call *demand-profile*, we can write $c_i(d,b) = \mathbb{E}_{X \sim p_i}[c^X(d,b)]$ for the expected number of out-of-stock events at station i and $c(\vec{d}, \vec{b}) = \sum_i c_i(d_i, b_i)$.

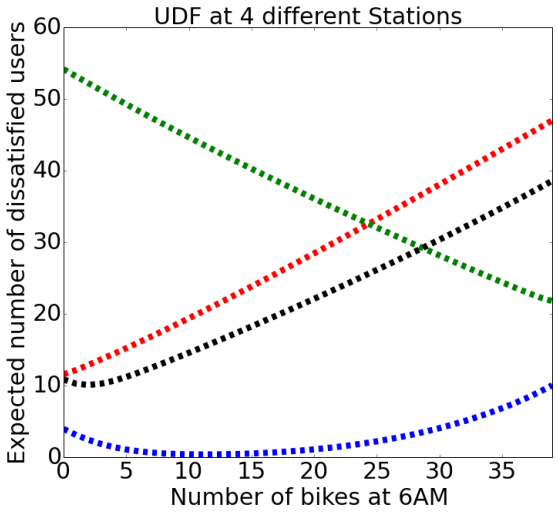


Figure 3.1: As a function of bikes for stations with capacity 39.

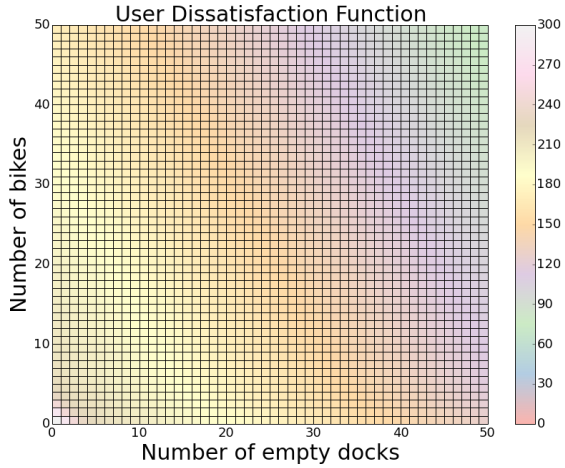


Figure 3.2: As a function of (d, b) at a single station.

Figure 3.3: Visualizations of user dissatisfaction functions based on real data.

3.2 Discussion of Assumptions

The formulation of the UDFs is based on several key assumptions. Before delving into the various applications of the UDFs, we discuss here these assumptions as well as the advantages that come along with them.

Seasonality and Frequency of Reallocations.

In contrast to bike rebalancing, the reallocation of docks (cf. Chapter 4) is a strategic question that involves docks being moved at most on an annual level. As such, one concern would be that the recommendations for a particular month do not yield improvement for other times of the year. Of course, one way of dealing with this would be to explicitly distinguish, in the demand profiles, between different seasons, i.e., have k different distributions for k different types of days and then consider the expectation over these as the objective. Though the user dissatisfaction functions accommodate that approach, we find on real data (cf. Section 4.4.3) that this is not actually necessary: the reallocations that yield greatest impact for the summer months of one year also perform very well for the winter months of another. This even held true in New York City, where the system significantly expanded year-over-year: despite the number of stations in the system more than doubling and total ridership increasing by around 70% from 2015 to 2017, we find that the estimated improvement due to reallocated docks is surprisingly stable across these different months.

Cost of Reallocations.

Throughout Chapter 4 we consider reallocations of docks as being bounded by the number of docks that are being moved instead of associating it with an explicit cost. Mathematically, this is equivalent to handling a fixed per-unit cost for each dock reallocated. This is motivated mostly by the real-life operations of our industry partner: the cost of physically reallocating capacity from one location to another is negligible when compared to the administrative effort and political implications associated with reallocating capacity. In particular, this

implies that the tactical question of how to carry out the reallocations is of minor importance in practice. Further, the cost of reallocating docks can be compared to the cost of rebalancing bikes: while the (one-off) reallocation of a single dock is about an order of magnitude more expensive than that of a single bike, the reallocated dock has daily impact on improved service levels (in contrast to the one-off impact of a rebalanced bike); thus, the cost amortizes extremely fast (Motivate estimates in as little as 2 weeks). Finally, the cost to acquire a new dock is 40-80 times higher than that of reallocating a dock. Thus, we focus our analysis only on reallocating capacity, even though we show in Appendix 12.3 that the algorithm also extends to capture the tradeoff between installing newly bought and reallocating existing docks.

Bike Rebalancing.

The optimization problem in Chapter 4 assumes that the bikes are optimally rebalanced initially, yet the user dissatisfaction functions assume that no rebalancing takes place over the course of the time horizon in which out-of-stock events are measured. The former is relaxed in Section 4.4.1 in which we consider a regime that assumes that no rebalancing occurs at all. The latter is motivated by the fact that rebalancing is quite concentrated (cf. Figure 3.4): in fact, in New York City more than 60% of all rebalancing is concentrated at just 28 of 762 stations. Given this concentration, our justification distinguishes between stations where very little rebalancing is done and stations where a lot of rebalancing is done. At the former kind of station, we assume we may discount rebalancing entirely. At the latter, having an allocation of bikes and docks with a lower user dissatisfaction function value leads to a lesser need for rebalancing; either way,

the user dissatisfaction function captures the operator's priorities without even taking into account rebalancing. Nevertheless, in Section 4.5, when analyzing the impact of reallocated capacity, we do explicitly consider the effect of rebalancing.

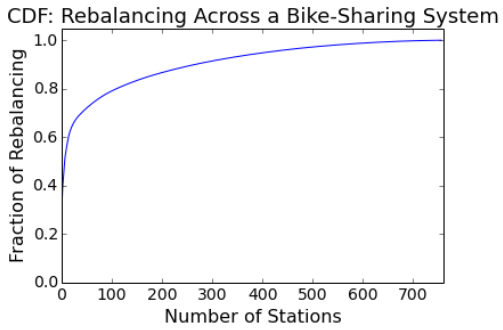


Figure 3.4: Fraction of rebalancing actions (bikes being added or taken) at stations within Citi Bike's system. Most of it happens at a small fraction of stations.

Out-of-stock Events and Demand Profiles.

In practice, we cannot observe attempted rentals at empty stations nor can we observe attempted returns at full stations. Worse still, given that most bike-sharing systems have mobile apps that allow customers to see real-time information about the current number of bikes and empty docks at each station, there might be customers who want to rent a bike at a station, see on the app that the station has only one bike available presently, and decide against going to the station out of concern that by the time they get there, the bike has already been taken by someone else — should such a case be considered an out-of-stock event (respectively, an attempted rental)? The user dissatisfaction functions assume that such events do not occur as the definition relies on out-of-stock events occurring only when stations are either entirely empty or entirely full.

Further, in order to compute the user dissatisfaction functions, we need to be

able to estimate the demand profiles: using only observed rentals and returns is insufficient for this as it ignores latent demand at empty/full stations. To get around this, we mostly adapt a combination of approaches by O'Mahony and Shmoys [2015], O'Mahony et al. [2016], and Parikh and Ukkusuri [2014]: we estimate Poisson arrival rates (independently for rentals and returns) for each 30 minute interval and use a formula developed by O'Mahony et al. [2016] to compute, for any initial condition (in number of bikes and empty docks) the expected number of out-of-stock events over the course of the interval. We plug these into a stochastic recursion suggested by Parikh and Ukkusuri [2014] to obtain the expected number of out-of-stock events over the course of a day as a function of the number of bikes and empty docks at 6AM (in Chapter 5, we apply the same method for different starting times). This is far from being the only approach to compute user dissatisfaction functions; for example, in Section 4.5 we explicitly combine empirically observed arrivals with estimated rates for times when rentals/returns are censored due to stations being empty/full.

Exogeneous Rentals and Returns.

The demand profiles assume that the sequences of arrivals are exogeneous, i.e., there is a fixed distribution that defines the sequence of rentals and returns. Before justifying this assumption, it is worth considering a setting in which it fails spectacularly: consider an allocation of bikes and docks that allocates no bikes at all. Of course, this would imply that no attempted rental is ever successful and therefore no returns ever occur. As such, the sequence of arrivals of returns at one station are not independent of the allocations elsewhere.

The main justification for this assumption comes from orthogonal work by

Jian et al. [2016]: this paper used a simulation optimization approach to find the configuration of bikes and docks across the system that minimizes the number of out-of-stock events over the course of the day. In contrast to the user dissatisfaction functions, decensoring the demand data for the simulation required additional modeling decisions. However, the simulation allowed for two different kinds of endogeneity:

- A return at one station had to be triggered by rentals at another. In particular, each (successful) rental caused a later attempt for a bike to be returned.
- A failed return at a station (due to out-of-stock events, i.e., stations being full) triggered a later attempt to return a bike at a station nearby.

While this still assumed that demand for rentals is exogeneous, it endogenized returns, excluding (at least) the example suggested above. However, it causes the resulting simulation optimization problem to be non-convex in an unbounded fashion; that is, one can construct examples in which two initial conditions are only *two bikes away from each other* (meaning they differ only by one station having two bikes fewer, the other two more); yet, one of them has arbitrarily many out-of-stock events fewer (in expectation), than the other. Even worse, both solutions still have strictly better objectives than the solution in between (in which one bike is moved). This level of sensitivity not only makes it harder to optimize, but also makes solutions difficult to interpret. Jian et al. [2016] proposed a range of different gradient-descent algorithms as heuristics to find good solutions, including an adaptation of the algorithm we present and analyze in Section 4.2. Despite the simulation adding key complexities to the system, the heuristics gave only limited improvements of approximately 3% (to the expected total number of out-of-stock events) when given the solution found by the algorithm

in Section 4.2 as a starting point. We discuss additional advantages of the user dissatisfaction functions below.

Advantages of User Dissatisfaction Functions.

The user dissatisfaction functions yield several advantages over a more complicated model such as the simulation. First, they provide a computable metric that can be used for several different operations: in Section 4.2 we show how to optimize over them for reallocated capacity and in Section 4.5 we use them to evaluate the improvement from already reallocated capacity. In Chapter 5 we use them to study an incentive program operated by Citi Bike in New York City, and they have been used extensively for motorized rebalancing (cf. Chapters 2 and 6). As such, the user dissatisfaction functions provide a single metric on which to evaluate different operational efforts to improve service quality, which adds value in itself. Second, for the particular example of reallocating dock capacity that we study in Chapter 4, they yield a tractable optimization problem which we prove in Section 4.2. Third, for the reallocation of dock capacity, the discrete convexity properties we prove imply that a partial implementation of the changes suggested by the optimization (cf. Section 4.4) is still guaranteed to yield improvement. Finally, given a solution to the optimization problem, it is easy to track the partial contribution to the objective from changed capacity at each station, making solutions interpretable.

CHAPTER 4

ALLOCATION OF DOCK CAPACITY

"The first step in solving any problem is recognizing there is one."

— W. McAvoy

While similar in spirit to the literature on rebalancing (for which the UDF was developed), in this chapter we use a different control to increase customer satisfaction. Specifically, we answer the question *how should bike-sharing systems allocate dock capacity to stations within the system so as to minimize the number of dissatisfied customers*? To answer this question, we consider two optimization models, both based on the underlying metric that system performance is captured by the expected number of customers that do not receive service. In the first model, we focus on planning for one day, say 6AM-midnight, where for each station we determine its allocation of bikes and docks; this framework assumes that there is sufficient rebalancing capacity to restore the desired bike allocation by 6am the next morning. Since in practice this turns out to be quite difficult, the second model considers a set-up induced by a long-run average that assumes that no rebalancing is done overnight. Through extensive computational experiments on real data-sets we found that there are dock allocations that simultaneously perform well with respect to both models, yielding huge improvements to both (in comparison to the current allocation). This section is based on a paper by Freund, Henderson, and Shmoys 2017.

Contribution

Our goal is to find an allocation of bikes and docks in the system that minimizes the total expected number of out-of-stock events within a system of n stations, i.e., $\sum_{i=1}^{n} c_i(d_i, b_i)$. Here, we assume that we are given the UDFs $c_i(\cdot, \cdot)\ \forall i$ (cf. Section 3) to measure the expected number of out-of-stock events at an individual bike-share station over the course of some time interval as a function of the initial number of bikes and the capacity at the station. Since the number of bikes and docks is limited, we need to accommodate a *budget constraint* B on the number of bikes in the system and another on the number of docks $D + B$ in the system. Other constraints are often important, such as lower and upper bounds on the allocation for a particular station; furthermore, through our collaboration with Citi Bike in NYC, it also became apparent that *operational constraints* limit the number of docks moved from the current system configuration. Thus, we aim to minimize the objective among solutions that require at most some number of docks moved. Via standard dynamic programming approaches, our methods also generalize to other practically motivated constraints, such as lower bounds on the allocation within particular neighborhoods (e.g., in Brooklyn).

We first design a discrete gradient-descent algorithm that provably solves the minimization problem with $O(n + B + D)$ oracle calls to evaluate cost functions and an overhead of $O((n + B + D)\log(n))$ elementary list operations (which, in practice, is negligible). Using scaling techniques and a subtle extension of our gradient-descent analysis, we improve the bound on oracle calls to $O(n\log(B+D))$, which still dominate an $O(\log(B + D)(n\log(n))$ term for elementary list operations.

The primary motivation of this analysis is to investigate whether the number of out-of-stock events in bike-sharing systems can be significantly reduced by a

data-driven approach. In Section 4.4, we apply the algorithms to data-sets from Boston, NYC, and Chicago to evaluate the impact on out-of-stock events. One shortcoming of that optimization problem is its assumption that we can perfectly restore the system overnight to the desired initial bike allocation. Through our ongoing collaboration with the operators of systems across the country, it has become evident that current rebalancing efforts overnight are vastly insufficient to realize such an optimal (or even near-optimal) allocation of bikes for the current allocation of docks. Thus, we consider in Section 4.4.1 the opposite regime, in which no overnight rebalancing occurs at all. To model this, we define an extension of the UDF under a long-run average regime. In this regime, the assumed allocation of bikes at each station is a function of only the number of docks and the estimated demand at that station. Interestingly, our empirical results reveal that operators of bike-sharing systems can *have their cake and eat it too*: optimizing dock allocations for one of the objectives (optimally rebalanced or long-run average) yields most of the obtainable improvement for the other.

Changes implemented by operators based on our recommendations, allow us to evaluate the impact of our analysis. In Section 4.5 we prove that observing rentals and returns after capacity has been added provides a natural way to estimate the reduction in out-of-stock events (due to dock capacity added) that can be computed in a very simple manner. Applying that approach to a small set of stations with added capacity in New York City, we derive estimates for the impact that changes to the system design have had. We discuss the computational efficiency of our algorithms in Section 4.6.

Relation to Multimodularity

Our algorithms and analyses crucially exploit that the user dissatisfaction functions $c_i(\cdot,\cdot)$ at each station are multimodular (cf. Definition 1). This provides an interesting connection to the literature on discrete convex analysis. In concurrent work by Kaspi et al. [2017] it was shown that the number of out-of-stock events $F(b, U-d-b)$ at a bike-share station with fixed capacity U, b bikes, and $U-d-b$ unusable bikes is M-natural convex in b and $U-d-b$ (see the book by Murota [2003] and the references therein). Unusable bikes effectively reduce the capacity at the station, since they are assumed to remain in the station over the entire time horizon. A station with capacity U, b bikes, and $U-b-d$ unusable bikes, must then have d empty docks; hence, $c(d,b) = F(b, U-d-b)$ for $d+b \leq U$, which parallels our result that $c(\cdot,\cdot)$ is multimodular. Though this would suggest that algorithms to minimize M-convex functions could solve our problem optimally, one can show that M-convexity is not preserved, even in the version with only budget constraints.[1] In fact, when including the operational constraints even discrete midpoint convexity, a strict generalization of multimodularity (Moriguchi et al. [2017]) which is in turn much weaker than M-natural convexity, breaks down.[2] Surprisingly, we are nevertheless able to design fast algorithms; these exploit not only the multimodularity of each individual c_i, but also the separability of the objective function (w.r.t. the stations); that is, the fact that each c_i is a function of only d_i and b_i. This not only extends ideas from the realm of unconstrained discrete convex minimization to the constrained setting, but also yields algorithms that (for our special case) have significantly faster running

[1] In Appendix 12.1 we provide an example in which a M-convex function restricted to an M-convex set is not M-convex; the example also shows that Murota's algorithm for M-convex function minimization can be suboptimal in our setting.

[2] In Appendix 12.2 we show that this holds true even in the simplest cases.

times than those that would usually arise in the context of multimodular function minization.

4.1 Notation

Recall from Section 3 that the user dissatisfaction function takes, for each station $i \in [n]$, a distribution, which we call *demand-profile* p_i over $\{(\pm 1)^s, s \in \mathbb{N}_0\}$, to map the number of docks and bikes to the expected number of out-of-stock events at a station, i.e., we can write $c_i(d, b) = \mathbb{E}_{X \sim p_i}[c^X(d, b)]$ for the expected number of out-of-stock events at station i and $c(\vec{d}, \vec{b}) = \sum_i c_i(d_i, b_i)$. In this section, we want to solve, given budgets B on the number of bikes and $D + B$ on the number of docks, a current allocation $(\vec{d}, \vec{b})$, a constraint z on the number of docks that may be moved, and lower/upper bounds l_i, u_i on the capacity of each station i, the following minimization problem

$$
\begin{aligned}
\text{minimize}_{(\vec{d},\vec{b})} \quad & c(\vec{d}, \vec{b}) & \\
s.t. \quad & \textstyle\sum_i d_i + b_i & \leq D + B, \\
& \textstyle\sum_i b_i & \leq B, \\
& \textstyle\sum_i |(\bar{d}_i + \bar{b}_i) - (d_i + b_i)| & \leq 2z, \\
\forall i \in [n]: \quad & l_i \leq \; d_i + b_i & \leq u_i.
\end{aligned}
$$

Here, the first constraint corresponds to a budget on the number of docks, the second to a budget on the number of bikes, the third to the operational constraints and the fourth to the lower and upper bound on the number of docks at each station. We assume without loss of generality that there exists an optimal solution in which the first two constraints hold with equality; to ensure this, we may add a dummy ("depot") station $\mathcal{D}$ that has $c_{\mathcal{D}}(\cdot, \cdot) = 0$, $l_{\mathcal{D}} = u_{\mathcal{D}} = B$, and run the

algorithm with a dock-budget of $D + 2B$. Here, D may include docks that are currently part of the system as well as inventory that is meant to be added to the system. In fact, in Appendix 12.3 we show that we can also efficiently solve an optimization problem that involves an additional trade-off between the value of D and the value of z.

4.2 A Discrete Gradient-Descent Algorithm

We begin this section by proving that $c^X(\cdot,\cdot)$ fulfills the following inequalities and is thus multimodular; while we are motivated by the UDFs defined in the last chapter, our algorithm only relies on these inequalities being satisfied.

Definition 1 (Hajek [1985], Altman et al. [2000])**.** *A function $f : \mathbb{N}_0^2 \to \mathbb{R}$ with*

$$f(d+1,b+1) - f(d+1,b) \geq f(d,b+1) - f(d,b); \tag{1}$$

$$f(d-1,b+1) - f(d-1,b) \geq f(d,b) - f(d,b-1); \tag{2}$$

$$f(d+1,b-1) - f(d,b-1) \geq f(d,b) - f(d-1,b); \tag{2}$$

for all d, b such that all terms are well-defined, is called multimodular*. For future reference, we also define the following implied[3] additional inequalities:*

$$f(d+2,b) - f(d+1,b) \geq f(d+1,b) - f(d,b); \tag{4}$$

$$f(d,b+2) - f(d,b+1) \geq f(d,b+1) - f(d,b); \tag{5}$$

$$f(d+1,b+1) - f(d,b+1) \geq f(d+1,b) - f(d,b). \tag{6}$$

After proving that the user dissatisfaction functions are multimodular, we define a natural neighborhood structure on the set of feasible allocations and

[3](6) and (1) are equivalent, (1) and (2) imply (5), and (3) and (6) imply (4).

define a discrete gradient-descent algorithm on this neighborhood structure. We prove that solutions to the problem without operational constraints that are locally optimal with respect to the neighborhood structure are also globally optimal; since the algorithm only terminates when finding a local optimum, this proves that it returns a globally optimal solution. Finally, we prove that the algorithm takes z iterations to find the best allocation obtainable by moving at most z docks; this not only proves that the gradient-descent algorithm solves the minimization problem when including operational constraints, but also guarantees that doing so requires at most $O(D + B)$ iterations.

4.2.1 Multimodular

Lemma 2. *The function $c^X(\cdot, \cdot)$ is multimodular for all X.*

Proof. We prove the lemma by induction, showing that $X(t)$ is multimodular for all t. With $t = 0$, by definition, $c^{X(t)}(\cdot, \cdot) = 0$ and thus there is nothing to show. Suppose that $c^{X(0)}(\cdot, \cdot)$ through $c^{X(t-1)}(\cdot, \cdot)$ are all multimodular. We prove that $c^{X(t)}(\cdot, \cdot)$ is then multimodular as well.

We begin by proving inequality (1). Notice first that if

$$\max\{c^{X(1)}(d+1, b+1), c^{X(1)}(d+1, b), c^{X(1)}(d, b+1), c^{X(1)}(d, b)\} = 0,$$

we can use that inequality (1), by the inductive assumption, holds after $t - 1$ customers. Else, we use the inductive assumption on inequalities (4) and (5) to prove inequality (1). If $X_1 = 1$ (and $d = 0$), then both sides of the inequality are 0 and $\delta_{X(1)}(d+1, b+1) = 0$, $\delta_{X(1)}(d+1, b) = 0$, $\delta_{X(1)}(d, b+1) = 0$, and $\delta_{X(1)}(d, b) = 0$. In that case, we may use the inductive assumption on inequality (5) applied to the

remaining $t-1$ customers. If instead $X_1 = -1$ (and $b = 0$), then both sides of the inequality are -1 and we have $\delta_{X(1)}(d+1, b+1) = d+b+2$, $\delta_{X(1)}(d+1, b) = d+b+1$, $\delta_{X(1)}(d, b+1) = d+b+1$, and $\delta_{X(1)}(d, b) = d+b$, so we may apply inequality (4) inductively to the remaining $t-1$ customers.

It remains to prove inequalities (2) and (3). We restrict ourselves to inequality (2) as the proof for inequality (3) is symmetric with each X_i replaced by $-X_i$ and the coordinates of each term exchanged. As before, if

$$\max\{c^{X(1)}(d-1, b+1), c^{X(1)}(d-1, b), c^{X(1)}(d, b), c^{X(1)}(d, b-1)\} = 0,$$

the inductive assumption applies. If instead $X_1 = 1$ and the maximum is positive, then the LHS and the RHS are both 0 and we have $\delta_{X(1)}(d-1, b+1) = 0$, $\delta_{X(1)}(d-1, b) = 0$, $\delta_{X(1)}(d, b) = 0$, $\delta_{X(1)}(d, b-1) = 0$. In that case, both sides of the inequality are subsequently coupled and the inequality holds with equality.

In contrast, if $X_1 = -1$ and the maximum is positive, then $b = 1$, the RHS is -1, and the LHS is 0. In this case we have $\delta_{X(1)}(d-1, b+1) = d$, $\delta_{X(1)}(d-1, b) = d$, $\delta_{X(1)}(d, b) = d+1$, $\delta_{X(1)}(d, b-1) = d$. Let $\hat{t}$ denote the next customer such that one of the four terms changes.

If $X_{\hat{t}} = 1$, then both terms on the LHS increase by 1, so it remains 0, whereas only the negative term on the RHS increases, so the inequality holds with $0 \geq -2$. Moreover, since $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = 0$, and $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = 0$; subsequently both sides of the inequality are again coupled.

Finally, if $X_{\hat{t}} = -1$, then both terms on the RHS, but only the negative term on the LHS, increase by 1 with customer $\hat{t}$. Thus, thereafter both sides are again equal. In this case as well, both sides remain coupled thereafter since we have $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = d+b$, and $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = d+b-1$. ■

Corollary 3. *The user dissatisfaction function $c_i(\cdot,\cdot)$ is multimodular for any demand-profile p_i.*

Proof. The proof is immediate from Lemma 2 and linearity of expectation. ■

4.2.2 Algorithm

We now present our algorithm before analyzing it for settings without the operational constraints. Intuitively, in each iteration the algorithm picks one dock and at most one bike within the system and moves them from one station to another. It chooses the dock, and the bike so as to maximize the reduction in objective value. To formalize this notion, we define the *movement of a dock* via the following transformations.

Definition 4. *We shall use the notation $(\vec{v}_{-i}, \hat{v}_i) := (v_1 \dots v_{i-1}, \hat{v}_i, v_{i+1} \dots v_n)$. Similarly, $(\vec{v}_{-i,-j}, \hat{v}_i, \hat{v}_j) := (v_1 \dots \hat{v}_i \dots \hat{v}_j \dots v_n)$. Then a* dock-move *from i to j corresponds to one of the following transformations of feasible solutions:*

1. $o_{ij}(\vec{d}, \vec{b}) = ((\vec{d}_{-i,-j}, d_i - 1, d_j + 1), \vec{b})$ *– Moving one open dock from i to j;*
2. $e_{ij}(\vec{d}, \vec{b}) = (\vec{d}, (\vec{b}_{-i,-j}, b_i - 1, b_j + 1))$ *– Moving a dock & a bike from i to j;*
3. $E_{ijh}(\vec{d}, \vec{b}) = ((\vec{d}_{-i,-h}, d_i - 1, d_h + 1), (\vec{b}_{-j,-h}, b_j + 1, b_h - 1))$ *– Moving a dock from i to j and one bike from h to j;*
4. $O_{ijh}(\vec{d}, \vec{b}) = ((\vec{d}_{-j,-h}, d_j + 1, d_h - 1), (\vec{b}_{-i,-h}, b_i - 1, b_h + 1))$ *– Moving one bike from i to h and one open dock from i to j (equivalently, one full dock from i to j).*

Further, we define the neighborhood $N(\vec{d}, \vec{b})$ of $(\vec{d}, \vec{b})$ as the set of allocations that are one

dock-move away from $(\vec{d}, \vec{b})$*. Formally,*

$$N(\vec{d}, \vec{b}) := \{o_{ij}(\vec{d}, \vec{b}), e_{ij}(\vec{d}, \vec{b}), E_{ijh}(\vec{d}, \vec{b}), O_{ijh}(\vec{d}, \vec{b}) : i, j, h \in [n]\}.$$

Finally, define the dock-move distance *between* $(\vec{d}, \vec{b})$ *and* $(\vec{d'}, \vec{b'})$ *as* $\sum_i |(d_i + b_i) - (d'_i + b'_i)|$.

This gives rise to a very simple algorithm: we first find the optimal allocation of bikes for the current allocation of docks; the convexity of each c_i in the number of bikes, with fixed number of docks, implies that this can be done greedily by taking out all the bikes and then adding them one by one. Then, while there exists a dock-move that improves the objective, we find the best possible such dock-move and update the allocation accordingly. Once no improving move exists, we return the current solution.

REMARK. A fast implementation of the above algorithm involves six binary heaps for the six possible ways in which the objective at each station can be affected by a dock-move: an added bike, a removed bike, an added empty dock, a removed empty dock, an added full dock, or a removed full dock. In each iteration, we use the heaps to find the best-possible move (in $O(1)$ time) and update only the values in the heaps that correspond to stations involved. The latter requires a constant number of oracle calls to evaluate the cost functions locally as well as heap-operations that can be implemented in amortized $O(n \log(n))$ time.

4.2.3 Optimality without Operational Constraints

We prove that the algorithm returns an optimal solution by showing that an allocation $(\vec{d}, \vec{b})$ that is locally optimal with respect to $N(\cdot, \cdot)$ must also be globally

optimal. Thus, when the algorithm terminates, the solution output is optimal. Before we prove Lemma 7 to establish this, we first define an allocation of bikes and docks as *bike-optimal* if it minimizes the objective among allocations with the same number of docks at each station and prove that bike-optimality is an invariant of the algorithm.

Definition 5. *Define an allocation* $(\vec{d}, \vec{b})$ *as* bike-optimal *if*

$$(\vec{d}, \vec{b}) \in \arg \min_{(\hat{\vec{d}}, \hat{\vec{b}}): \forall i, d_i + b_i = \hat{d}_i + \hat{b}_i,\ \sum_i \hat{b}_i = B} \{c(\hat{\vec{d}}, \hat{\vec{b}})\}.$$

Lemma 6. *Suppose* $(\vec{d}, \vec{b})$ *is bike-optimal. Given* i *and* j*, one of the possible dock-moves from* i *to* j*, i.e.,* $e_{ij}(\vec{d}, \vec{b}), o_{ij}(\vec{d}, \vec{b}), E_{ijh}(\vec{d}, \vec{b})$*, or* $O_{ijh}(\vec{d}, \vec{b})$*, is bike-optimal. Equivalently, when moving a dock from* i *to* j*, one has to move at most one bike within the system to maintain bike-optimality.*

Proof. It is known that multimodular functions fulfill certain convexity properties (see e.g., Murota [2003], Raviv and Kolka [2013]); in particular, for fixed d and b it is known that $c_i(k, d+b-k)$ is a convex function of $k \in \{0, \ldots, d+b\}$. Thus, if the best allocation out of $e_{ij}(\vec{d}, \vec{b}), o_{ij}(\vec{d}, \vec{b}), E_{ijh}(\vec{d}, \vec{b})$, and $O_{ijh}(\vec{d}, \vec{b})$, was not bike-optimal, there would have to be two stations such that moving a bike from one to the other improves the objective. By the bike-optimality of $(\vec{d}, \vec{b})$, at least one of these two stations must have been involved in the move. We prove that the result holds if e_{ij} was the best of the set of possible moves $\{e_{ij}, o_{ij}, E_{ijh}, O_{i,j,h}\}_{i,j,h \in [n]}$ – the other three cases are essentially symmetric. Let ℓ denote a generic third station. Then a bike improving the objective could correspond to one being moved from ℓ to j, from i to j, from i to ℓ, from ℓ to i, from j to ℓ or from j to i. In this case, moves from ℓ to j, i to j and i to ℓ yield the allocations $E_{ij\ell}(\vec{d}, \vec{b})$, $o_{ij}(\vec{d}, \vec{b})$ and $O_{ij\ell}(\vec{d}, \vec{b})$, respectively. Since e_{ij} is assumed to be the minimizer among the possible dock-moves, none of these have objective smaller than that

of $e_{ij}(\vec{d}, \vec{b})$. It remains to show that moving a bike from ℓ to i, j to ℓ or j to i yields no improvement. These all follow from bike-optimality of $(\vec{d}, \vec{b})$ and the multimodular inequalities. Specifically, an additional bike at i yields less improvement and a bike fewer at j has greater cost in $e_{ij}(\vec{d}, \vec{b})$ than in $(\vec{d}, \vec{b})$, since

$$c_i(d_i - 1, b_i) - c_i(d_i - 2, b_i + 1) \leq c_i(d_i, b_i) - c_i(d_i - 1, b_i + 1)$$

$$c_j(d_j + 2, b_j - 1) - c_j(d_j + 1, b_j) \geq c_j(d_j + 1, b_j - 1) - c_j(d_j, b_j).$$

Both of the above inequalities follow from inequality (3). ■

By Lemma 6, to prove optimality of the algorithm, it now suffices to prove that bike-optimal solutions that are locally optimal w.r.t. our neighborhood structure are also global optimal.

Lemma 7. *Suppose $(\vec{d}, \vec{b})$ is bike-optimal, but does not minimize $c(\cdot, \cdot)$ subject to budget constraints. Let $(\vec{d}^\star, \vec{b}^\star)$ denote a better (feasible) solution at minimum dock-distance from $(\vec{d}, \vec{b})$. Since $(\vec{d}, \vec{b})$ is bike-optimal, there exist j and k such that $b_j + d_j < b_j^\star + d_j^\star$ and $b_k + d_k > b_k^\star + d_k^\star$. Pick any such j and k; then either there exists a dock-move to j or one from k that improves the objective.*

Proof. The proof of the lemma follows a a case-by-case analysis, each of which resembles the same idea: $(\vec{d}^\star, \vec{b}^\star)$ minimizes the dock-move distance to $(\vec{d}, \vec{b})$ among solutions with lower function value than $(\vec{d}, \vec{b})$, i.e., among all $(\vec{d}^\star, \vec{b}^\star)$ such that $\sum_i d_i + b_i = \sum_i d_i^\star + b_i^\star$, $\sum_i b_i = \sum_i b_i^\star$, and $c(\vec{d}^\star, \vec{b}^\star) < c(\vec{d}, \vec{b})$, $(\vec{d}^\star, \vec{b}^\star)$ has minimum dock-move distance to $(\vec{d}, \vec{b})$. We show that with j and k as in the statement of the lemma, either there exists a dock-move to j, or one from k that improves the objective, or there exists a solution $(\vec{d}^{\star\star}, \vec{b}^{\star\star})$ with objective value lower than $(\vec{d}, \vec{b})$, $\sum_i d_i + b_i = \sum_i d_i^{\star\star} + b_i^{\star\star}$, and $\sum_i b_i = \sum_i b_i^{\star\star}$, such that $(\vec{d}^{\star\star}, \vec{b}^{\star\star})$ has smaller dock-move distance to $(\vec{d}, \vec{b})$. Since the latter contradicts our choice

of $(\vec{d}^\star, \vec{b}^\star)$, this proves, that in $(\vec{d}, \vec{b})$ there must be a dock-move to j or one from k that yields a lower objective. We distinguish among the following cases:

1. $d_j < d_j^\star$ and $d_k > d_k^\star$;
2. $b_j < b_j^\star$ and $b_k > b_k^\star$;
3. $d_j < d_j^\star$, $b_k > b_k^\star$, and $b_j \geq b_j^\star$
   (a) and there exists ℓ with $d_l + b_l \geq d_l^\star + b_l^\star$, $b_l < b_l^\star$;
   (b) and there exists ℓ with $d_l + b_l < d_l^\star + b_l^\star$, $b_l < b_l^\star$;
   (c) for all $\ell \notin \{j, k\}$, we have $b_l \geq b_l^\star$, so $\sum_i b_i > \sum_i b_i^\star$;
4. $b_j < b_j^\star$, $d_j \geq d_j^\star$, $b_k \leq b_k^\star$ and $d_k > d_k^\star$,
   (a) and there exists ℓ with $d_\ell + b_\ell > d_\ell^\star + b_\ell^\star$ and $b_\ell > b_\ell^\star$;
   (b) and there exists ℓ with $d_\ell + b_\ell \leq d_\ell^\star + b_\ell^\star$ and $b_\ell > b_\ell^\star$;
   (c) for all $\ell \notin \{j, k\}$, we have $b_\ell \leq b_\ell^\star$, so $\sum_i b_i < \sum_i b_i^\star$.

We show that in case (1) a move from k to j yields improvement. The proof for case (2) is symmetric. Thus, in cases (3a) and (4a) there exists a move from k to ℓ, respectively from ℓ to j, that yields improvement. Since the proofs for cases (3b) and (4b) are also symmetric, we present only the proofs for (3b). Cases (3c) and (4c) contradict our assumption that $\sum_i b_i = \sum_i b_i^\star$ and can thus be excluded. For case (1), we define $(\vec{d}^{\star\star}, \vec{b}^{\star\star}) = e_{jk}(\vec{d}^\star, \vec{b}^\star)$, so

$$c((\vec{d^{\star\star}}, \vec{b}^{\star\star})) - c((\vec{d}^\star, \vec{b}^\star)) = (c_j(d_j^\star - 1, b_j^\star) - c_j(d_j^\star, b_j^\star) + c_k(d_k^\star + 1, b_k^\star) - c_k(d_k^\star, b_k^\star).$$

Given that $\sum_i |d_i - d_i^\star| + |b_i - b_i^\star| > \sum_i |d_i - d_i^{\star\star}| + |b_i - b_i^{\star\star}|$, the definition of $(\vec{d}^\star, \vec{b}^\star)$ implies that this difference must be positive. Setting $(\vec{d}', \vec{b}') = e_{kj}(\vec{d}, \vec{b})$, we bound

$$c((\vec{d}, \vec{b})) - c((\vec{d}', \vec{b}')) = \Big(c_j(d_j, b_j) - c_j(d_j + 1, b_j)\Big) + \Big(c_k(d_k, b_k) - c_k(d_k - 1, b_k)\Big)$$

$$\geq \Big(c_j(d_j^\star - 1, b_j^\star) - c_j(d_j^\star, b_j^\star)\Big) + \Big(c_k(d_k^\star + 1, b_k^\star) - c_k(d_k^\star, b_k^\star)\Big) = c(\vec{d}^{\star\star}, \vec{b}^{\star\star}) - c(\vec{d}^\star, \vec{b}^\star) > 0.$$

We prove the inequality between the second and third expression by first showing that

$$c_j(d_j, b_j) - c_j(d_j + 1, b_j) \geq c_j(d_j^\star - 1, b_j^\star) - c_j(d_j^\star, b_j^\star).$$

Applying inequality (3) given in the definition of multimodularity, t times (where $t \geq 0$) allows us to bound the RHS by $c_j(d_j^\star - 1 - t, b_j^\star + t) - c_j(d_j^\star - t, b_j^\star + t)$. Setting $t = d_j^\star - d_j - 1 \geq 0$, we then find that the RHS is bounded above by

$$c_j(d_j, b_j^\star + d_j^\star - d_j - 1) - c_j(d_j + 1, b_j^\star + d_j^\star - d_j - 1).$$

On the other hand, applying inequality (6) repeatedly to the LHS shows that $\forall s \geq 0$, the LHS is at least $c_j(d_j, b_j + s) - c_j(d_j + 1, b_j + s)$. Hence, by setting $s = b_j^\star + d_j^\star - d_j - b_j - 1$, which is non-negative since $b_j + d_j < b_j^\star + d_j^\star$, we bound the LHS from below by

$$c_j(d_j, b_j + b_j^\star + d_j^\star - d_j - b_j - 1) - c_j(d_j + 1, b_j + b_j^\star + d_j^\star - d_j - b_j - 1).$$

This equals the upper bound on the RHS and thus proves the desired inequality. Similarly, to show

$$c_k(d_k - 1, b_k) - c_k(d_k, b_k) \leq c_k(d_k^\star, b_k^\star) - c_k(d_k^\star + 1, b_k^\star), \tag{4.1}$$

we apply inequality (3) $d_k - d_k^\star - 1$ times to bound the LHS in (4.1) by $c_k(d_k^\star, b_k + d_k - d_k^\star + 1) - c_k(d_k^\star + 1, b_k + d_k - d_k^\star + 1)$. Thereafter, we apply inequality (5) $b_k + d_k - d_k' + 1 - b_k' \geq 0$ times to obtain the desired bound.

In case (3b), we define $(\vec{d}^{\star\star}, \vec{b}^{\star\star}) = E_{jk\ell}(\vec{d}^\star, \vec{b}^\star)$ and $(\vec{d}', \vec{b}') = O_{kj\ell}(\vec{d}, \vec{b})$. As in the first case, we need to show that $c((\vec{d}, \vec{b})) - c((\vec{d}', \vec{b}')) \geq c(\vec{d}^{\star\star}, \vec{b}^{\star\star}) - c(\vec{d}^\star, \vec{b}^\star)$.

Since all terms not involving j, k, and ℓ cancel out and the terms involving j and k can be bounded the same way as before, deriving that

$$c_\ell(d_\ell, b_\ell) - c_\ell(d_\ell - 1, b_\ell + 1) \geq c_\ell(d_\ell^\star + 1, b_\ell^\star - 1) - c_\ell(d_\ell^\star, b_\ell^\star)$$

suffices. We obtain this by repeatedly applying inequalities (3) and (4) to the LHS. ■

4.2.4 Operational Constraints & Running Time

In this section, we show that the allocation algorithm is optimal for the operational constraints introduced in Section 4.1 by proving that in z iterations it finds the best allocation obtainable by moving at most z docks. We thereby also provide an upper bound on the running-time of the algorithm, since any two feasible dock-allocations can be at most $D + B$ dock-moves apart. We begin by first formally defining the set of feasible solutions with respect to the operational constraints.

Definition 8. *Define the* z-ball $S_z(\vec{d}, \vec{b})$ *around* $(\vec{d}, \vec{b})$ *as the set of allocations with dock-move distance at most* $2z$*, i.e.,* $S_0(\vec{d}, \vec{b}) = \{(\vec{d}, \vec{b})\}$ *and*

$$S_z(\vec{d}, \vec{b}) = S_{z-1}(\vec{d}, \vec{b}) \cup \Big(\bigcup_{(\vec{d}', \vec{b}') \in S_{z-1}(\vec{d}, \vec{b})} N(\vec{d}', \vec{b}') \Big).$$

We now want to prove that Lemma 7 continues to hold in the constrained setting; in particular, we show that with the operational constraints as well, local optima are global optima.

Lemma 9. *With* $(\vec{d^+}, \vec{b^+}) \in S_z(\vec{d}, \vec{b}) \setminus S_{z-1}(\vec{d}, \vec{b})$ *bike-optimal and* $c(\vec{d^\star}, \vec{b^\star}) < c(\vec{d^+}, \vec{b^+})$ *for some* $(\vec{d^\star}, \vec{b^\star}) \in S_z(\vec{d}, \vec{b}) \setminus S_{z-1}(\vec{d}, \vec{b})$*, there exists* $(\vec{d'}, \vec{b'}) \in S_z(\vec{d}, \vec{b}) \cap N(\vec{d^+}, \vec{b^+})$ *such that* $c(\vec{d'}, \vec{b'}) < c(\vec{d^+}, \vec{b^+})$.

Proof. Notice that this lemma closely resembles Lemma 7: the sole difference lies in Lemma 7 not enforcing the dock-move to maintain a bound on the distance to some allocation $(\vec{d}, \vec{b})$.

Define $(\vec{d}^\star, \vec{b}^\star)$ as in Lemma 7 with the additional restriction that $(\vec{d}^\star, \vec{b}^\star)$ be in $S_z(\vec{d}, \vec{b})$, i.e., pick a solution in $S_z(\vec{d}, \vec{b})$ that minimizes the dock-move distance to $(\vec{d^+}, \vec{b^+})$ among solutions with strictly smaller objective value. We argue again that bike-optimality of $(\vec{d^+}, \vec{b^+})$ implies that there exist j and k, such that $d_j^+ + b_j^+ < d_j^\star + b_j^\star$, and $d_k^+ + b_k^+ > d_k^\star + b_k^\star$. Further, for any such j and k, we can apply the proof of Lemma 7 to find a move involving at least one of the two that decreases both the objective value and the dock-move distance to $(\vec{d}^\star, \vec{b}^\star)$.

We aim to find j and k such that the move identified, say from ℓ to m, is guaranteed to remain within $S_z(\vec{d}, \vec{b})$. Notice that $|\{j\} \cap \{m\}| + |\{k\} \cap \{\ell\}| \geq 1$. We know that $d_m^\star + b_m^\star > d_m^+ + b_m^+$ and $d_\ell^\star + b_\ell^\star < d_\ell^+ + b_\ell^+$. Suppose the move from ℓ to m yields a solution outside of $S_z(\vec{d}, \vec{b})$. It follows that $d_m^+ + b_m^+ \geq d_m + b_m$ and $d_\ell^+ + b_\ell^+ \leq d_\ell + b_\ell$, so in particular either $d_j^+ + b_j^+ \geq d_j + b_j$ or $d_k^+ + b_k^+ \leq d_k + b_k$. Thus, if we can identify j and k such that those two inequalities do not hold, we are guaranteed that the identified move remains within $S_z(\vec{d}, \vec{b})$. Define $k := \arg\max_i \{d_i^+ + b_i^+ - \max\{d_i + b_i, d_i^\star + b_i^\star\}\}$. We can then write

$$\max_i \{d_i^+ + b_i^+ - \max\{d_i + b_i, d_i^\star + b_i^\star\}\} \quad \geq \min_i \{1, \max_{i: d_i^+ + b_i^+ > d_i + b_i} \{(d_i^+ + b_i^+) - (d_i^\star + b_i^\star)\}\}.$$

The minimum is at least 1 unless it is the case for all i that if $d_i^+ + b_i^+ > d_i + b_i$ then $d_i^+ + b_i^+ \leq d_i^\star + b_i^\star$. Thus, unless the above condition fails, we have identified a k with the required properties. Suppose the condition does fail. Then $\sum_i d_i + b_i = \sum_i d_i^\star + b_i^\star = \sum_i d_i^+ + b_i^+$ and

$$2z = \sum_i |(d_i + b_i) - (d_i^\star + b_i^\star)| = \sum_i |(d_i + b_i) - (d_i^+ + b_i^+)|$$

imply that for all i with $\max\{d_i^+ + b_i^+, d_i^\star + b_i^\star\} > d_i + b_i$, we have $d_i^+ + b_i^+ = d_i^\star + b_i^\star$. Thus, it must be the case that m fulfills $d_m^+ + b_m^+ < d_m + b_m$. The argument for j is symmetric. ■

By Lemma 9, it suffices to show that the solution found in the zth iteration is locally optimal among solutions in $S_z(\vec{d}, \vec{b})$ to prove the following theorem.

Theorem 10. *Starting with a bike-optimal allocation $(\vec{d}, \vec{b})$, in the z-th iteration, the discrete gradient-descent algorithm finds an optimal allocation among those in $S_z(\vec{d}, \vec{b})$. Hence, the discrete gradient-descent algorithm terminates in at most $D + B$ iterations.*

Proof. We prove the theorem by induction on z. The base-case $z = 0$ holds trivially. Suppose in the zth iteration, the discrete gradient-descent algorithm has found the allocation $(\vec{d}^z, \vec{b}^z) \in \arg\min_{(\vec{d}^\star, \vec{b}^\star) \in S_z(\vec{d}, \vec{b})} c(\vec{d}^\star, \vec{b}^\star)$. We need to show that $(\vec{d}^{z+1}, \vec{b}^{z+1})$ minimizes the objective among solutions in $S_{z+1}(\vec{d}, \vec{b})$, where

$$(\vec{d}^{z+1}, \vec{b}^{z+1}) := \arg \min_{(\vec{d}^{z+1}, \vec{b}^{z+1}) \in N(\vec{d}^z, \vec{b}^z))} \{c(\vec{d}^{z+1}, \vec{b}^{z+1})\}.$$

We first observe that by Lemma 9, it suffices to show that there is no better solution in $S_{z+1}(\vec{d}, \vec{b})$ that is just one dock-move away from $(\vec{d}^{z+1}, \vec{b}^{z+1})$. Further, by Lemma 6 and the choice of dock-moves in the discrete gradient-descent algorithm we know that $(\vec{d}^{z+1}, \vec{b}^{z+1})$ must be bike-optimal. Let i be the station from which a dock was moved and let j be the station to which it was moved in the $z + 1$st iteration. We denote a third station by h if the $z + 1$st move involved a third one (recall that a dock-move from i to j can take an additional bike from i to a third station h or take one from h to j). We can then immediately exclude the following cases:

1. Any dock-move in which i receives a dock from some station ℓ, including possibly $\ell = j$ or $\ell = h$, can be excluded since the discrete gradient-descent algorithm could have chosen to take a dock from ℓ instead of i and found a bike-optimal allocation (by Lemma 6).
2. The same holds for any dock-move in which a dock is taken from j.
3. A dock-move not involving either of i, j, and h yields the same improvement as it would have prior to the $z+1$st iteration. Furthermore, if such a dock-move yields a solution within $S_{z+1}(\vec{d}, \vec{b})$, then prior to the $z+1$st iteration it would have yielded a solution within $S_z(\vec{d}, \vec{b})$. Hence, by the induction assumption, it cannot yield any improvement.
4. A dock-move from station i (or to j), as is implied by the fourth, fifth, and sixth inequality in the definition of multimodularity increases the objective at i more (decreases the objective at j less) than it would have prior to the $z+1$st iteration.

We are left with dock-moves either from or to h as well as dock-moves that involve one of the three stations only via a bike being moved. Suppose that the dock-move in iteration $z+1$ was E_{ijh}; the case of O_{ijh} is symmetric. In this case, by inequality (2), a subsequent move of a dock and a bike from h, i.e., $o_{h\ell}$ or $O_{h\ell m}$ for some m, increases the objective at h by at least as much as it did before and can thus be excluded. The same holds for the move of an empty dock to h (by inequality (3)).

However, subsequent moves of an empty dock from h (or a full dock to h) have a lower cost (greater improvement) and require a more careful argument. Suppose $e_{h\ell}$ yielded an improvement – the cases for $E_{h\ell m}$, $o_{\ell h}$, and $E_{\ell hm}$ are similar. Notice first that if it were the case that $d_h^z + b_h^z > d_h + b_h$ and $d_\ell^z + b_\ell^z < d_\ell +$

b_ℓ, then $e_{h\ell}(E_{ijh}(\vec{d}^z, \vec{b}^z)) \in S_z(\vec{d}, \vec{b})$ and has a lower objective than $(\vec{d}^z, \vec{b}^z)$ which contradicts the inductive assumption. Furthermore, since it must be the case that $e_{h\ell}(E_{ijh}(\vec{d}^z, \vec{b}^z))$ is an element of $S_{z+1}(\vec{d}, \vec{b})$ but not of $S_z(\vec{d}, \vec{b})$, it must also follow that either

1. $d_h^z + b_h^z > d_h + b_h$ and $d_\ell^z + b_\ell^z \geq d_\ell + b_\ell$ or
2. $d_h^z + b_h^z \leq d_h + b_h$ and $d_\ell^z + b_\ell^z < d_\ell + b_\ell$,

since otherwise a dock-move from h to ℓ would either yield a solution in S_z or one not in S_{z+1}. Notice further that the inductive assumption implies that $(\vec{d}^{z+1}, \vec{b}^{z+1}) \notin S_z(\vec{d}, \vec{b})$. Thus, we know that $d_i^{z+1} + b_i^{z+1} < d_i + b_i$ and $d_j^{z+1} + b_j^{z+1} < d_j + b_j$. We can thus argue in the following way about

$$c(e_{h\ell}(\vec{d}^{z+1}, \vec{b}^{z+1})) - c(\vec{d}^{z+1}, \vec{b}^{z+1}) = \quad c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1) + c_l(d_\ell^z + 1, b_\ell^z) - c_l(d_\ell^z, b_\ell^z).$$

In the first case, since $o_{h\ell}(\vec{d}^z, \vec{b}^z) \in S_z(\vec{d}, \vec{b})$, the inductive assumption implies that $c_h(d_h^z, b_h^z - 1) + c_j(d_j^z, b_j^z + 1) \geq c_h(d_h^z, b_h^z) + c_j(d_j^z, b_j^z)$. Further, by the choice of the discrete gradient-descent algorithm, an additional empty dock at ℓ has no more improvement than an additional dock and an additional bike at j minus the cost of taking the bike from h; otherwise, the discrete gradient-descent algorithm would have moved an empty dock from h to ℓ in the $z+1$st iteration. Thus,

$$c_\ell(d_\ell^z + 1, b_\ell^z) - c_\ell(d_\ell^z, b_\ell^z) \leq c_j(d_j^z, b_j^z) - c_j(d_j^z, b_j^z + 1) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z)$$

$$\leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z, b_h^z) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z) \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1),$$

implying that $c(e_{h\ell}(\vec{d}^{z+1}, \vec{b}^{z+1})) - c(\vec{d}^{z+1}, \vec{b}^{z+1}) \geq 0$.

In the second case, since we know that $e_{i\ell}(\vec{d}^z, \vec{b}^z) \in S_z(\vec{d}, \vec{b})$, the inductive assumption implies $c_\ell(d_\ell^z + 1, b_\ell^z) + c_i(d_i^z - 1, b_i^z) \geq c_\ell(d_\ell^z, b_\ell^z) + c_i(d_i^z, b_i^z)$. Further, the

choice of the discrete gradient-descent algorithm to take the dock from i, not h, implies that $c_i(d_i^z, b_i^z) - c_i(d_i^z - 1, b_i^z) \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1)$. Combining these two inequalities again, we again see that $e_{h\ell}$ does not yield an improvement.

The remaining cases are ones in which a move involves only i, j, or h as the third station that a bike is taken from/added to. Suppose that the transformation in iteration $z + 1$ was E_{ijh} – the other cases are similar. A subsequent move of a bike to i (by inequality (3)) or j (by inequality (2)) yields at most the improvement that it would have had prior to iteration $z + 1$. The same holds for taking a bike from h (by combining inequalities (2) and (3)). Thus, the remaining cases are those in which a bike is taken from i or j as well as the ones in which a bike is added to h.

For a bike taken from i, notice that the algorithm's choice was to take a bike from h rather than from i, so the increase in objective in taking it from i now is at least what it was at h in the $z + 1$st iteration. Similarly, since the discrete gradient-descent algorithm chose E_{ijh} over e_{ij}, taking the bike from j has cost at least the cost it had prior to the $z + 1$st iteration at h. But since $E_{\ell mh}$, for some ℓ and m for which it was feasible before the $z + 1$st iteration, did not yield an improvement then, it follows that $E_{\ell mi}$ and $E_{\ell mj}$ do not yield an improvement after the $z + 1$st iteration.

For a bike added to h, the argument is similar to the one about a dock taken from h after a bike was taken from h. For $O_{\ell mh}$ to be feasible, for some ℓ, m within S_{z+1}, it must be the case that either $d_m^z + b_m^z < d_m + b_m$ or $d_\ell^z + b_\ell^z > d_\ell + b_\ell$.

In the former case, $e_{im}(\vec{d}^z, \vec{b}^z) \in S_z(\vec{d}, \vec{b})$, so the inductive assumption implies that the increase in cost of taking a dock from i in the $z + 1$st iteration is at least

the decrease realized by moving a dock to m. But the increase in objective in taking a dock and a bike from ℓ is at least the increase at i and h in the $z+1$st iteration, since otherwise the discrete gradient-descent algorithm would have taken the bike and dock from ℓ. Hence, the decrease in objective at h and at m is bounded above by the increase in objective at ℓ.

In the latter case, the inductive assumption implies that an increase in objective at ℓ due to $O_{\ell mh}$ is bounded below by the increase in objective prior to the $z+1$st iteration due to $o_{\ell j}$ (since $o_{\ell j}(\vec{d}^z, \vec{b}^z) \in S_z(\vec{d}, \vec{b})$). That improvement however is greater than or equal to the decrease $O_{\ell mh}$ yields at m and at h combined by the choice of the discrete gradient-descent algorithm in iteration $z+1$. Thus, $O_{\ell mh}$ cannot yield an improvement. ■

4.3 Scaling Algorithm

We now extend our analysis in Section 4.2 to adapt our algorithm to a scaling algorithm that finds the optimal allocation of bikes and docks in $O(n \log(B+D))$ iterations. The idea underlying the scaling algorithm is to proceed in $\lfloor \log_2(B+D) \rfloor + 1$ phases, where in the kth phase each move involves $\alpha_k = 2^{\lfloor \log_2(B+D) \rfloor + 1 - k}$ bikes/docks rather than just one. The kth phase begins by finding the bike-optimal allocation of bikes (given the constraints of only moving α_k bikes at a time) and terminates when no move of α_k docks yields improvement. Notice first that the multimodularity of $c(d, b)$ implies multimodularity of $c(\alpha_k d, \alpha_k b)$ for all k. Thus, our analysis in the last two sections implies that in the kth phase, the scaling algorithm finds the optimal allocation among all that differ in a multiple of α_k in each coordinate from $(\vec{d}, \vec{b})$. Further, since $\alpha_{\lfloor \log_2(B+D) \rfloor + 1} = 1$, it finds the

globally optimal allocation in phase $\lfloor \log_2(B + D) \rfloor + 1$. What remains to be shown, is a bound on the number of iterations in each phase. Lemma 12 shows that that number is bounded by $O(n)$ and thus proves Theorem 11.

Theorem 11. *The scaling algorithm finds an optimal allocation in* $O(n \log(B + D))$ *iterations.*

Lemma 12. *The number of iterations in each phase is no more than* $5n$.

Proof. By Theorem 10, the number of dock-moves required in each iteration is the minimum number of dock-moves with which an optimal allocation (for that phase) could be obtained. We argue that the dock-move distance between optimal allocations in two subsequent phases cannot be too large. Notice first that if a phase requires $L > 4n$ dock-moves, then the pigeonhole principle implies that there must exist a sequence of dock-moves of length L that leads to the same allocation and involves the exact same dock-move twice. For example, if the moves e_{ij}, e_{kj} and $e_{i\ell}$ occur, then the moves e_{ij}, e_{ij}, and $e_{k\ell}$ yield the same changes, but involve e_{ij} twice. The same argument holds for the other kinds of moves. By Theorem 10, carrying out all of the L moves except for the two e_{ij} cannot yield the optimal objective for this phase. Thus, beginning the phase with all but those two moves, we find a suboptimal allocation such that doing the e_{ij} does yield an optimal allocation; this implies in particular that the e_{ij} yield improvement at that point. Now, notice that beginning the phase (before moving to a bike-optimal allocation) with the two e_{ij} moves cannot yield improvement since it gives an allocation that would have been feasible in the previous phase.

We now want to bound the improvement of the two moves at the end in terms of the improvement at the beginning. While multimodularity implies diminishing returns in each iteration of the gradient-descent algorithm, this relies

on the allocations being bike-optimal. Though the allocation at the beginning of the phase might not be bike-optimal (for the permitted number of bikes to be moved in each iteration of this phase), it cannot be more than n bike-moves away from being bike-optimal. This allows us to count each dock-move occurring in that phase as either one of the at most $4n$ moves with no duplicates or as one of the at most n moves before improvements of subsequent moves are at most what they were prior to moving to bike-optimal. Combining the two bounds, we can derive a contradiction from $L > 5n$ and thus prove the lemma. ■

4.4 Case Studies

In this Section we present the results of case studies based on data from three different bike-sharing systems: Citi Bike in NYC, Hubway in Boston, and Divvy in Chicago. Some of our results are based on an extension of the user dissatisfaction function which we first define in Section 4.4.1. Thereafter, in Section 4.4.2 we describe the data-sets underlying our computation. Finally, in Section 4.4.3 we describe the insights obtained from our analysis.

4.4.1 Long-Run-Average Cost

A topic that has come up repeatedly in discussions with operators of bike-share systems is the fact that their means to rebalance overnight does not usually suffice to begin the day with the bike-optimal allocation. In some cities, like Boston, no rebalancing at all happens overnight. As such, it is desirable to optimize for reallocations that are robust with respect to the amount of overnight

rebalancing. To capture such an objective, we define the long-run average of the user dissatisfaction function. Rather than mapping an initial condition in bikes and empty docks to the expected number of out-of-stock events over the course of one day, the long-run average maps to the average number of out-of-stock events over the course of infinitely many days. Formally, denoting by $X \oplus Y$ the concatenation of arrival sequences X and Y, i.e., $(X_1, \ldots, X_t, Y_1, \ldots, Y_s)$, we define the long-run average of a station i with demand profile p_i as follows.

Definition 13. *The long-run-average of the user dissatisfaction function at station i with demand profile p_i is*

$$c_i^{\pi}(d, b) = \lim_{T \to \infty} \frac{\mathbb{E}_{Y_j \sim p_i}[c^{Y_1 \oplus Y_2 \oplus \ldots \oplus Y_T}(d, b)]}{T}.$$

We can compute $c^{\pi}(d, b)$ by computing, for a given demand profile p_i, the transition probabilities $\rho_{xy} := \sum_X p_i(X) \mathbf{1}_{\delta_X(d_i + b_i - x, x) = y}$; in other words, ρ_{xy} is the probability of station i having y bikes at the end of a day, given that it had x at the beginning, and given that each sequence of arrivals X occurs with probability $p_i(X)$. Given the resulting transition probabilities, we define a discrete Markov chain on $\{0, \ldots, d_i + b_i\}$ and denote its stationary distribution by $\pi_{p_i}^{d_i + b_i}$. This permits us to compute $c^{\pi}(d, b) = \sum_{k=0}^{d+b} \pi_{p_i}^{d+b}(k) c_i(d + b - k, k)$. Furthermore, from the definition of $c^{\pi}(\cdot, \cdot)$ it is immediately clear that $c^{\pi}(\cdot, \cdot)$ is also multimodular; as such, all results proven in the previous sections about $c(\cdot, \cdot)$ also extend to $c^{\pi}(\cdot, \cdot)$. In fact, we observe that $c^{\pi}(\cdot, \cdot)$ depends only on the sum of its two arguments but not on the value of each. Before comparing the results of optimizing over $c^{\pi}(\cdot, \cdot)$ and over $c(\cdot, \cdot)$, we now give some intuition for why the long-run average provides a contrasting regime.

Intuition for the Long-run Average.

It is instructive to consider examples to illustrate where optimizing over the long-run average deviates from optimizing over a single day. To simplify manners, we restrict ourselves to demand profiles that only have point mass for a single sequence of arrivals. A station at which the sequence of arrivals consists of k rentals followed by k returns has the long-run average of its user dissatisfaction decrease by 2 for each of the first k docks allocated; similarly, the user dissatisfaction function over a single day decreases by 2 for each full dock added (and by 1 for each empty dock added). At a station at which only k rentals occur, the user dissatisfaction function also decreases by 1 for each of the first k full docks added; however, its long-run average remains unchanged: no matter how many docks and bikes are added, the long-run average of the station is to be empty at the beginning of the day and therefore all k customers experience out-of-stock events.

Two lessons can be derived from these examples. First, stations at which demand is antipodal (rentals in the morning, returns in the afternoon or vice-versa) tend to make better use of additional capacity in the long-run average regime. Second, optimizing over one regime can, in principle, return solutions that are very bad in the other.

4.4.2 Data Sets

We use data-sets from the bike-sharing systems of three major American cities to investigate the effect different allocations of docks might have in each city. The three cities, New York City, Boston, and Chicago, vary widely in the sizes of their

systems. When the data was collected (summer 2016), Boston had 1300 bikes and 2700 docks across 160 stations, Chicago had 4700 bikes and 9500 docks across 582 stations, and NYC had 6750 bikes and 14840 docks across 447 stations.[4]

For each station (in each system), we compute piece-wise constant Poisson arrival rates to inform our demand profiles. To be precise, we take all weeekday rentals/returns in the month of June 2016, bucket them in the 30-minute interval of the day at which they occur, and divide the number of rentals/returns at each station within each half-hour interval by the number of minutes at which the station was non-empty/non-full. We compute the user dissatisfaction functions assuming that the demand profiles stem from these Poisson arrivals (cf. O'Mahony et al. [2016] and Parikh and Ukkusuri [2014]). Some of our results in this section rely on the same procedure with data collected from other months.

Given that (in practice) we do not usually know the lower and upper bounds on the size of each station, we set the lower bound to be the current minimum capacity within the system and the upper bound to be the maximum one. Furthermore, we assume that $D + B$ is equal to the current allocated capacity in the system, i.e., we only reallocate existing docks.

4.4.3 Impact on Objective.

We summarize our results in Table 4.1. The columns Present, OPT, and 150-moved compare the objective with (i) the allocation before any docks were moved, (ii) the optimal allocation of bikes and docks, and (iii) the best allocation

[4]We remark that these numbers were obtained using the respective JSON feed of each system and do not necessarily capture the entire fleet size, e.g., in New York City a significant number of bikes is kept in depots over night.

of bikes and docks that can be achieved by moving at most 150 docks from the current allocation. The columns headed c contain the bike-optimal objective for a given allocation of docks, the columns headed c^π the long-run-average objective (for the same allocation). Two interesting observations can be made. First, though the optimizations are done over bike-optimal allocations without regard to the long-run average, the latter improves significantly in all cases. Second, in each of the cities, moving 150 docks yields a significant portion of the total improvement feasible to obtain the optimum. This stands in contrast to the large number of moves needed to find the actual optimum (displayed in the column Moves to OPT) and is due to the diminishing returns of the moves.

	Present		OPT		150-moved		Moves to OPT
City	c	c^π	c	c^π	c	c^π	
Boston	854	1118	640	943	700	984	407
Chicago	1460	2340	759	1846	1224	2123	1553
NYC	6416	9475	4829	8180	6150	9192	2721

Table 4.1: Summary of main computational results with c denoting bike-optimal, c^π the long-run-average cost.

A more complete picture of these insights is given in Figure 4.1. The x-axis shows the number of docks moved starting from the present allocation, the y-axis shows the *improvement in objective*, i.e., the difference between the initial objective and the objective after moving x docks. Each of the solid lines corresponds to different demand estimates being used to evaluate the same allocation of docks. The dotted lines (in the same colors) represent the maximum improvement, for each of the demand estimates, that can be achieved by reallocating docks; while these are not achieved through the dock moves suggested by the estimates based on June 2016 data, significant improvement is made towards them in every case. In particular, the initial moves yield approximately the same improvement for the different objectives/demand estimates. Thereafter, the various improvements

diverge, especially for the NYC data from August 2016. This may be partially due to the system expansion in NYC that occurred in the summer of 2016, but does not contradict that all allocations corresponding to values on the x-axis are optimal in the sense of Theorem 10.

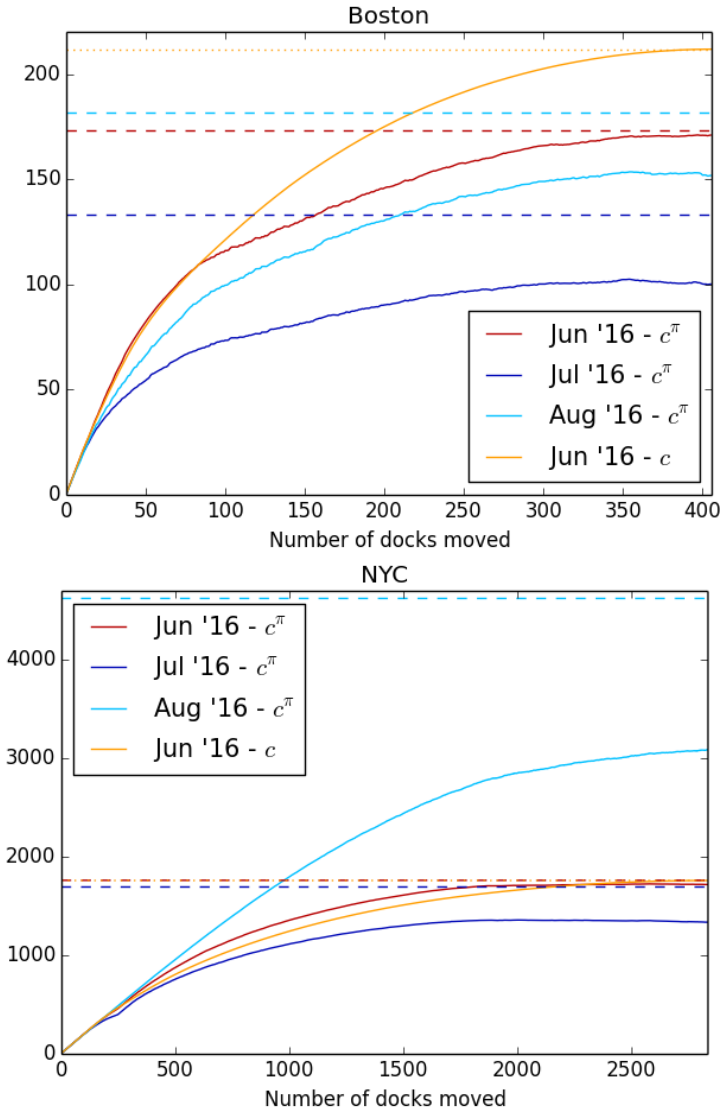


Figure 4.1: Improvement in objective for moves to bike-optimal allocation for June ’16 data.

	June 2016	March 2017	November 2017
New York City	358.7	260.3	294.6

Table 4.2: Improvement of 200 docks moved based on long-run average evaluated with demand estimates June 2016, evaluated with demand estimates from 2017.

Seasonal Effects.

As we mentioned in Section 4.1 we also consider the impact of seasonal effects. In Table 4.2 we show the improvement in objective when optimizing the movement of 200 docks in New York City based on demand estimates in June 2016 and evaluate the objective with the long-run average based on demand estimates based on March and November 2017. The estimated improvements show that optimizing with respect to any one season yields significant improvement with respect to any other.

Operational Considerations.

We conclude this part of our analysis with two remarks to contextualize the operational impact our suggestions can have. First, it is instructive to compare the estimated improvement realized through reallocating docks to the estimated improvement realized through current rebalancing efforts: according to its monthly report, Citi Bike rebalanced an average 3,452 bikes per day in June 2016 (monthly report, NYCBS [2016]). A simple coupling argument implies that a single bike yields at most a change of 1 in the user dissatisfaction function; thus, rebalancing reduced out-of-stock events by *at most* 3,452 per day (assuming that each rebalanced bike actually has that much impact is extremely conservative). Contrasting that to the estimated impact of strategically moving, for example, 500

docks diminishes the estimated number of out-of-stock events *by about as much as a fifth* of Citi Bike's (daily) rebalancing efforts.

Second, discussions with operators uncovered an additional operational constraint that can arise due to the physical design of the docks. Since these usually come in triples or quadruples, the exact moves suggested may not be feasible; e.g., it may be necessary to move docks in multiples of 4. By running the scaling algorithm without the last two iterations, we can find an allocation in which docks are only moved in multiples of 4. With that allocation, the objective of the bike-optimal allocation is 673, 847, and 4896 in Boston, Chicago, and NYC respectively, showing that despite this additional constraint almost all of the improvements can be realized.

4.5 A Posteriori Evaluation of Impact

In this section we apply the user dissatisfaction function to estimate the impact implemented changes in the system have had on out-of-stock events. One way to do so would be to estimate new demand rates after docks have been reallocated, compute new user dissatisfaction functions for stations with added (decreased) capacity, and evaluate for those stations and the new demand rates the decrease (increase) between the old and the new number of docks. A drawback of such an approach is the heavy reliance on the assumed underlying stochastic process. Instead, we present here a data-driven approach with only little reliance on assumed underlying demand profiles.

Throughout this section, we denote by d and b the number of empty docks and bikes at a station after docks were reallocated, whereas d' and b' denote the

respective numbers before docks were reallocated. Notice that while $d + b$ and $d' + b'$ are known (capacity before and after docks were moved) and b can be found on any given morning (number of bikes in the station at 6AM), we rely on some assumed value for b' — for that, in our implementation, we picked both $\min\{d' + b', b\}$ and $b \times \frac{(d'+b')}{d+b}$, that is, either the same number of bikes (unless that would be larger than the old capacity before docks were added) or the same proportion of docks filled with bikes.

4.5.1 Arrivals at Stations with Increased Capacity

In earlier sections, we assumed a known distribution for the sequence of arrivals based on which we compute the user dissatisfaction functions. In contrast, in this section we rely exclusively on observed arrivals (without any assumed knowledge of the underlying stochastic process) to analyze stations with increased capacity. This is motivated by a rigorous argument to justify that censoring need not be taken care of explicitly in this case. To formalize our argument, we need to introduce some additional notation for the arrival sequences. Recall from Section 4.1 that a sequence of customers arriving at a bike-share station to either rent or return a bike was denoted by $X = (X_1, \ldots, X_S)$ and that X included failed rentals and returns, which in practice would not be observed because they are censored. Which X_i are censored, of course, depends on the (initial) number of bikes and docks at the station. Let us denote by $\hat{X}^{(d,b)}$ the subsequence of X that only includes those customers whose rentals/returns are successful (hence, non-censored) at a station initialized with d empty docks and b bikes, i.e., the ones that do not experience out-of-stock events. Given the notation $c^X(\cdot, \cdot)$ used in Section 4.1 for a particular sequence for arrivals, we can then compute

$c^{\hat{X}^{(d,b)}}(\cdot,\cdot)$. In particular, denoting the number of empty docks and bikes without the added capacity by d', b', we may compute $c^{\hat{X}^{(d,b)}}(d', b')$. The following proposition then motivates the notion that censoring may be ignored at stations with added capacity.

Proposition 14. *For any X, $d' \leq d$ and $b' \leq b$, we have*

$$c^X(d', b') - c^X(d, b) = c^{\hat{X}^{(d,b)}}(d', b') - c^{\hat{X}^{(d,b)}}(d, b) = c^{\hat{X}^{(d,b)}}(d', b').$$

Proof. The proof of the second equality follows immediately from $\hat{X}^{(d,b)}$ including exactly those customers among X that are not censored, when a station is initialized with d empty docks and b bikes, so $c^{\hat{X}^{(d,b)}}(d, b) = 0$. Now, on the left-hand side, we can inductively go through all customers among X that are out-of-stock events when the station is initialized with d empty docks and b bikes. Since $d \geq d'$ and $b \geq b'$, each one of those increases both terms in the difference by 1. Thus, taking them out of X does not affect the value of the difference. But then, we are left with only $\hat{X}^{(d,b)}$. ■

Extension to Rebalancing

Based on our reasoning in Section 3.2, our analysis of the user dissatisfaction functions and the resulting dock allocation optimization problems (cf. Sections 4.1 and 4.2) did not consider the rebalancing of bikes. In contrast, in the *a posteriori* analysis, we are able to take rebalancing into account.

To simplify the exposition, we restrict ourselves here to rebalancing that adds bikes to a station, though the reasoning extends to rebalancing that removes bikes. The simplest approach to treat bikes added through rebalancing is to treat them

simply as returns and thus include them (as virtual customers) in the sequence of arrivals X. However, this may cause an unreasonable increase to the value of $c^{X(d,b)}(d', b')$ (when the number of bikes added is greater than the number of empty docks would have been at that point in time if the station had initially had d' empty docks). In that case, the virtual customers (corresponding to rebalanced bikes) would incur out-of-stock events and thereby increase the value of the user dissatisfaction function. A more optimistic way that also treats rebalanced bikes as virtual customers would be to redefine the user dissatisfaction function in such a way so that out-of-stock events are only incurred by returns that correspond to non-rebalanced bikes. This, in essence, decouples the user dissatisfaction functions into subsequences, each of which is evaluated independently.

Extension to Stations with Decreased Capacity.

Theorem 14 does not apply to stations with decreased capacity: suppose $d < d'$ and $b = b'$; once the station (initialized with d empty docks and b bikes) becomes full, $\hat{X}^{(d,b)}$ observe no further returns even though these would be part of $\hat{X}^{(d',b')}$. To account for out-of-stock events occurring in that way, we fill in the censored periods with demand estimates. Noticeably, this does not usually require knowledge of the full demand-profile; for example, for a station that is non-empty and non-full over the course of the day, no estimates are needed at all. Further, for periods of time in which the station is full, we only need to estimate the number of intended returns – rentals over that period of time would not be censored.

4.5.2 Measured Impact

We consider 3 stations at which capacity was increased and 3 stations at which it was decreased based on our recommendations. For two of the stations at which capacity was increased 12 docks were added, for one of them the capacity was increased by 10; the decreases were by the same amounts, so in total this involved reallocating 34 docks. In Figure 4.2 we present the impact for each weekday in April 2018 (without the extension to rebalancing). For stations with added capacity we set d and b according to the number of bikes at 6AM. We evaluated

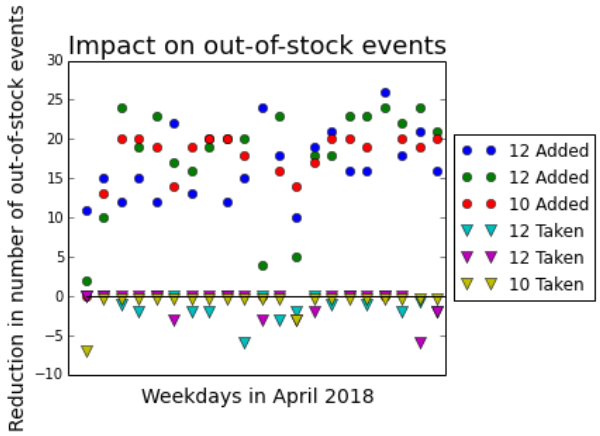


Figure 4.2: Evaluation of impact at stations with increased and decreased capacity.

$c^{\hat{X}^{(d,b)}}(d', b')$ for stations with docks added (cf. Proposition 14) using the observed arrivals $\hat{X}^{(d,b)}$ for each day. For the stations with docks taken away we estimated $\hat{X}$ by assuming a fluid number of rentals (returns) whenever the station was empty (full), where the rate is based on decensored estimated demand from the same month. We use that to compute $c^{\hat{X}^{(d,b)}}(d', b') - c^{\hat{X}^{(d,b)}}(d, b)$ for these stations. The resulting values for different implementations are summarized in Table 4.3; aggregated over the entire month, the net reduction in out-of-stock events varies between 831 and 1062.

	No Rebalancing		Rebalancing	
	$\min\{b, d' + b'\}$	$b \times (\frac{d'+b'}{d+b})$	$\min\{b, d' + b'\}$	$b \times (\frac{d'+b'}{d+b})$
Decrease where capacity was added	831.0	1121.0	882.0	1027.0
Increase where capacity was taken	0	58.7	0	59.7
Net Reduction	831.0	1062.3	882.0	967.3

Table 4.3: Estimated impact of reallocated capacity on out-of-stock events.

4.6 Running Time

Even though the reallocation of docks is a strategic question, the time to solve the associated optimization models is not irrelevant for practical considerations. Given the expensive computation of each user dissatisfaction value, an early approach to compute the LP-relaxation of the optimization problem took a whole weekend to solve — noticeably, this was due to the time to set up the LP; once it was set up, the LP could be solved quickly. While this is acceptable for a one-off analysis, in practice system operators care about regularly running different analyses that include different demand patterns, different bounds on number of docks moved, and even different bounds on station sizes. Having a fast algorithm allows system operators to run the analysis without our support: we provided them with a Jupyter notebook (Kluyver et al. [2016]) that includes the entire workflow from estimating the demand profiles to computing the user dissatisfaction functions to running the optimization problem to creating map-based visualizations of the resulting solutions (cf. Figure 4.3) and does not rely on specialized optimization software like Gurobi or CPLEX. Crucially, this workflow happens in a matter of minutes rather than hours or days (cf. Table 4.4).

To complete our analysis, we now compare the measured running times of the algorithm with and without scaling techniques. Given that the running-time of each algorithm is dominated by the computational effort to compute values of the user dissatisfaction functions (the effort for which grows as a cubic function of the capacity), we only computed values that the respective algorithm needed. In Figure 4.4, we plot the number of user dissatisfaction functions that are computed by each algorithm. It is noticeable in Chicago that the scaling algorithm created unnecessary overhead by requiring values for large capacities

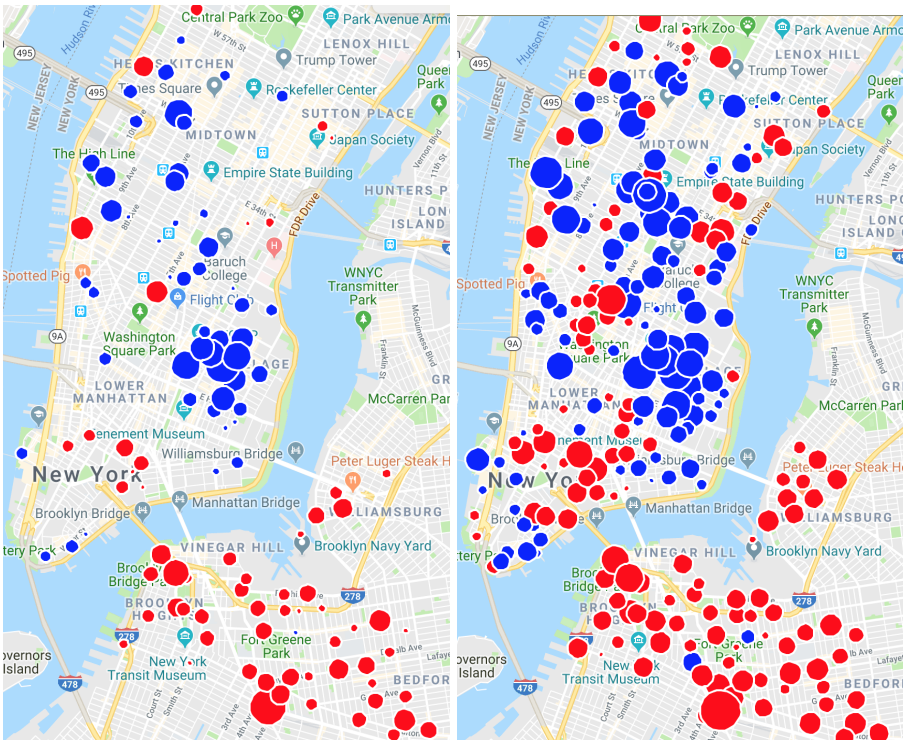


Figure 4.3: Visualization of docks moved by optimal solution for $z \in \{500, 1500\}$; red circles correspond to docks being taken, blue circles to docks being added.

at many stations that were not required without scaling. This illustrates why the algorithm without scaling outperforms the scaling algorithm in both Boston and Chicago (cf. Table 4.1). In NYC on the other hand, the scaling algorithm performed significantly better. Motivated by this contrast, we implemented a hybrid algorithm that only iterates over 8, 4, and 1, rather than all powers of 2. The hybrid outperforms both the algorithm without scaling and the scaling algorithm on all three data-sets. All three algorithms vastly outperform the linear programming based approach that needs to evaluate every value of the user dissatisfaction functions at all stations before solving.

	Running Time (Minutes)		
	No scaling	Hybrid	Scaling
New York City	18.08	14.02	12.27
Chicago	7.03	5.67	8.78
Boston	1.44	1.37	1.83

Table 4.4: Comparison of the running times of each of the three algorithms in each of the three cities

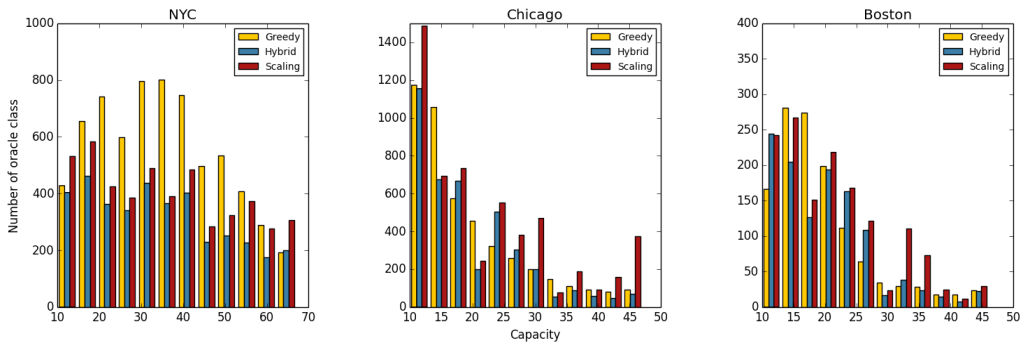


Figure 4.4: Number of UDF evaluations by each algorithm in each city.

CHAPTER 5

TRADEOFFS FOR INCENTIVES IN NEW YORK'S CITI BIKE SYSTEM

"Never Doubt That A Small Group Of Thoughtful Committed Citizens Can Change The World." – President Bartlet

In this chapter, based on Chung, Freund, and Shmoys [2018], we study the impact of *Bike Angels*, an incentive program we set up with New York City's Citi Bike system in 2015 to crowdsource some of the operational challenges related to imbalance. We develop a performance metric for both online- and offline-policies to set incentives within the system; our results indicate that though Citi Bike's original offline policy performed well in a regime in which incentives given to customers are not associated to costs, there is ample space for improvement when the costs of the incentives are taken into consideration. Motivated by these findings, we develop several online- and offline- policies to investigate the trade-offs between real-time and offline decision-making; one of our online policies has since been adopted by Citi Bike.

When the Bike Angels incentive scheme was first set up, we identified pairs of stations that (i) were close to each other, and (ii) had asymmetric demand patterns: we chose pairs of stations that were nearby to each other but over a certain time interval had the property that one station (A) overwhelmingly had bikes returned whereas the other (B) overwhelmingly had bikes rented. We then invited users that often used one of the two and asked them to switch to the other, i.e., we asked a small number of customers likely to return bikes at A in such intervals to become *Bike Angels* by instead returning bikes at B.

While the pilot showed that customer behavior could be significantly affected

through gamification (Bikeshare [2016]), it also posed questions about how to optimally design such an incentive scheme, in particular with respect to *when* and *where* to incentivize. A second version of Bike Angels, at a much larger scale, statically incentivized a sizeable fraction of stations within the system either to enourage rentals or returns throughout each rush hour. This program was viewed as a great success, in spite of the fact that the pre-determined incentives made no use of the vast amounts of historic and real-time data available. At worst, this sometimes led to incentives encouraging customers to rent (return) bikes at empty (full) stations.

In this chapter, we evaluate the efficiency of this incentive scheme with regard to its goal to reduce out-of-stock events. We apply the user dissatisfaction functions to evaluate, for every rental/return rewarded by the program, an estimate of its reduction in future out-of-stock events. Further, we design a range of different policies, dictating the times at which each station is incentivized. The design of these policies involves trade-offs between their *efficiency* and *simplicity*. Below, we explain what we mean by those two terms:

Efficiency. The main goal of *Bike Angels* is to help rebalance the system, i.e., to reduce the number of out-of-stock events. We explain in this chapter how the user dissatisfaction functions introduced by Raviv and Kolka [2013] (cf. Chapter 3) can be used to evaluate the impact of each individual rental/return on the expected number of future out-of-stock events. Combining these with a cost for each point awarded in the incentive scheme, we obtain a score for each incentivized rental/return that can be interpreted as an offline evaluation of the efficiency of the incentive. In that sense, a perfectly efficient scheme would incentivize exactly those rentals/returns that net a positive score when accounting for both

the impact on out-of-stock events and the cost of incentives.

Simplicity. Perfect efficiency, with respect to the score described above, is attainable in such an incentive scheme, but it requires the operator to decide in real-time whether or not to incentivize at each location. This is undesirable from the users' perspective as it reduces predictability on where incentives will be given in the future, i.e., when customers prepare their trip, they do not know whether or not their origin and, even more so, their destination has rentals and respectively, returns, incentivized. This raises the bar for participation. Further, from an operator's perspective, the IT set-up for a dynamic scheme requires more maintenance than, say, the completely static version.

We show that even though there exist natural trade-offs between simplicity and efficiency, there are a variety of incentive policies that span the continuum from maximally efficient and maximally dynamic to less efficient and entirely static. Our results employ a data-driven methodology to design such policies, one of which is now in place at Citi Bike.

5.1 The Incentive Scheme

Before outlining the structure of our exposition, it is worthwhile to give a high-level description of the *Bike Angel* program.

Bike angels accrue points by doing rides that benefit the balance of the system. Although the way in which stations are chosen for incentives has changed over time, the following accounting has been the guiding principle since the original pilot ended: the program awards 1 point for a trip from an incentivized rental

station to a neutral station or from a neutral station to an incentivized return station and 2 points for a trip from an incentivized rental to an incentivized return station. The exact reward structure of the incentive scheme has also changed over time, yet the basic idea that more points translate into higher rewards has remained the same. For example, in February 2018, the program rewarded users as follows NYCBS [2018]:

- For 10 points accrued, bike angels receive a 24-hour day pass;
- For every 20 points, up to 80, bike angels receive a free membership extension of 1 week;
- For every 10 points above 80, bike angels receive $1 as a gift card;
- The 5 angels with the most points receive gift cards worth $100, $75, $50, $25, and $25 respectively.

It is worthwhile to mention that bike angels occasionally get bonuses for beneficial trips on days with special conditions (e.g., due to weather); however, since our analysis mostly focuses on the choice of stations/times to incentivize, we ignore such bonuses.

We present our results as follows: in the next section we describe the datasets used in our analysis and formally define the models developed to evaluate the incentive scheme. Thereafter, we define the various policies before giving a detailed comparison of the different policies, analyzing the inherent tradeoffs between simplicity and efficiency. Finally, we conclude by reporting on the changes Citi Bike implemented for the Bike Angel program based on our analysis.

5.2 Data Analysis and Definitions

In this section, we describe the data-sets at our disposal, the system parameters we derive from them, and the models by which we evaluate the different policies for the incentive scheme.

Data-Sets

Our analysis relies on three different data-sets. First, we have a list of all trips in the system with origin location, destination location, and respective times of the start and the end of the trip (from NYCBS [2017a]). Second, we have a list that indicates which of the above trips were rewarded with points by the existing offline incentive policy (from Bikeshare [2016]); the list also indicates for each trip whether the point was awarded for the rental, the return, or both. Finally, for each minute and each station in the system, we have the number of bikes reported to be at the station at that time as well as the total number of docks available (from NYCBS [2017b]).

We use data collected over three different time periods in 2016 in our analysis: 04/01–05/13 (base), 10/03–10/30 (training) and 10/31–12/14 (testing). The base and training periods are used to help calculate the system parameters, such as the rental/return rates and the stochastic interpretation of the incentives' effect explained later in this section, and also serve as the training period for some of our incentive policy algorithms. The testing period is used to evaluate and compare our suite of policies.

Data Censoring

In our data, censoring proves to be a challenge, since the data does not accurately reflect the true demand within the system. For example, a rider wanting to rent or return a bike to a station, with all docks empty or full respectively, cannot do so; thus, the corresponding demand would not be reflected in our data sets. To account for this problem of demand-censoring, when estimating average rental/return demand, we follow the methodology of O'Mahony and Shmoys [2015] and examine station behavior on a minute-by-minute basis, only considering "active rental/return minutes", that is, minutes when a station has non-empty or non-full docks, respectively.

Rental and Return Rates

Our analysis relies on three types of rental and return rates: regular-angel, incentive-angel, and non-angel rates. Regular-angel rates are the natural rental/return rates into a station for all angel riders combined, when no incentive is provided. The incentive-angel rates are the rates for the same set of customers when incentives are given. The non-angel rates are the rates for all other customers and are assumed to be unaffected by the incentive program. In our calculation of rates, we follow the work by O'Mahony and Shmoys [2015] in assuming that the rates are independent across stations and calculate rates for every half-hour interval of the day starting from 12AM in the units bikes per minute.

To calculate the incentive-angel and non-angel rates, we take the data for a station's previous q weekdays, and divide the total number of all rentals/returns

by the specified riders by the total "active rental/return minutes" for each half-hour interval. In the results we present, q is set to 20.

Formally, for each day d, station s, and time index t, let $(m^{-}_{d,s,t}, m^{+}_{d,s,t})$ denote the active rental/return minutes, $(r^{a,+}_{d,s,t}, r^{a,-}_{d,s,t})$, $(r^{n,+}_{d,s,t}, r^{n,-}_{d,s,t})$ the number of returns/rentals for angel and non-angel riders respectively, and $D(d)$ be the set of the preceding q weekdays before day d. Then the incentive-angel return rates $\lambda^{i,+}_{d,s,t}$ for day d, station s and time index t, where $t \in \{0, 1, 2, ..., 47\}$ for the 48 half-hour intervals of the day (e.g. t=12 is 6 a.m.) are calculated as the rate per minute normalized to account for only active minutes, as follows (similar for rental rates $\lambda^{i,-}_{d,s,t}$ and non-angel return/rental rates $\lambda^{n,+}_{d,s,t}$ and $\lambda^{n,-}_{d,s,t}$):

$$\lambda^{i,+}_{d,s,t} = \frac{1}{q} \frac{\sum_{d' \in D(d)} r^{a,+}_{d',s,t}}{\sum_{d' \in D(d)} m^{+}_{d',s,t}} \tag{5.1}$$

To estimate the regular-angel rates, we used data from the base period when no incentive program was offered. The regular-angel rates are assumed to be identical across days, and are calculated by dividing the angel rider's total number of all rentals/returns by the total "active minutes" in the base period for each half-hour interval, and applying a correction factor. The correction factor accounts for any general change in system usage from the base time period to our testing time period. The correction factor is calculated by dividing the average daily total number of trips by non-angel riders during the training period by their average daily total number of trips during the base time period. Finally, to reflect our prior beliefs that incentivization actually helps our system, we capped the regular-angel rates to be at most the incentive-angel rates. If we define D to be the set of weekdays in the base time period, the regular-angel return rates are

calculated as follows (similarly for rental rates):

$$\lambda^{r,+}_{d,s,t} = min(\frac{U}{|D|}\frac{\sum_{d' \in D} r^{a,+}_{d',s,t}}{\sum_{d' \in D} m^{+}_{d',s,t}}, \lambda^{i,+}_{d,s,t}) \tag{5.2}$$

Stochastic View of Incentives

Our analysis also relies on the probability that an incentivized rental/return would not have occurred if it had not been for the incentive; in particular, the rebalancing due to a rental/return that happens regardless of the incentives should not be attributed to the incentive scheme. We compute these probabilities for each day, station and half-hour interval by dividing the difference of the incentive-angel and regular-angel rates by the incentive-angel rate. Formally, with $p^{+}_{d,s,t}$ denoting the probability that a rental was triggered by the incentives, we compute this probabilities as

$$p^{+}_{d,s,t} = \frac{\lambda^{i,+}_{d,s,t} - \lambda^{r,+}_{d,s,t}}{\lambda^{i,+}_{d,s,t}}. \tag{5.3}$$

The probability of a return having been triggered by incentives is defined symmetrically and denoted as $p^{-}_{d,s,t}$.

User Dissatisfaction Function

Given the non-angel rental and return rates, we can compute, for a given station s with ℓ bikes at time-index t, the user dissatisfaction function for an interval starting at time t, which is the expected number of intended returns when the station is at capacity plus the expected number of intended rentals when the station is empty. We denote this expectation by $c_{s,t}(\ell)$ (cf. Chapter 3).

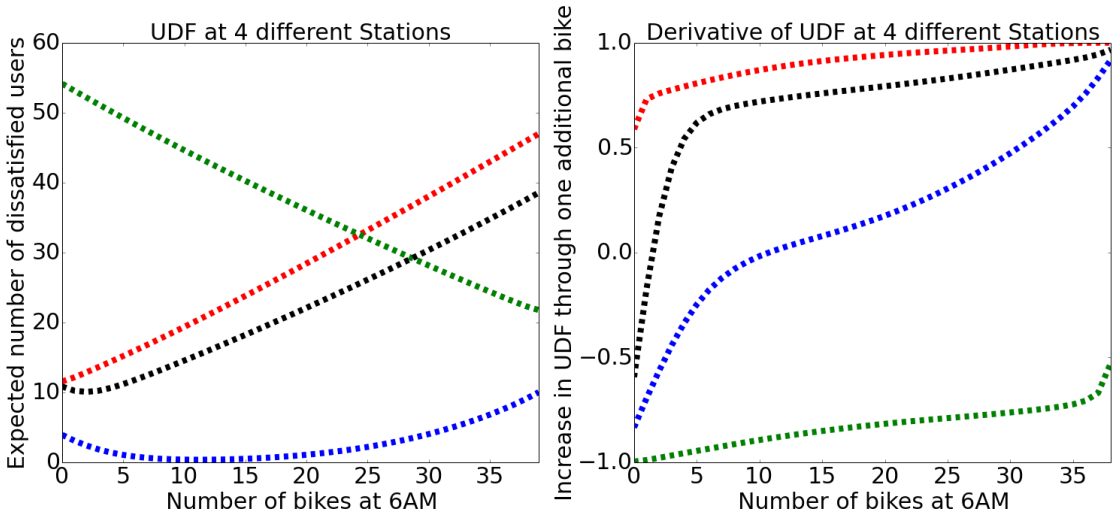


Figure 5.1: On the left-hand side, we display the user dissatisfaction function for four stations with different demand patterns as in Figure 3.1. On the right-hand side, we show the respective discrete derivatives.

Performance Metrics and Policies

For each incentivized return, we can estimate the reduction in out-of-stock events by computing the discrete derivative of the user dissatisfaction function with respect to one additional bike, evaluated as a function of the number of bikes present in the station before the arrival. Similarly, for incentivized rentals, the estimated reduction in out-of-stock events equals the difference in the value of the user dissatisfaction function evaluated with one bike less and the actual number of bikes in the station. In Figure 5.1, we display the respective user dissatisfaction functions and the value of the derivatives for three different stations at 6AM. Evaluating the user dissatisfaction functions for every half-hour interval, we estimate the impact of each rental/return on future out-of-stock events by evaluating the derivative of the user dissatisfaction function for that time-interval at the number of bikes present at the time of the return

(and similarly for rentals). If we let δ_r denote the estimated reduction on future out-of-stock events for trip r, let ℓ denote the bike-level at the station before the trip completed, and let s and t denote the station and time-index associated with the trip, we find that we may compute δ_r for a return as follows (and similarly for rentals):

$$\delta_r = c_{s,t}(\ell) - c_{s,t}(\ell + 1) \tag{5.4}$$

Although δ_r captures the impact of a return (rental) on the estimated number of out-of-stock events, it lacks two important aspects for our analysis. First, it lacks the cost associated with the incentive itself. To capture the operator's cost of the incentive scheme in the form of electronic gift cards, membership extensions, and other rewards (cf. NYCBS [2018]), we include a constant cost parameter β for every point awarded by an incentive scheme. Second, for every incentivized rental (return), we can incorporate the probability $p_{d,s,t}$ that the rental (return) would not have occurred without an incentive given; here, d, s, t is the date, station and time index respectively in which the rental (return) occurred. This gives a stochastic perspective of the causal effects of incentives, in contrast to the deterministic assumption that incentivized rentals (returns) are always caused by the incentive. Using the above, we can define for every single incentivized return (rental) the performance of incentivizing it with both the deterministic and the stochastic perspective as

$$\Delta_r = \begin{cases} \delta_r - \beta & \text{if deterministic} \\ \delta_r \cdot p_{d,s,t} - \beta & \text{if stochastic.} \end{cases} \tag{5.5}$$

Given Δ_r as a performance measure for incentivizing a single return (rental) r, we define the performance of a policy as the sum of all Δ_r corresponding to incentivized rentals (returns) r. Formally, denoting by τ the set of all rentals (returns) in the testing period, we define a policy P as a function $P : \tau \to \{0, 1\}$ identifying for each rental (return) r whether or not P incentivizes it. Then, to compute the overall performance $H(P)$ of a policy P we compute

$$H(P) = \sum_{r \in \tau} P(r) \cdot \Delta_r \tag{5.6}$$

5.3 Policies

In this section, we define a number of incentive policies and identify the fundamental differences between offline and online policies. During the time periods from which our data is drawn, two 6-hour time periods were incentivized in the Citi Bike system: 6AM-12PM and 4PM-10PM – we refer to them as the AM and PM periods, respectively. Since all of the incentivized trip data comes from only these two periods (cf. Figure 5.2), we test/apply our policies only for these periods. Furthermore, each policy treats the two time periods independently, and calculates an incentive scheme for each time period and station. From the figure, we see a generally equal distribution of rental/returns in the PM period, but a return-skewed distribution in the AM period.

Offline Policies

We first describe the offline policies. *Offline* policies are characterized by the fact that the decision, for each AM/PM period, as to which interval will be

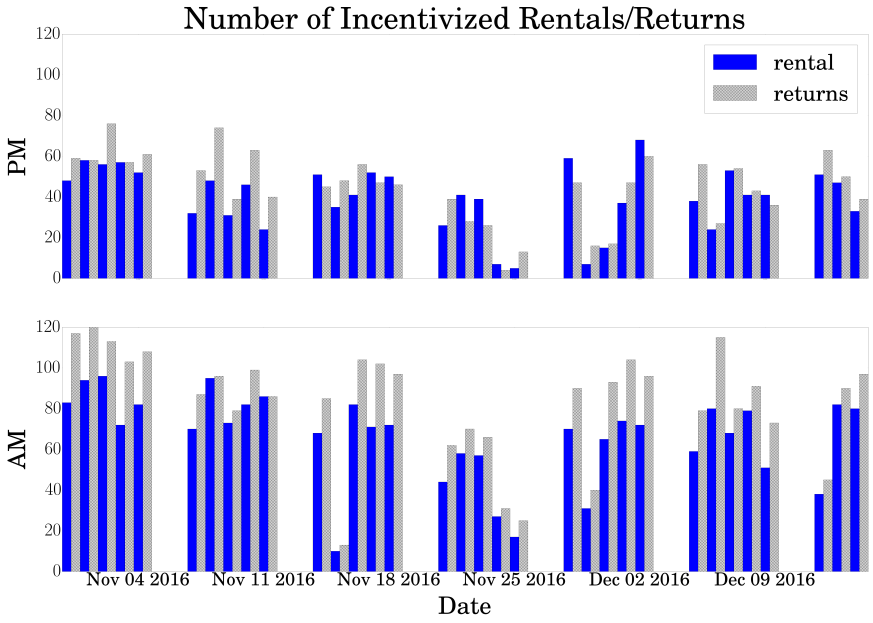


Figure 5.2: Total number of incentivized rentals/returns in the test period.

incentivized for each station is made (irrevocably) at the beginning of that time interval (6AM and 4PM for AM and PM, respectively); in doing so, these policies have access to past data and the number of bikes at that time, i.e., 6AM and 4PM, respectively. (Note that this use of term offline is different from its use in other contexts, where offline often means that the algorithm has access to *all* of the data for the input to be optimized before needing to commit to any decision.)

Static

The Static model is the simplest offline policy and was also the policy employed in Citi Bike at the time that this research project initiated. This policy determines a subset of stations to incentivize for which the entire AM period is incentivized on every day, and similar subset is determined for the PM period. This model serves as a baseline against which all other policies are compared.

Static Hindsight

The Static Hindsight model is an offline policy that chooses, for each station, the optimal continuous incentive period when looking back in hindsight for the past q' weekdays. More precisely, the model assumes an incentive interval can start/stop every 30-minutes, starting from the beginning of the time period (e.g., 06:00 – 07:30 or 07:00 – 12:00 for the AM period), and chooses the incentive interval that achieves the highest performance when constrained on the past q' weekdays of trip data. Throughout, we will use the term incentive interval to refer to such a continuous time period with these discrete start/end times. In our results, q' is set to be 10. Thus, we let $D(d)$ denote the set of dates which are q' days in hindsight; we let $\tau(D(d), s)$ denote the possible morning incentive trips belonging to those dates and station s, and let $I_{d,s}$ to be the incentive interval for station s on day d. Furthermore, let us say trip $r \in I$ if and only if the trip occurred during the interval I. The interval $I^{\star}_{d,s}$ chosen by Static Hindsight for the morning period is

$$\arg\max_{I} \sum_{r \in \tau(D(d),s)} \Delta_r \cdot 1_{r \in I}, \tag{5.7}$$

where the notation $1_{r \in I}$ means the indicator function that is equal to 1 if $r \in I$, and is 0, otherwise.

Cluster Hindsight

This offline policy uses a clustering of the stations to help make incentive decisions. As we describe in more detail below, the model first groups the stations by station dock-capacity, clusters each group by bike-level behavior, and finally chooses the optimal incentive interval for each cluster when looking back in

hindsight (similar to Static Hindsight).

We compute groups of stations by sorting them from smallest to largest capacity, and dividing them into g roughly equal-size groups. The intuition for the grouping is our prior belief that station activity level is highly correlated with station capacity size and it is unwise, for example, to compare stations with bike-capacity 5 with stations of bike-capacity 60. The actual data-points used to cluster are 12-dimensional vectors consisting of the 12 half-hour interval bike-level percentages (bike-level divided by station dock capacity) for a station and date. The data points were obtained from our training data period, and bike-level percentages, rather than absolute values, were used to normalize the data points. To cluster each group of stations, we run the k-means clustering algorithm (k-centroids) with the objective of minimizing the distortion, which is defined to be the sum of the squared Euclidean distances between each vector and its centroid. Finally, our model uses a hard labeling system to associate each station with a single cluster (since for each station we have multiple points corresponding to distinct dates), and labels a station as belonging to the cluster for which its data points are most prevalent.

To calculate the actual incentive intervals for each cluster, the model finds the optimal single, continuous incentive interval when looking back in hindsight of q' weekdays for the stations in the cluster. Then let us define $C(s)$ to be the cluster of stations which s belongs to. The interval that the model chooses is as follows:

$$I^{\star}_{d,s} = \arg\max_{I} \sum_{s' \in C(s)} \sum_{r \in \tau(D(d),s')} \Delta_r \cdot 1_{r \in I} \tag{5.8}$$

The number of groups and the number of clusters are hyper-parameters of the model. To fit these parameters and avoid high bias/variance problems, k-fold

cross-validation was used with our training data, and the parameters with the best average scores were chosen. Due to the temporal aspect of the data (e.g., weather plays a big role in system behavior) the k-folds were also divided to maintain the temporal order. The final parameters used in our results are $g = 10$, $k = 3$ and $q' = 10$.

Fluid Model

A natural way to use the incentive-angel and non-angel rates to define an offline policy is by defining a so-called fluid model, in which it is assumed that exactly the expected number of rentals and returns occurs continuously per unit of time. Within such a model, it is easy to find the interval during which incentivizing minimizes the (fluid) number of out-of-stock events. In fact, one can show that under mild assumptions on the data, it is guaranteed that incentivizing over a single interval is optimal in such a model.

Though fluid models have successfully been applied to operational questions in bike-sharing, e.g., in Jian et al. [2016], we found in our analysis that the fluid model was vastly dominated by all other models; thus, we omit both its formal definition here and its performance in the results section.

Offline Benchmark

Offline policies are constrained in two ways: first, by the fact that we allow them to incentivize only during subintervals, as opposed to incentivizing, say, from 6-8AM and then again from 9-10AM but not in between; second, since the

decision is based solely on the number of bikes at each station at 6AM, they lack information about demand later in the course of the rush hour period. To distinguish between the lack of information and the constraint of incentivizing only during a subinterval, we define the following *static optimal* benchmark that has access to full information, but is constrained to incentivize only during a subinterval.

Static Optimal

The static optimal model is not a feasible policy since it assumes future knowledge that is not given in practice; instead, it is a model built in hindsight to benchmark offline policies. More specifically, for each week, station and AM/PM period, the Static Optimal model considers in hindsight, with complete knowledge, the best possible single continuous incentive period in which to incentivize.

Online Policies

In contrast to the offline policies described above, online policies gain information over the course of the rush hour period and thus are able to adopt whether or not to incentivize at a given station at a given time. We define only a very simple set of online policies.

Dynamic

The Dynamic model is a completely online policy that chooses in real-time for each trip whether or not it is incentivized which immediately implies perfect

efficiency. Formally, the model incentivizes all trips with $\Delta_r > 0$ (cf. Equation 5), i.e., the performance is exactly the sum of all trips that contribute positively to the objective. This policy serves as an upper-bound on what any policy can possibly achieve within our analysis.

Dynamic CC (X)

Beyond the fully dynamic model, we define a parameterized family of policies, wherein each policy breaks up the incentive period into intervals and the parameter X dictates the length of the intervals. For a given X, Dynamic CC starts at the beginning of each AM/PM period and decides every X minutes whether or not to incentivize trips for the next X minutes. To decide whether to incentivize the next X minutes, the model simulates the occurrence of one incentivized trip with the current number of bikes and empty docks at the station. If the simulated value of Δ_r is positive, Dynamic CC chooses to incentivize trips at the station for the next X minutes. For example, the Dynamic CC 15 policy at 6 AM will check the station's current number of available bikes and docks, and simulate an incentivized trip for that user dissatisfaction function and that number of bikes and docks available. If the Δ_r value for this trip is positive, it will choose to incentivize the station from 06:00–06:15. Then, at 06:15 it simulates a new trip, based on updated information, to determine whether or not to incentivize from 06:15–06:30.

We consider the Dynamic CC policy for $X \in \{15, 30, 60, 120\}$. Furthermore, one could also view the Dynamic model as Dynamic CC with $X = 0$.

5.4 Results

In this section, we present the performance of a number of policies when run with varying cost-parameter for both the AM and PM periods. The data-set for the test period, on which our analysis relies, consists of a total of 4944 trips in the AM and 2800 trips in the PM, across 147 stations. We focus our analysis on the deterministic regime in which we assume that an incentivized rental/return is always triggered by the incentive given and would not have occurred otherwise. Thereafter, we contrast those results with the ones in the stochastic regime. The scores for the former are displayed in Table 5.1 and for the latter in Table 5.2. All scores are given as the fraction of improvement of the optimal dynamic policy (i.e., each policy's absolute score is divided by the dynamic policy's absolute score). We begin by giving a high-level summary of our most interesting findings before providing more details for each of them.

Key Observations of Deterministic Results

Considering the rows indexed by Static for both AM and PM periods and both tables, it is noticeable that the static policy performs quite well, especially in the regime with low cost parameters, which is strong evidence of the operator's domain expertise in the initial choice of incentivized stations. Next, we observe the near-optimal performance of the Static Optimal benchmark across all cost parameters; this supports our decision to restrict the offline policies to incentivize each rush hour only over one sub-interval tailored for that station. For the dynamic policies, we observe a smooth decrease in performance as we transition from the maximally dynamic to more static policies, which highlights the trade-

Time	Policies	Cost Parameter				
		0.0	0.1	0.2	0.3	0.4
AM	Dynamic	1.0	1.0	1.0	1.0	1.0
	Stat. Opt.	0.985	0.981	0.976	0.97	0.961
	Static	0.939	0.911	0.870	0.809	0.715
	Stat. HS	0.960	0.947	0.929	0.905	0.873
	Clus. HS	0.961	0.944	0.918	0.887	0.845
	Dyn 15	1.0	1.0	1.0	1.0	1.0
	Dyn 30	0.995	0.994	0.993	0.992	0.990
	Dyn 60	0.989	0.984	0.980	0.977	0.969
	Dyn 120	0.967	0.953	0.944	0.940	0.924
PM	Dynamic	1.0	1.0	1.0	1.0	1.0
	Stat. Opt.	0.980	0.977	0.974	0.972	0.967
	Static	0.844	0.777	0.680	0.538	0.318
	Stat. HS	0.943	0.934	0.917	0.891	0.858
	Clus. HS	0.932	0.917	0.897	0.866	0.830
	Dyn 15	1.0	1.0	1.0	1.0	1.0
	Dyn 30	0.986	0.984	0.982	0.983	0.980
	Dyn 60	0.975	0.965	0.956	0.958	0.951
	Dyn 120	0.949	0.926	0.909	0.919	0.904

Table 5.1: Relative performances of each policy during AM and PM periods compared to the completely online policy under the deterministic performance evaluation.

offs between efficiency and simplicity in the policies. Noticeably, none of the offline policies handle high cost-parameters well. Finally, we observe a general decrease in performance from the AM to PM period, indicating that the system behavior is more erratic in the afternoon.

Static Performance

At first glance, the strong performance of the Static policy in Table 5.1 (0.939 in the AM without costs) may seem surprising. It is explained, however, by the operator's domain expertise when deciding which stations to include in the incentive scheme. Its degrading performance with increased costs is explained by

Period	Policies	Cost Parameter			
		0.0	0.01	0.1	0.2
AM	Dynamic	1.0	1.0	1.0	1.0
	Stat. Opt.	0.979	0.973	0.945	0.912
	Static	0.928	0.916	0.771	0.479
	Static HS	0.957	0.944	0.878	0.796
	Clus. HS	0.958	0.931	0.831	0.699
	Dyn 15	1.0	1.0	1.0	1.0
	Dyn 30	0.995	0.995	0.993	0.991
	Dyn 60	0.985	0.870	0.832	0.773
	Dyn 120	0.966	0.68	0.612	0.514
PM	Dynamic	1.0	1.0	1.0	1.0
	Stat. Opt.	0.978	0.971	0.943	0.917
	Static	0.923	0.903	0.646	0.15
	Stat. HS	0.967	0.948	0.883	0.784
	Clus. HS	0.958	0.935	0.839	0.682
	Dyn 15	1.0	1.0	1.0	1.0
	Dyn 30	0.992	0.990	0.992	0.99
	Dyn 60	0.982	0.889	0.850	0.807
	Dyn 120	0.960	0.823	0.756	0.702

Table 5.2: Relative performance of each policy during AM and PM periods compared to the completely online policy under the probabilistic performance evaluation.

the fact that the policy does not adapt to the shrinking set of trips with positive impact as costs increase.

Despite its reasonable performance in the AM in the regime without costs, the Static policy is still dominated by all policies across all cost parameters. This is especially prominent in the PM period, where differences range from 9% (Stat. HS) up to 15% (Dyn 15) even without cost parameters; the differences are even higher when accounting for costs. All of the incentivized trips in the PM, grouped by their improvement on the objective, δ_r, are visualized in Figure 5.3. Though the vast majority of incentivized trips have positive-impact, the other policies are able to accurately exclude those trips that do not, thus achieving

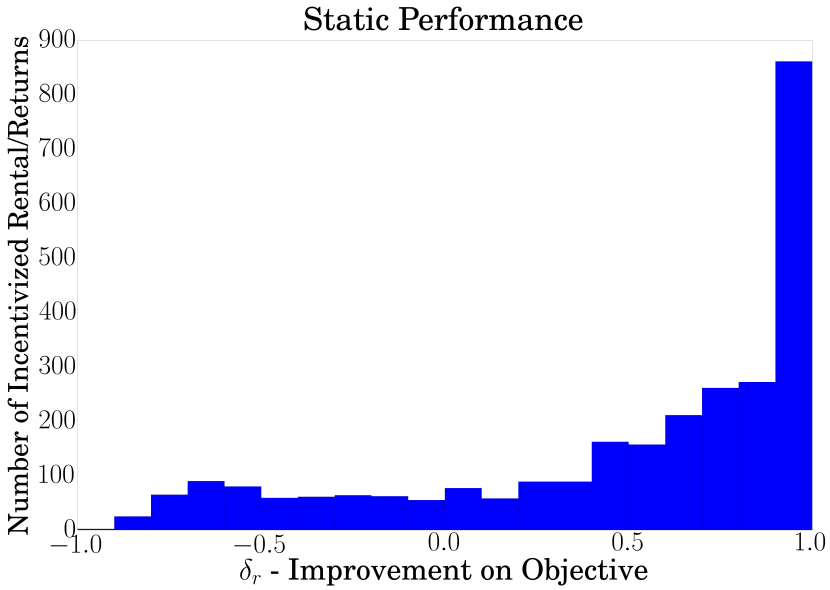


Figure 5.3: Static Policy's total number of incentivized trips, grouped by impact on objective (δ_r) for the PM period.

better performance. These results exemplify the importance of a data-driven approach to improve the Bike Angels program. For example, even a rather simple policy with minimal overhead, such as Static Hindsight, significantly improves the efficiency of the incentives.

Static Optimal Benchmark Performance

The online policies have two advantages over the offline policies: the flexibility to adapt decisions with updated information and the flexibility to incentivize over periods that are not sub-intervals. Comparing the performance of the Static Optimal to the Dynamic policy helps us distinguish between these two effects. More specifically, the benchmark operates with perfect information but is constrained to only incentivize over sub-intervals.

In Table 5.1 we do indeed observe Static Opt achieving a near optimal score of 0.98, which only slightly degrades with increased costs. This supports our assumption that it suffices to incentivize over a single sub-interval. However, Static Opt assumes perfect knowledge and still only matches the performance of Dynamic CC (60). In that sense, it also demonstrates the limitations of the best feasible offline policies.

Online to Offline Policies

Unsurprisingly, we find that the online policies outperform offline policies. As we transition from the maximally dynamic and online policy to the entirely static and offline policy, the decrease in performance occurs in a somewhat smooth way.

An interesting result in this context is the performance of the Dynamic CC (15) policy in Tables 5.1 and 5.2, as it achieves optimal scores for all sets of parameters; this demonstrates that making decisions completely online/in real time is not necessary to obtain perfect efficiency.

On the other end of the spectrum from online to offline, the results of Table 5.1 demonstrate that the offline policies Static Hindsight and Cluster Hindsight perform almost on par with the online Dynamic CC (120) policy, especially in the regime with low cost parameters; this points to the limited advantage of the simple online policies as the cost parameter increases.

The Cost Parameter

In considering the relative performance change of policies with increasing cost parameters, we find that the offline policies are comparatively much worse at handling high cost parameters than the online policies. Intuitively, it might seem that the interval incentive restrictions of offline policies leads to this result. For example, imagine taking the original incentive interval with no cost parameter, and changing some of the positive-impact trips within the interval to become negative-impact trips (due to increase in cost). Then unless these trips all exist at the outer limits of the interval, these trips will "shatter" the incentive interval: either the offline policies give up on the positive-impact trips at the beginning or they give up on the positive-impact trips at the end or they include the negative-impact trips in the middle. Online policies on the other hand can avoid this conundrum since they are not restricted to a single sub-interval.

However, the performance of the Static Optimal benchmark in Table 5.1 does not significantly degrade with high cost parameters. Thus, despite the interval restriction, the offline policies still have room to have better predictions yield improved performance.

In contrast, Figure 5.4 displays for the Dynamic CC (60) all incentivized trips with their time of day, their expected impact on future out-of-stock events, and for each one, the decision whether or not it is incentivized when run with 2 different cost parameters.The cost parameter is specified in each plot by the y-value of the horizontal line dividing the positive-impact trips (above the line) and negative-impact trips (below the line). As the cost parameter increases from 0.0 to 0.3, the policy excludes most trips having a δ_r value between that range, that it had previously included. Thereby, it manages to retain its near-optimal

performance.

AM and PM Periods

Comparing the policy performances from the AM to PM period, we find there is a significant performance difference (i.e., all policies perform worse in the PM). However, in light of the results in Table 1, we see the relative performance order of the policies with each other is consistent. This indicates that the system behavior of Citi Bike is fundamentally more erratic in the afternoon, leading to all policies having a harder time predicting when to incentivize.

Stochastic Evaluation

In this section we highlight four noticeable differences between the results in Tables 5.1 and 5.2.

Limiting Cost Parameters

When evaluating stochastic performances we limit the cost parameters to only go as high as 0.2. This is because when incorporating the likelihood of trips into the δ_r calculations, the expected impact of a trip before subtracting the cost is much lower; in particular, it is easy to show that the impact of a single incentivized rental/return is at best a reduction of 1 in the number of out-of-stock events. But then, if for example the probability of a particular rental having been triggered by an incentive was 0.5, then even with an impact of 0.8, there would be no improvement with cost parameter 0.4.

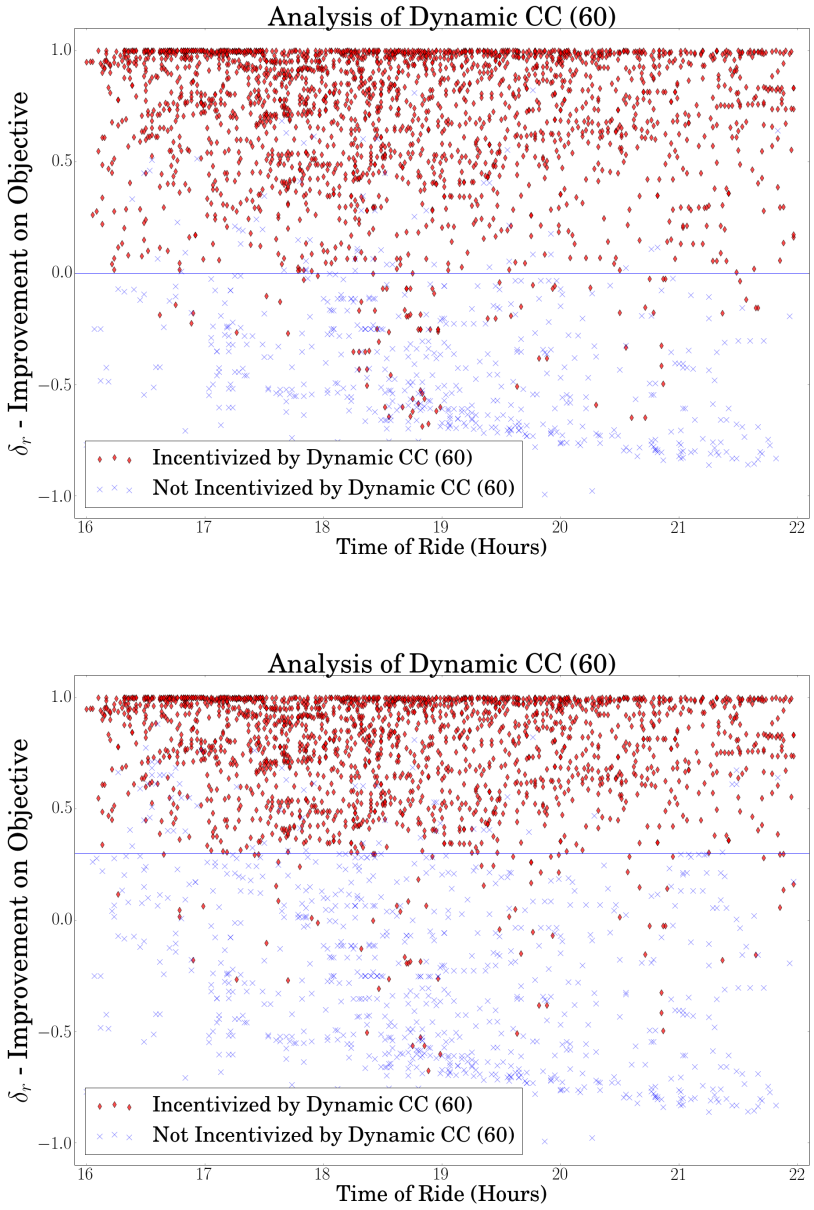


Figure 5.4: Scatter plots of incentivized trips indicating which trips are included/excluded in Dynamic CC (60) incentivization policy, when cost parameter is 0.0 (top) and 0.3 (bottom).

Relative Order

The results in Table 5.2 show that the same general trends in relative performance order (most dynamic to most static) that hold for the deterministic results, also hold for the stochastic results. Likewise, the results indicate that the Static Optimal benchmark still performs near optimally in the stochastic setting.

A key difference between the deterministic and stochastic results is the sensitivity to cost parameters. In particular, except for the Dynamic CC policies parameterized with 15 and 30, all policies decrease significantly in performance with even small increases in the cost parameter. Intuitively, due to the newly introduced stochasticity, the expected impacts of all rides are reduced, and small changes in cost are still large, relative to the reduced impacts.

Advantage of Hindsight Policies

Another interesting difference found between the deterministic and stochastic results is the relative performances of the Dynamic CC 60 and 120 policies compared to the Static Hindsight and Cluster Hindsight policies. Without stochasticity, the online policies dominated the offline policies in performance. However, when introducing stochasticity and for higher cost parameter regimes, the opposite seems to be true. Intuitively this makes sense. The Dynamic CC X policies consider only the status of the station at the beginning of each X-minute interval, which includes the probability of an incentivized rental/return having occurred due to the incentive at that time (thus ignoring the differing probabilities existing within the X-minute interval). In contrast, the Static Hindsight and Cluster Hindsight policies actually incorporate all probabilities when computing the optimal

incentive intervals retrospectively.

Static Policy

Finally, from the stochastic results we see an even greater difference in performance between the baseline Static policy and all other policies (especially with increasing costs). The large differences again underline the performance improvements that can be obtained by transitioning from the Static policy to a data-driven policy.

5.5 Conclusion

We have proposed a number of data-driven policies to guide incentives for rebalancing in bike-sharing systems via crowdsourcing. While our analysis clearly displays the performance differences between these policies, the superior performance of the more dynamic policies comes with the cost of greater complexity for users and operators alike.

There are other important considerations beyond their simplicity and performance. For example, when comparing the performance of the Static and Cluster Hindsight policies, it seems unclear at first glance what additional value the Cluster Hindsight policy provides, given that it relies on heavier machinery – after all, they perform very similarly. However, the Static Hindsight policy can only be defined for stations for which the Static policy had been in place, whereas the Cluster Hindsight policy can be defined for other stations as well. Thus, in a way, each of the policies presented has its own advantage.

Most importantly, our analysis shows that slightly limiting the online fashion of decision-making only causes limited decreases in performance. This adds a data-driven analysis to a recent stream of literature in operations management that compares dynamic and static decision-making in similar applications.

CHAPTER 6

REBALANCING

"No, I make nerds look good." — S. Sabbith

In this chapter, we report on a collaborative effort with Citi Bike to develop and implement data-driven tools to guide their rebalancing efforts. In particular, we provide new models to guide truck routing for overnight rebalancing and new optimization problems for non-motorized rebalancing efforts during the day. Finally, we evaluate how our practical methods have had an impact on Citi Bike's rebalancing in New York City. The chapter is based on [Freund, Norouzi-Fard, Paul, Henderson, and Shmoys, 2016].

6.1 Motivation

In the past few years, bike-share systems have developed different approaches to rebalancing. The most common approach employs trucks to move bikes to high-demand areas (cf. Chapter 2). This is particularly effective overnight, when both traffic and demand are low. During the day, vehicular traffic impairs these efforts, and operators supplement their *motorized* rebalancing through *non-motorized* rebalancing means such as *trikes* and *corrals*. A trike is a trailer that typically holds at most eighteen bikes and is towed by a cyclist to relocate bikes between a station A, with high supply, and a station B, with low supply (cf. Figure 6.1). A corral, on the other hand, artificially increases the capacity of a popular station by having an employee store bikes in between docks, thereby using all of the available space (cf. Figure 6.1).

This section is based on work by Freund, Norouzi-Fard, Paul, Henderson, and Shmoys 2016; we provide data-driven methods to help New York City Bike-share (NYCBS) improve overall utilization of their system and reduce customer dissatisfaction. We formulate and attack the underlying optimization problems that arise in truck-based rebalancing overnight, trike-based rebalancing during the day, and the placement of a limited number of seasonal corrals. For each of these settings we describe the methods developed and their impact for NYCBS.

First, we consider optimally routing trucks to relocate bikes overnight. This is called the *overnight rebalancing problem*. Our objective in this problem is to minimize expected customer dissatisfaction over the next day as measured by the user dissatisfaction function (cf. Chapter 3). We present an integer program (IP) that constructs routes for a given number of trucks and show how to find good solutions when given limited computation time in practice. Next, we consider the *mid-rush rebalancing problem*, studying how to optimally assign trikes to circulate between pairs of stations. Here, we use a maximum-weight k-edge matching to assign trike routes and maximize the impact on customer satisfaction. Finally, we consider how to optimally place corrals at stations where our goal is to minimize the number of customers who cannot find an open dock within a quarter mile of their preferred destination. We model this question as as a maximum coverage problem that we solve within seconds using a simple integer programming formulation.

The methods in this chapter made it to different stages of development at Citi Bike. We have completed trial runs using our overnight rebalancing schedules, routing three to four of their trucks over an eight-hour period. In the trials that we ran, our routes were able to improve the efficiency of NYCBS's truck fleet, as

measured by reduction in the user dissatisfaction function, by 12% on average – offline, the improvement we found was as high as 20%. NYCBS has also used our proposed placement of corrals during the 2016 summer season. Further, our method to match trikes to stations have suggested station pairs that inform their operational schedule.

This chapter thus summarizes our methodological contributions and their impact on day-to-day operations in New York City. The work described distinguishes itself from the existing literature in that it was conducted in close cooperation with NYCBS and already impacts their day-to-day operations. This stands in contrast to the findings of de Chardon et al. [2016], which conclude that very little of the existing work on rebalancing has had an impact in practice. Additionally, while there has been extensive work on overnight rebalancing, very little research has been conducted on non-motorized rebalancing efforts, which form a crucial part of NYCBS's operations. One such work is O'Mahony et al. [2016], which studies the routing of trikes; this paper partitions the set of stations into *producers*, which are stations likely to fill up, and *consumers*, likely to empty out. In investigating the problem of setting trike routes, the model proposed in O'Mahony et al. [2016] aims to minimize the distance of any consumer (producer) to another consumer (producer) that is rebalanced by one of the trikes. While our work on trikes is driven by the same application, our objective is again to minimize the user dissatisfaction function.

Figure 6.1: Pictures of a corral and a trike being used in NYC.

6.2 Overnight Rebalancing

Similar to the earlier work on overnight rebalancing, we use an IP formulation to model this optimization problem. The main distinguishing feature of our approach is that we greedily partition the problem into subproblems, thereby controlling the size of the IP. In particular, we find routes for subintervals of the entire time horizon, subsets of trucks, and subsets of stations. Our choice of stations is based on carefully weighing the potential benefits of rebalancing at each station and the distance of the station from the location of the truck at the beginning of the interval.

Integer Programming Formulation

We begin by introducing our integer program. The formulation is time-indexed, and we assume the given time for overnight rebalancing has been broken into T identical time steps. In each time step, a truck will either pick up/drop off bikes or move to an adjacent station. In that way, the edges between the stations are unweighted and traversing any edge takes one time step. In order to make

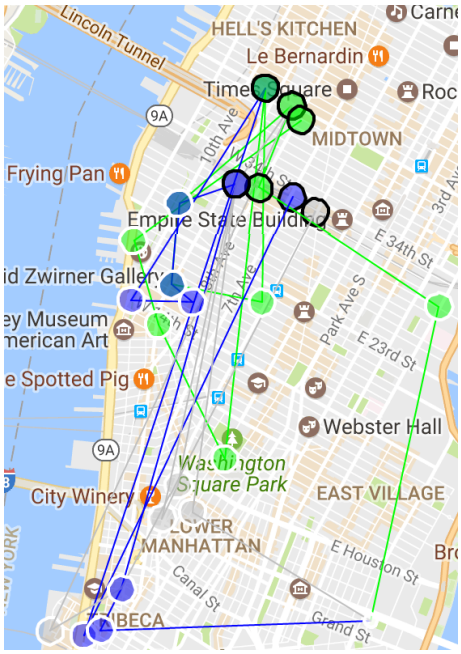


Figure 6.2: Truck routes for three trucks on August 8, 2016. Each circle corresponds to a station at which at least one of the trucks stops. A white outer circle corresponds to a pick-up, a black outer circle to a drop-off. Initially, all trucks start at a NYCBS depot in the East Village.

this assumption reasonable, we add dummy stations to break long distances into individual time steps. More precisely, if the travel time between two stations s_1 and s_2 is ℓ time steps, we add a path of $\ell - 1$ (dummy) stations between them, that allows us to move between such two stations in exactly ℓ time steps by traversing one edge in each step. This technique significantly reduces the dimension of the IP and distinguishes our formulation previous ones.

Notation.

After adding all dummy stations, let S be the set of stations, T be the number of time steps, and K be the number of trucks. For $s \in S$, $t \in [T]$, and $k \in [K]$, the variable x_{stk} represents whether or not truck k is at station s at time t. Similarly,

the variable y_{stk} represents the number of bikes at station s at time t to which truck k has access. This prevents multiple trucks from moving the same bikes. Lastly, the variable b_{tk} represents the number of bikes in truck k at time t.

We use the following notation:

- $N(s)$ denotes the neighborhood of s, that is, the stations to which a truck can move from s in a single time step.
- γ is the number of bikes that can be picked up or dropped off in one time step.
- $\text{start}(s)$ is the number of bikes in station s at time $t = 1$.
- $\min(s)$ is the minimizer of the user dissatisfaction function at station s. That is, the number of bikes at station s that minimizes the expected number of dissatisfied customers.
- $c_s = \frac{c_s(\text{start}(s)) - c_s(\min(s))}{|\text{start}(s) - \min(s)|}$ is a linear approximation of the slope of c_s and gives the improvement per bike moved at s (see Figure 6.3, Chapter 3).
- S_+ is the set of stations s for which $\text{start}(s) > \min(s)$. For example, the state in Figure 6.3 is in S_+.
- S_- is the set of stations s for which $\text{start}(s) \leq \min(s)$.

For ease of presentation, we state here only the main constraints of the IP before we explain the effect of each. The reader is advised to read the IP in parallel with the explanations below.

$$\text{maximize}_{x,y,b} \sum_{s\in S, k\in[K]} (y_{s1k} - y_{sTk})c_s$$

subject to:

$$x_{stk} \leq x_{s(t-1)k} + \sum_{s':s\in N(s')} x_{s'(t-1)k}, \qquad \forall s,t,k; \tag{6.1}$$

$$\sum_{s\in S} x_{stk} = 1, \qquad \forall t,k; \tag{6.2}$$

$$\sum_{k\in[K]} y_{s1k} = \text{start}(s), \qquad \forall s; \tag{6.3}$$

$$\text{start}(s) \leq \sum_{k\in[K]} y_{stk} \leq \min(s), \qquad \forall s \in S_-, t; \tag{6.4}$$

$$\min(s) \leq \sum_{k\in[K]} y_{stk} \leq \text{start}(s), \qquad \forall s \in S_+, t; \tag{6.5}$$

$$\sum_{s\in S} y_{stk} + b_{tk} = \sum_{s\in S} y_{s1k} + b_{1k}, \qquad \forall t,k \tag{6.6}$$

$$|y_{stk} - y_{s(t-1)k}| \leq \gamma x_{stk}, \qquad \forall s,t,k; \tag{6.7}$$

$$|y_{stk} - y_{s(t-1)k}| + \gamma|x_{stk} - x_{s(t-1)k}| \leq \gamma, \qquad \forall s,t,k. \tag{6.8}$$

Below we explain the function of each part of the IP.

- The objective function is the summation of changes in the linearized user dissatisfaction functions at each station, i.e. the reduction in expected number of dissatisfied customers due to the relocation of bikes.
- **Constraint (6.1)** allows each truck to move only to a station adjacent to the one at which it currently is.
- **Constraint (6.2)** indicates that at each time step, each truck must be in exactly one station.
- **Constraint (6.3)** initiates the number of bikes at every station. Notice that at this point already, the bikes are distributed among the K trucks.

- **Constraints (6.4)** and **(6.5)** guarantee that the number of bikes in each station s remains between start(s) and the minimizer min(s). In other words, we enforce that moving a bike only improves the setup (cf. *Pareto Constraints and Optimal Fleet Size*).
- **Constraint (6.6)** enforces that the total number of bikes in the system does not change over time.
- **Constraint (6.7)** makes sure that we pick/drop bikes at a station from a truck only if the truck is at that station and that the number of bikes moved is bounded by γ, the number of bikes rebalancers are able to move within one period.
- **Constraint (6.8)** ensures the truck either moves or picks/drops bikes in one time step but not both. In most of the previous works, researchers have omitted this constraint. This constraint makes the IP significantly harder to solve but makes the resulting path viable in practice. Notice that the absolute values in the constraints can be linearized.

Moreover, we add capacities to the truck by bounding b_{tk}. In practice, we extend this IP to fix the starting/finishing stations for each truck, as well as the number of bikes in each truck at the beginning of the night.

Pareto Constraints and Optimal Fleet Size

Notice that the extension of the linearization of $c_s(\cdot)$ beyond the point min(s) does not capture the actual behavior of the UDF. In particular, at that point the latter sees more dissatisfied customers while the former sees fewer. This, however, is not the reason we impose the *Pareto constraints* (6.4) and (6.5), since optimizing

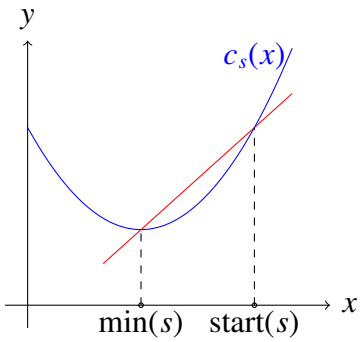


Figure 6.3: Linearization of $c_s(\cdot)$.

over the linear envelope of $c_s(\cdot)$ is not significantly harder than over the linearization; instead, the constraints are imposed to ensure that service for customers at one station is not sacrificed for improved service for customers elsewhere. This is particularly true when, as is the case in NYC, $\sum_s \text{start}(s) << \sum_s \min(s)$. At the time we started the pilots, the bike fleet size was about 7.000 whereas the minimizers summed to about 10.000. In such a setting, not having fairness constraints can yield undesirable outcomes. For instance, without fairness constraints we might find solutions in which bikes can be picked up from a station far away that has more bikes than needed but are instead picked up from a station nearby that has fewer than needed. While this may lead to fewer dissatisfied customers in total, NYCBS aims for rebalancing to be a Pareto improvement to the system (i.e., no station is worse off, some are improved); thus, we include constraints (6.4) and (6.5).

Solution Methods

As presented, state-of-the-art IP solvers are too slow to solve this formulation within the limited time window between when the operator receives the data and when the trucks must start their routes a few minutes later. The follow-

ing heuristic methods help decrease the computation time to solve this IP and improve the quality of the solution returned.

Reducing the number of edges. To avoid adding too many dummy stations and inflating the size of the IP, we choose a threshold d on the distance between two stations and only add a path between stations s_1 and s_2 if they are at most d time steps apart.

Dividing T into smaller time intervals. At the start of the rebalancing period, there is little time (typically about 20 minutes) between the time all the data (state of the system, number of bikes on each truck, etc.) becomes available and the time when trucks are meant to begin their routes. To gain computation time, we break T into smaller intervals and only route trucks for the first few hours of the route. While this part of the route is being executed, we then use the time to solve for the next interval. This segmentation of the computation time greatly improved the quality of our overall routes.

Greedily selecting stations. For each time interval, we further reduce the size of the IP by removing stations where the room for improvement is low. We rank stations based on a combination of c_s, the potential benefit of each bike picked or dropped, and $|\min(s) - \text{start}(s)|$, the number of available/required bikes, and run the IP with roughly 40 stations, further refined by excluding stations too far away from the starting point of the truck.

Splitting trucks. Instead of solving one IP for all K trucks, we break the computation time into K equal pieces and solve for the route of the first truck, then the second truck, and so forth. For example, if we have two trucks and two hours of computation time, we would solve for the route of each

truck in one hour.

To further decrease the size of the IP, we only add stations in reasonable proximity to the current position of the truck. We then compute the path over one time interval and update the set of stations. On the one hand, our IP can find very good solutions for the smaller instances on which we solve. On the other hand, by picking stations in the described greedy fashion, we ensure that the combination of the solutions to the small instances has objective close to the global optimum. The routes we construct for each truck and time interval are compatible in that they can be pieced together to form one coherent route. We show in our results that these heuristics still yield good solutions to the original IP.

Results

In this section, we summarize our results. First, we report the gap between the solution we return using the techniques above and an optimal solution of a valid LP relaxation. Second, we compare our solutions to the current routes employed by NYCBS, as guided by tools based on our earlier work. On average, our solutions reduce customer dissatisfaction by 20% compared to this previous computationally informed approach. All results were produced using Gurobi v6.5 on a machine with 8GB RAM and an Intel i7-2600 processor with 3.4GHZ.

IP gap

360 Stations		
# of Trucks	Avg Objective Function	Avg Gap
1	148	28.1%
2	232	22.1%
3	301	12.3%
4	330	9.5%

Figure 6.4: *A posteriori* optimization for overnight truck rebalancing in Manhattan.

When solving the IP, we assume the number of time steps is 60, each time step corresponds to 6 minutes, and that workers can load/unload up to 7 bikes in each time step. These numbers are based on discussions with NYCBS. To evaluate the performance of our IP, we used our IP to route various numbers of trucks in Manhattan, which has around 360 stations. As inputs for start(s), we used the number of bikes at each station at midnight over the course of a week. In Figure 6.4, we report the average impact on the number of dissatisfied customers and the average integrality gap over the week. The worst integrality gap occurred Sunday night when the distribution of bikes was furthest from the commuter demand during the week. We emphasize that this gap is computed with respect to the LP-relaxation on all 360 stations, all trucks, and all time-steps, i.e. the LP-relaxation of the provided integer program; hence, it includes the gap between the solution we find for the smaller problem instances (on 30-40 stations) and the optimum of these smaller instances.

The average integrality gap decreases as the number of trucks grows. This seems to be a consequence of the Pareto constraints, since there are fewer bikes available to be moved (per truck). With more trucks, it is easier to achieve the best possible solution. We remark that a much smaller integrality gap can be

obtained without constraints (6.7) and (6.8) as has been done in Raviv et al. [2013], Ho and Szeto [2014], and Forma et al. [2015]. The solutions obtained this way, however, allow for as many stops (with loading/unloading of few bikes at each) as there are time periods; in discussions with NYCBS we found that in many cases that the time required to find a place to park the truck is not sufficiently dominated by driving/(un)loading time that it could be ignored. We thus had to add these constraints, even though it makes the IP much harder to solve.

Practical Results

Dispatchers at NYCBS currently use a myopic decision aid to route trucks. This aid is based on user dissatisfaction functions and was developed in close cooperation with our group. While this decision aid shows dispatchers the optimal fill level at stations and indicates stations where rebalancing could yield large improvements, it does not provide optimized routes. In contrast, our IP solutions look to globally optimize routes. We formulated our model with feedback from NYCBS; their expertise led, for example, to the refinement that in each time step a truck can either move or pick/drop bikes but not both. Over the course of a week in July 2016, we then compared our proposed routes with the manual routes executed by the dispatchers at NYCBS. On average our results showed objectives about 20% higher. Together with the NYCBS management, we reviewed our proposed routes and are now running pilots to route between 2 and 5 of their trucks overnight.

Pilot Experiences

Despite the improvements, the pilots conducted with NYCBS proved to be more difficult than expected, as we quickly describe here.

Unknown constraints. In an aborted first attempt at a pilot, we routed a truck into a narrow street, in which it did not fit. While this is a rare event, it can happen and significantly complicates the routing problem. To deal with this problem, we included in subsequent pilots alternative stations (non-optimized) at which each truck could stop without interfering with the routes of the other trucks.

Cost of solving offline. Solving the system offline can have detrimental consequences when unexpected demand occurs late at night. While the system significantly slows down at night, we have encountered cases where our route dictated picking up some number of bikes from a station and by the time the truck had arrived there, fewer than that were left. Similarly, it may occur that the Pareto constraint is violated in the execution of the route because the number of bikes at a station changes between the time we solve and the time the truck arrives. To avoid this occurring regularly, we excluded certain areas that tend to slow down later than others (e.g., East Village) in the first part of the route.

Evaluating results. In evaluating our results, we found that certain system conditions greatly influence the efficiency of rebalancing. On nights when the system is in great imbalance, it is not uncommon for each truck to move enough bikes to reduce the expected number of dissatisfied customers by > 100 users. In contrast, on nights where the system is already reasonably

balanced, it is often not feasible for a truck to move bikes to reduce the objective by > 50.

Although the final point indicates that evaluating results is difficult, the improvements through rebalancing on the nights during which we ran this pilot were on average 12% higher than in the rest of that month.

6.3 Trikes

In this section, we study the impact of trikes on the expected number of dissatisfied users and solve a corresponding optimization problem. Recall that a trike is a trailer towed by a cyclist that holds at most 18 bikes at a time (cf. Figure 6.1), though, when we did this work with Citi Bike they could hold at most 5. Citi Bike uses trikes between fixed pairs of stations to move bikes from high-demand, low-supply stations to low-demand, high-supply stations. NYCBS considers trikes a preferred choice of rebalancing during rush hour when trucks are slowed down by traffic. We first assume that every station may have only one trike route incident to it. In this regime, the problem of finding the optimal m trike routes can be formulated as a bipartite maximum m-edge matching, where the weight of an edge between two stations corresponds to the reduction in user dissatisfaction with a trike added between them. Next, we generalize these ideas to the case where a station can utilize multiple trikes. With a slight variation, we show that this as well as can be formulated as a matching problem and efficiently solved.

Model

Similar to the user dissatisfaction function defined in Chapter 3, we model the *user dissatisfaction function with a trike* between stations A and B as follows: Poisson processes with rates $\lambda_A, \mu_A, \lambda_B, \mu_B$ correspond to arrivals and departures of users at station A and B, respectively. We use the random variable $Y_s(t) \in \{0, \ldots, k_s\}$ to represent the number of bikes at station $s \in \{A, B\}$ at time t, where k_s is the capacity of station $s \in \{A, B\}$. As before, a dissatisfied customer at s corresponds to an arrival (resp. departure) when s is full (resp. empty), i.e., $Y_s(t) = k_s$ (resp. $Y_s(t) = 0$).

In addition to arrivals and departures of users, we also have a trike with capacity k_R that moves as many bikes as possible from A to B. We assume that at times $t_1, \ldots, t_r$ the trike stops at one of the stations. Without loss of generality, the stop at t_i is at A if i is odd and at B if it is even. In other words, the trike cycles back and forth between A and B. When the trike arrives at station A it picks up as many bikes as possible given the number of bikes at A and the number of bikes already in the trike; similarly, when the trike arrives at station B it drops off as many bikes as possible given the number of available docks at B and the number of bikes already in the trike. We use the random variable $Y_R(t) \in \{0, \ldots, k_R\}$ to represent the number of bikes in the trike at time t.

We are interested in the expected number of dissatisfied users at A and B over the time horizon from t_0 to t_{r+1}. We can write the expected number of dissatisfied

users at stations A and B as follows:

$$c_{A\to B} = \sum_{i=0}^{r}\sum_{j=0}^{k_A} \Pr(Y_A(t_i^+) = j)c_A^i(j) + \sum_{i=0}^{r}\sum_{j=0}^{k_B} \Pr(Y_B(t_i^+) = j)c_B^i(j)$$

The two terms correspond to the expected number of dissatisfied users at each station; to obtain them, notice that at the beginning of time-interval i with probability $\Pr(Y_s(t_i^+) = j)$ there are j bikes at station s and in that case an expected $c_s^i(j)$ users will be dissatisfied in that interval. Thus, we need to find $\Pr(Y_s(t_i^+) = j)\ \forall s, i, j$. For ease of notation, we denote $\Pr(Y_A(t_i^+) = \alpha, X_B(t_i^+) = \beta, X_R(t_i^+) = \rho)$ as $\pi^i(\alpha, \beta, \rho)$ and remark that by setting $Y_R(t_0^+) = 0$ with probability 1 and assuming that $X_A(t_0), X_B(t_0)$ are independent we obtain:

$$\pi^0(\alpha,\beta,\rho) = \begin{cases} 0, \text{ if } \rho > 0 \\ \Pr(X_A(t_0^+) = \alpha) \cdot \Pr(X_B(t_0^+) = \beta), \text{ else.} \end{cases}$$

This will represent the base case to recursively compute $F_{A\to B}$.

To calculate $\pi^i(\alpha, \beta, \rho)$ in general we need to analyze the system changes in each time interval. Specifically, given the Poisson process rates and the current number of bikes x at a station $s \in \{A, B\}$, we can compute the expected number of dissatisfied customers $c_s^i(\cdot)$ in an interval (t_i, t_{i+1}). In doing so, we also obtain the probability that there are y bikes in station s at the start of the next time interval. More precisely, $\forall s \in \{A, B\}, x, y \in \{0, \ldots, k_s\}$ we let

$$P_{x,y}^{s,i} := \Pr(Y_s(t_{i+1}^-) = y | Y_s(t_i^+) = x),$$

where $Y_s(t^+) := \lim_{\epsilon\to 0^+} Y_s(t+\epsilon)$, i.e., $Y_s(t_i^+)$ is the number of bikes at s just after the trailer has stopped at the station and $Y_s(t^-) := \lim_{\epsilon\to 0^+} Y_s(t-\epsilon)$ is the number of bikes at s just before the trailer has stopped at the station.

Thus, for even i, we have $\pi^{i+1}(\alpha,\beta,\rho) = 0$ if $\alpha > 0$ and $\rho < k_R$, as otherwise the trike would pick up more bikes. Otherwise, we obtain

$$\pi^{i+1}(\alpha,\beta,\rho) = \sum_{x=0}^{k_A}\sum_{y=0}^{k_B}\sum_{z=0}^{\rho}\pi^i(x,y,z)P^{A,i}_{x,\alpha+\rho-z}P^{B,i}_{y,\beta},$$

where we define $P^{s,i}_{x,y} = 0$ for $y \notin \{0,\ldots,k_s\}$. This is because, with $Y_R(t_i^+) = z$, the event that $\{Y_A(t_{i+1}^+) = \alpha$ and $Y_R(t_{i+1}^+) = \rho\}$ happens if and only if the number of bikes at A before the pick-up at t_{i+1} is $\alpha + \rho - z$.

Similarly, for odd i, we have $\pi^{i+1}(\alpha,\beta,\rho) = 0$ if $\beta < k_B$ and $\rho > 0$. Otherwise,

$$\pi^{i+1}(\alpha,\beta,\rho) = \sum_{x=0}^{k_A}\sum_{y=0}^{k_B}\sum_{z=\rho}^{k_R}\pi^i(x,y,z)P^{A,i}_{x,\alpha}P^{B,i}_{y,\beta+\rho-z}.$$

Recognizing that

$$\Pr(Y_A(t_i^+) = j) = \textstyle\sum_{y=0}^{k_B}\sum_{z=0}^{k_R}\pi^i(j,y,z) \quad \text{and}$$
$$\Pr(Y_B(t_i^+) = j) = \textstyle\sum_{x=0}^{k_A}\sum_{z=0}^{k_R}\pi^i(x,j,z),$$

we can now compute all $\Pr(Y_s(t_i^+) = j)$ and $\pi^i(\alpha,\beta,\rho)$ recursively starting with the base cases for $i = 0$ and incrementing i. Thus, we can efficiently compute $c_{A\to B}$.

Given $c_{A\to B}$ for each pair of stations, we use these values to assign the trike routes. To formulate the problem as a maximum m-edge matching problem, we create a bipartite graph $G = (X \cup Y, E)$ where $X = Y$ is the set of stations. We set the weight of edge (A, B) equal to the difference of the expected number of dissatisfied users at stations A and B without the trike, given by

$$\sum_{j=0}^{k_A}\Pr(Y_A(t_0) = j)F_A(j) + \sum_{j=0}^{k_B}\Pr(Y_B(t_0) = j)F_B(j),$$

and the expected number with the trike, $F_{A\to B}$. Thus, the weight of any matching is equal to the reduction in the expected number of dissatisfied customers from

the corresponding trike routes and vice versa. Further, there is a well-known and efficient algorithm for finding the maximum m-edge matching for any bipartite graph.

Relaxed Assumptions

If we allowed more than one trike incident to a particular station, the dimensionality of the dynamic program explodes, yielding it infeasible. A slight variation of the model, however, allows us to model this problem as a maximum-weight matching as well. Instead of coupling the random variables of numerous stations and trikes, we estimate for each station how many bikes trikes can pick up/drop off bikes over the time horizon in which they operate. To find a solution for k trikes we then create for each station s nodes $s_1, \ldots s_k$ and create an edge (A_i, B_j) with weight set to be the improvement at station A through the ith additional trike and at B through the jth additional trike. Notice that, in contrast to the earlier formulation, this does not incorporate the chance that a trike incident to station A does not have an effect because there are no bikes to pick up at the adjacent station B.

Results

Setting $k_R = 5$ and eight trike stops between 7:30 AM and 9:15 AM (thus moving a total of at most 20 bikes per station pair), we use the above dynamic program to compute the weight of the maximum matchings of various sizes; in Figure 6.5 we display the maximum matching of size 8, in Figure 6.6 the weight of all maximum matchings of size 1 through 20. Notice that for up to 5 trailers, there is

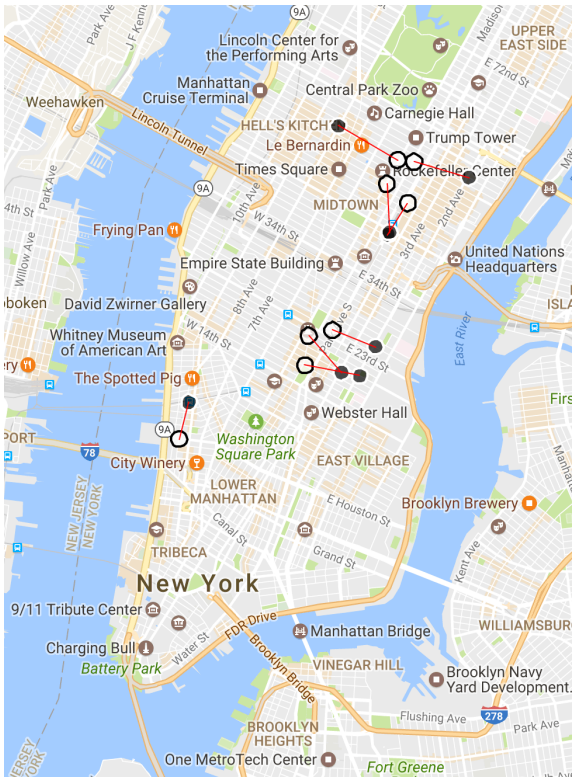


Figure 6.5: Trike routes identified by the maximum-weight matching formulation. Red lines indicate trikes that pick up bikes at white circles and drop them off at black ones. (Map data: Google Maps)

an improvement of about 20 (fewer dissatisfied customers). While these numbers can be viewed in comparison to the numbers in Figure 6.4, the costs of operating a truck are significantly higher than those of a trailer. The relative cost of trucks and trailers thus affect which of the two (truck/trailer/both) is more efficient to improve service quality.

6.4 Corrals

Our work on corrals is motivated by studies on the correlation between distance to transportation modes and willingness to use these modes. A study in Regional

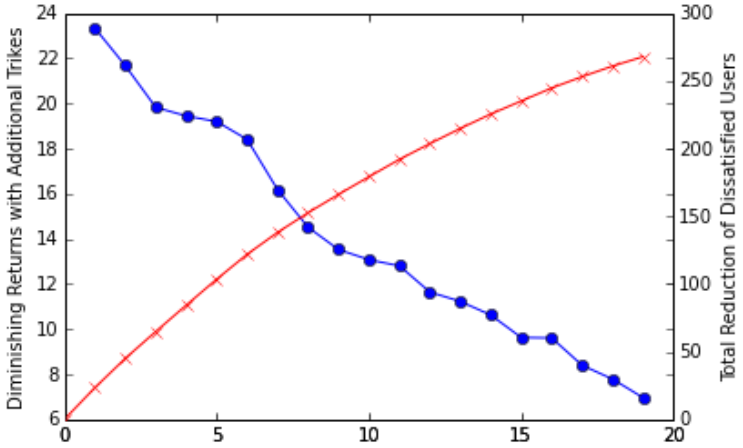


Figure 6.6: Total improvement (red) of trikes and corresponding diminishing returns (blue).

Plan Association [1997], for example, claims that commuters are much more likely to use public transportation when living within a quarter mile of a station than when further away. Furthermore, Kabra et al. [2015] shows that shorter distances to available stations correlates with increased demand for bike-share systems.

Model

Based on the findings described before, we say a station has a *shortage* if no station within a quarter mile has at least 15% of its docks available. In other words, a station is in shortage if a user intending to end their trip at that station will likely have to search more than a quarter mile away to find an available dock. Thus, stations in shortage significantly impact customer utilization.

We define a shortage measure for the system given by the total time stations are in shortage. Formally, let $N(s) \subseteq S$ be the set of neighboring stations within a quarter mile of s, including s itself, and let $A_{s,t}$ the indicator of the event that

station s has at least 15% of its docks available at time t. For a set of stations S and a set of points in time $\mathcal{T}$, we define the shortage measure as

$$w(S, \mathcal{T}) = \sum_{s \in S} \sum_{t \in \mathcal{T}} \max\{0, 1 - \sum_{j \in N(s)} A_{j,t}\}.$$

It is possible to extend this measure to a weighted version that associates a time-dependent coefficient based on dock demand to each station. Since the unweighted version was the one that informed decision-making for NYCBS, we restrict ourselves to that.

By placing a corral at a station s, the amount of time that stations in $N(s)$ are in shortage is significantly reduced. Given a budget B, the goal is to place at most B corrals to minimize the shortage measure. For a set of past time points in $\mathcal{T}'$, let

$$w_s = \sum_{t \in \mathcal{T}'} \max\{0, 1 - \sum_{j \in N(s)} A_{j,t}\}.$$

This represents the amount of time that station s is in shortage and is equivalent to the reduction in the shortage measure for s if a station in $N(s)$ is assigned a corral. It is then natural to model corral placement as a maximum coverage problem via the following IP:

$$\begin{aligned} \text{maximize}_{x,y} \quad & \sum_{s \in S} w_s x_s \\ \text{subject to:} \quad & x_s \leq \sum_{s' \in N(s)} y_{s'} \quad \forall s \\ & \sum_{s \in S} y_s \leq B \\ & x_s, y_s \in \{0, 1\} \quad \forall s. \end{aligned}$$

In the given IP, the variable y_s represents whether or not a corral is placed at

station s and the variable x_s represents whether or not there is a corral placed within a quarter mile from station s. The first constraint sets x_s to be 0 if no corral is assigned in $N(s)$, and the second constraint corresponds to the constraint that we can assign at most B corrals.

Results

The evaluation of our work in this section is two-fold. Given the real implementation of our suggested corrals by Citi Bike, we are able to observe the change in shortages from 2015 (without corrals) to 2016 (with corrals). However, we cannot be certain this corresponds 1-to-1 with a decrease in unsatisfied customers. Since the presence/absence of these is in general difficult to measure, we also use a discrete-event simulation to estimate the reduction in dissatisfied customers through our choice of corrals.

Our discrete-event simulation is based on Poisson arrivals with fixed destinations at each station. In contrast to the user dissatisfaction function described in Chapter 3, the discrete-event simulation captures interdependencies between stations. For instance, customers who do not find a dock at a particular station, roam around to nearby stations until eventually they do. For details of the implementation, we refer the reader to Jian et al. [2016] and O'Mahony [2015]; we use the same simulation with the addition of allowing for some stations to have temporarily increased capacity between 7.30AM and 5.30PM.

Simulated Impact on Unsatisfied Customers

To compare the performance of our coverage formulation, we consider several allocations of corrals (cf. Figure 6.7):

1. *coverage '15/'16* are the allocations of corrals obtained by solving the coverage problem using data from 10 days in June 2015, and June 2016 respectively.
2. *k-median* is based only on the biking distance between separate stations. Using the Google Maps API, we obtained pairwise distances and then solved the resulting k-median instance.
3. *max minutes* relies only on the number of minutes a station had no docks available in the month of May 2016.
4. *None* allocates no corrals at all.

Our results are visualized in Figure 6.7. Based on data from June 2016, we simulated 40 realizations of demand and computed, using common random numbers, for each realization and each allocation of corrals, the number of unsatisfied customers. Notice first that adding corrals generally improves the objective (though corrals can also increase the expected number of dissatisfied customers and sometimes do, as the plot shows). Moreover, the solutions obtained from the coverage formulation (run both with data from 2015 and with more current data from 2016) outperforms the solutions that are based only on geography (*k-median*) or only on the number of full minutes (*max minutes*): on average the two coverage solutions observe 304 and 162 fewer unsatisfied customers, compared to just 4 fewer for the k-median solution.; the max minutes solutions ranks in the middle with an average reduction of 77.

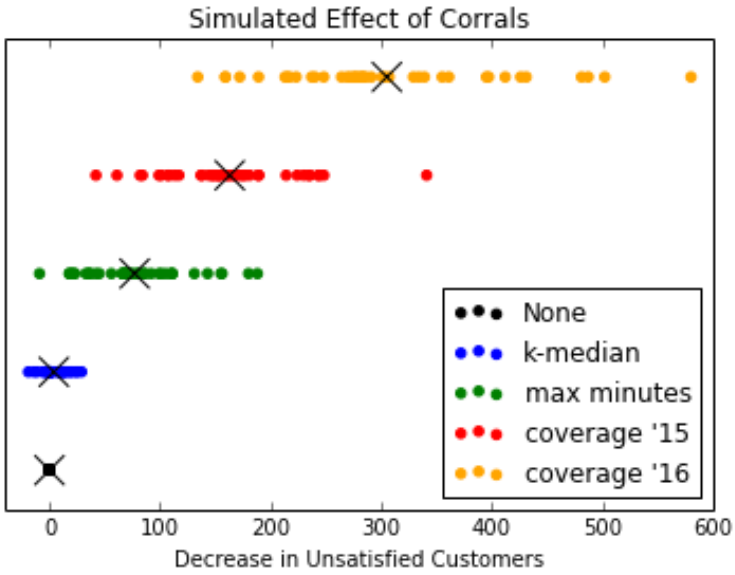


Figure 6.7: Results of 40 simulated days with different sets of corrals; × denotes the average performance.

Observed Real Impact

Six corrals identified by the maximum coverage formulation (see Figure 6.8) with $B = 6$ have been in place for most of Summer 2016. To evaluate the benefit of these corrals, we first consider the value of the shortage measure in July 2016 (with the corrals) and in July 2015 (without). As the size of the system has expanded significantly, we restricted the shortage measure to include only stations S in Manhattan that were in use for most of July 2015 and July 2016. For reference, $|S| = 277$. We find a 31.6% reduction in the time that stations were in shortage. Further, to show that the majority of the improvement is indeed due to the effects of the corrals, we also calculate the shortage measure in July 2015 and July 2016 for $\bar{S}$ which excludes the stations with corrals from S. This should give an idea as to how much the shortage measure has decreased through other rebalancing efforts. Here, we only find a 14.6% improvement which shows that

the majority of the improvement was due to the added corrals. Results of these analyses are summarized in Figure 6.9.

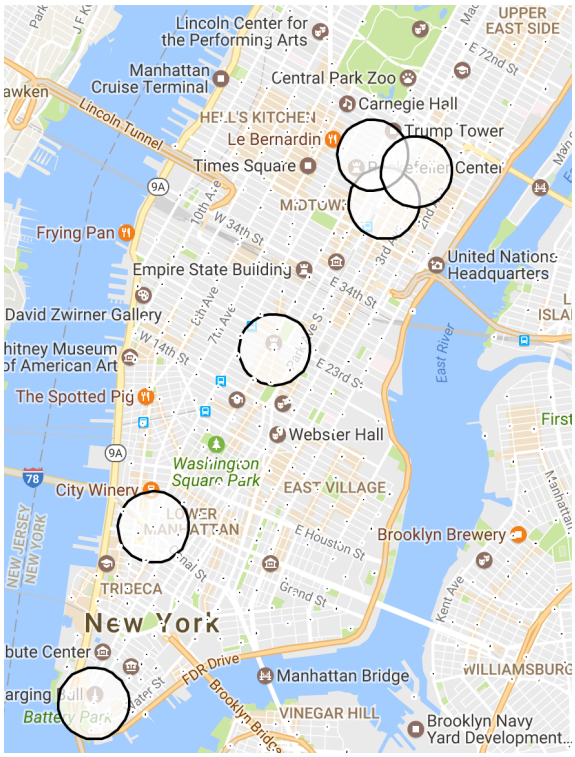


Figure 6.8: Corral stations in NYC with $\frac{1}{4}$-mile radius (Map data: Google Maps).

	S	$\bar{S}$
July 2015	49435	56989
July 2016	33804	48657
Percent Reduction	31.6%	14.6%

Figure 6.9: Shortage measure for July 2015 and July 2016.

CHAPTER 7

SCHEDULING MAINTENANCE

Nina: Now you seem lonely and broken to me.

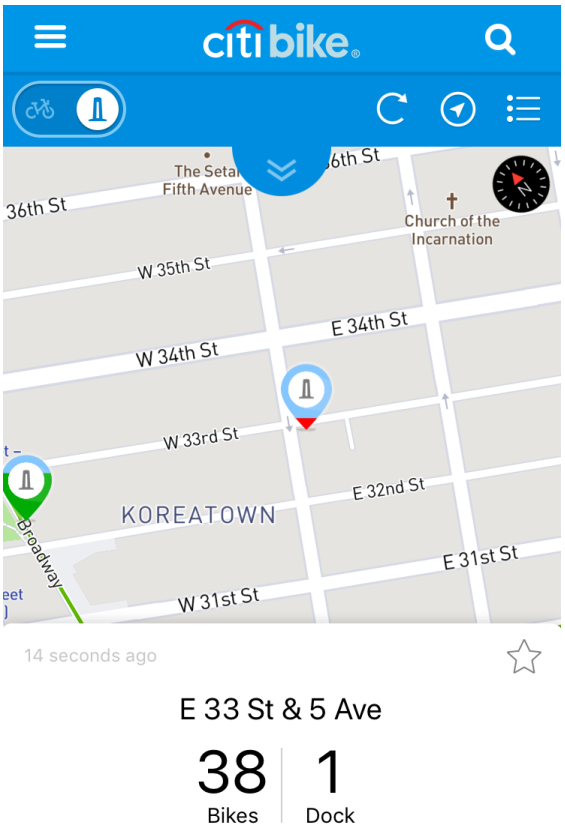
Will: I do?

Nina: But don't worry I can fix you.

Most modern bike-sharing systems have mobile apps that allow users to observe the current system state, i.e., number of bikes and number of available empty docks, at each station. The information on these apps, however, can be misleading when maintenance issues arise. In particular, empty docks are sometimes reported as available even though, in fact, maintenance issues cause them to be unavailable. This can cause great customer dissatisfaction when all empty dock(s) at a station are not usable. For example, Figure 7.1 displays a station with one available dock – if that dock was broken, arriving customers would not be able to return their bikes, despite the app reporting that a dock is available.[1]

The effect of a broken dock is thus amplified at a station where it is particularly likely that this dock would be the last empty dock at the station. It is thus natural to ask two questions to improve the operations of bike-sharing systems: first, can we identify which docks are broken and second, how should the operator schedule maintenance given knowledge of the location of broken docks. We answered the first question by implementing a system for New York City Bikeshare that automatically identifies broken docks based on data, similar to Kaspi et al. [2016]. In response to the second question, we consider in this chapter

[1]The data at that station on that day, clearly indicates that the dock was not in fact broken, as it saw repeated use.



[h]

Figure 7.1: Screenshot from the Citi Bike app, taken on December 19th, 2017, indicating that the station at E 33 St & 5 Ave had one empty dock available at the time.

a stylized version that models the routing question as a variation of the traveling salesman problem. In this version, the goal is to maximize the reduction in dissatisfied customers caused by broken docks by appropriately scheduling a repairman over a finite time horizon to fix docks. The work in this chapter appeared as part of a paper by Paul, Freund, Ferber, Shmoys, and Williamson [2017].

7.1 Problem Definition

In the classical traveling salesman problem, we are given an undirected graph $G = (V, E)$ with an edge cost $c_e \geq 0$ for each $e \in E$. The goal is to construct a tour visiting each vertex in the graph while minimizing the cost of edges in the tour. If, however, we are given a bound on the cost of the tour, then we might not be able to visit all vertices. In particular, suppose that we are given a budget $D \geq 0$ and a weight w_v for each $v \in V$. In the *budgeted prize-collecting traveling salesman problem*, a valid tour is a multiset of edges F such that (a) F specifies a tour on a subset $S \subseteq V$ and (b) the cost of the edges in F is at most D. The goal is to find a valid tour F that maximizes $\sum_{v \in S} w_v$, the weight of all vertices visited. Here, we do not require the graph to be complete and allow a tour to visit nodes more than once. Similarly, in the *budgeted prize-collecting minimum spanning tree problem*, a valid tree is a set of edges T such that (a) T specifies a spanning tree on a subset $S \subseteq V$ and (b) the cost of the edges in T is at most D. Again, the goal is to find a valid tree T that maximizes $\sum_{v \in S} w_v$.

The budgeted version of the traveling salesman problem arises naturally in many routing problems that have a distance or time constraint. For example, in the maintenance application we consider here, D corresponds to the duration of the repairman's shift, whereas w_v corresponds to the decrease in out-of-stock events due to a dock being repaired. For example, a natural such weight would correspond to the increase in the long-run average cost at the station when the station size is reduced by 1. Throughout most of this chapter, we focus on the tour version in a setting in which $w_v = 1$ for all $v \in V$. In Section 7.6 we extend our algorithm to handle settings in which vertices have weights and the goal is to maximize the weight of vertices visited; in that section we also explain how

the analysis extends to the tree version. In Section 7.7, we apply our algorithm to instances using Citi Bike data in New York City.

Our algorithm provides a 2-approximation guarantee for both problems. It is based on a primal-dual subroutine which uses a linear programming relaxation of this problem. First, we search for a "good" value for the dual variable corresponding to the budget constraint in the primal. Having set this variable, we can then increase the other dual variables and form a forest of edges whose corresponding dual constraint is tight. For the tour problem, we then choose a tree in this forest and carefully prune it so that doubling this tree forms a tour that will be just within budget. For the tree problem, we prune edges such that the tree itself is just within budget. Lastly, we show that either our constructed tour/tree is within a factor of 2 of optimal or we can identify a subgraph that contains the optimal solution and that we can recurse on.

Literature Review

Before we delve into the technical details of this section, we give a brief overview of the existing work in combinatorial optimization related to the budgeted prize-collecting minimum spanning tree/tour problems. Many prize-collecting variants of both the traveling salesman problem (TSP) and the minimum spanning tree problem (MST) seek to balance the number of vertices in the tree or tour with the cost of edges used. Johnson et al. [2000] characterize four main variants of prize-collecting MST problems: the Goemans-Williamson Minimization problem that minimizes the cost of edges plus a penalty for vertices not in the tree, the Net Worth Maximization problem that maximizes the weight of vertices in the tree minus the cost of used edges, the Quota problem that minimizes the cost of a tree

containing at least Q vertices, and, finally, the Budget problem that maximizes the number of vertices in the tree subject to the cost of the tree being at most D. All of the variants above can be extended to a corresponding TSP version that constructs a tour rather than a tree.

Our algorithm is most similar to that of Garg [2005], who presents a 2-approximation algorithm for the Quota problem for MST, improving upon the previous results of Garg [1996], Arya and Ramesh [1998], and Blum et al. [1996]. Johnson et al. [2000] observe that a 2-approximation algorithm to the Quota problem yields a $(3 + \epsilon)$-approximation algorithm to the corresponding Budget problem. To our knowledge this was the previously best-known guarantee for the MST variant. Prior to this result, Levin [2004] proved a $(4 + \epsilon)$-approximation algorithm. Our 2-approximation algorithm for the budgeted prize-collecting MST thus improves upon the best known approximation ratio. While our algorithm is similar to that of Garg [2005], our analysis differs in how we find the threshold value for the dual variable; further, our overall proof relies on more precise accounting.

For the Goemans-Williamson Minimization problem for MST, Archer et al. [2011] obtain a $(2 - \epsilon)$-approximation guarantee, improving upon the long-standing bound of 2 obtained by Goemans and Williamson [1995]. Further, Archer et al. [2011] successfully applied this algorithm to telecommunication network problems. Lastly, Feigenbaum et al. [2001] show the Net Worth Maximization problem for MST is NP-hard to approximate within any constant.

To the best of our knowledge, the previous best approximation guarantee for the budgeted prize-collecting TSP arises from a special case of a result by Chekuri et al. [2012]. Their work provides a $(2 + \epsilon)$-approximation algorithm for

the more general orienteering problem, where the goal is to find an $s - t$ path, where s and t are given, with bounded cost that maximizes the number of vertices visited on the path. By setting $s = t$ and iterating over all vertices, this yields a $(2 + \epsilon)$-approximation algorithm for the budgeted prize-collecting TSP. The orienteering problem itself has attracted much attention within the combinatorial optimization community, with other variants studied by Vidyarthi and Shukla [2015], Chekuri and Korula [2007], Chen and Har-Peled [2006], Chekuri and Pal [2005], and Gupta et al. [2012].

There exist other adaptations of prize-collecting problems not discussed above. Specifically, Ausiello et al. [2004] present a 2-approximation algorithm for an on-line variant of the Quota problem for the TSP. Frederickson and Wittman [2012] study the so-called traveling repairmen problem, in which each vertex can only be visited within a specific time window and the goal is to either maximize the number of vertices visited within a certain time period or to minimize the time visiting all vertices; they give constant-factor approximation algorithms for both variations of this problem. Lastly, Nagarajan and Ravi [2012] study the problem of minimizing the number of tours to cover all vertices subject to each tour having bounded distance. They give a 2-approximation algorithm for tree metric distances.

The remainder of the chapter is structured as follows. In Section 7.2, we present the linear programming (LP) relaxation for the budgeted prize-collecting traveling salesman problem. In Section 7.3, we use this LP to develop the primal-dual subroutine that will inform our decisions. In Section 7.4, we use this subroutine to present an outline of the entire parameterized primal-dual algorithm and the proof of its approximation ratio, providing some intuition behind what

types of tours will be near optimal. In Section 7.5, we prove an upper bound on the size of an optimal solution. Next, we show how to set the dual variable corresponding to the budget constraint and how to construct our proposed tour that is within a factor of 2 of optimal. For ease of presentation, we present these results for the budgeted prize-collecting traveling salesman problem with unit weights and show how the analysis extends to the weighted case and the MST version in Section 7.6. Last, we present computational experiments in Section 7.7.

7.2 LP Formulation

For each $S \subseteq V$, we denote by $\delta(S)$ the edges in the cut of S and by $z_S \in \{0, 1\}$ a variable representing whether or not the vertices in S are the ones on which the tour is constructed; for each edge $e \in E$, we let $x_e \in \mathbb{Z}^+$ be a variable representing how many copies of e to include in the tour. Then, the following is a linear programming relaxation for the budgeted prize-collecting traveling salesman problem.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{S \subseteq V} |S| z_S \\
\text{subject to} \quad & \sum_{e : e \in \delta(S)} x_e \geq 2 \sum_{T : S \subsetneq T} z_T \quad \forall S \subsetneq V \\
& \sum_{e \in E} c_e x_e \leq D \\
& \sum_{S \subseteq V} z_S \leq 1 \\
& z_S, x_e \geq 0
\end{aligned}
$$

The first constraint states that if the tour visits the vertices in a subset T and $S \subsetneq T$ then we must have at least two edges across the cut S. The dual of this linear program is given by the following.

$$\begin{aligned}
&\text{minimize } \Lambda_1 D + \Lambda_2 \\
&\text{subject to } (2 \sum_{T:T \subsetneq S} y_T) + \Lambda_2 \geq |S| \quad \forall S \subseteq V \\
&\qquad \sum_{S:e \in \delta(S)} y_S \leq \Lambda_1 c_e \quad \forall e \in E \\
&\qquad \Lambda_1, \Lambda_2, y_S \geq 0
\end{aligned}$$

In order to construct a tour, we rely on a primal-dual subroutine. We first note that if we find $\Lambda_1 \geq 0$ and $y_S \geq 0$ that satisfy the dual constraint for each edge, then we can always set Λ_2 to be the maximum of 0 and $\lambda_2 := \max_{S \subseteq V} \left[|S| - (2 \sum_{T:T \subsetneq S} y_T) \right]$ so that we have a feasible dual solution. Suppose that we first set the value of Λ_1. The primal-dual subroutine uses this value to construct a full dual solution and corresponding potential tours. These tours might or might not be feasible with respect to the budget constraint. Therefore, we need to adjust Λ_1 to find a feasible solution with bounded approximation ratio.

7.3 Primal-Dual Subroutine

The primal-dual subroutine for a fixed Λ_1 is similar to the 2-approximation algorithm for the prize-collecting traveling salesman problem without a budget constraint presented by Goemans and Williamson [1995]. Similar to Goemans and Williamson, we define a potential of a set S as a function of the dual variables

of the strict subsets of S.

Definition 15. *For any subset $S \subseteq V$, we define the **potential** of S to be*

$$\pi(S) := |S| - (2 \sum_{T:T \subsetneq S} y_T).$$

Definition 16. *A subset $S \subseteq V$ is **neutral** if $2 \sum_{T:T \subseteq S} y_T = |S|$. In other words, if $y_S = \frac{1}{2}\pi(S)$.*

At the beginning of the primal-dual subroutine, we set each y_S to be 0 and set the collection of active sets to be all singleton nodes. Further, we set $T = \emptyset$. Then, in each iteration, we increase y_S corresponding to each $S \subseteq V$ in the collection of active sets until either a dual constraint for an edge between two sets becomes tight, or a set becomes neutral. If an edge becomes tight between two subsets S_1 and S_2, we add the edge to T and replace both S_1 and S_2 in the collection of active sets by $S_1 \cup S_2$. We remark that the potential of this new set S is then equal to

$$\pi(S) = \pi(S_1) + \pi(S_2) - 2y_{S_1} - 2y_{S_2}.$$

If instead a set becomes neutral, we mark it inactive and remove it from the collection of active sets. Once there are no more active sets, we prune inactive sets of degree 1 (cf. Algorithm 1).

Since sets are removed from the set of active subsets when an incident edge becomes tight, the dual constraint for every edge will remain satisfied throughout the subroutine. Thus, λ_1 and the y_S satisfy the dual constraints for every edge and we can extend them to a feasible dual solution $(y, \lambda_1, \lambda_2)$. Further, by construction, the dual constraint of every $e \in T$ is tight and the edges in the set T, and thus also those in T', form a forest throughout the subroutine. Finally, beyond being part of

a feasible solution, the construction of y also guarantees that $\mathcal{S} = \{S : y_S > 0\}$, the collection of all sets active at some point throughout the subroutine, is laminar. Indeed, each set in $\mathcal{S}$, other than singletons, is the union of two disjoint other sets in $\mathcal{S}$. We denote by $\mathcal{S}^+ = \mathcal{S} \cup \{V\}$, which is also laminar.[2]

Algorithm 1: Primal-Dual Algorithm (PD(λ_1))

$\forall S \subseteq V y_S \leftarrow 0$, $\Lambda_1 \leftarrow \lambda_1$, $T \leftarrow \{\}$.
mark each $i \in V$ as active.
while there exists an active subset **do**
  raise y_S uniformly for all active subsets S until either
  if an active set S becomes neutral **then**
    mark S as inactive.
  else if the dual constraint for edge e between S_1 and S_2 becomes tight **then**
    $T \leftarrow T \cup \{e\}$.
    mark $S = S_1 \cup S_2$ as active, remove S_1 and S_2 from the active subsets.
  end if
end while
$T' \leftarrow T$.
while there exists a set S marked inactive such that $|\delta(S) \cap T'| = 1$ **do**
  remove all edges with at least one endpoint in S from T'.
end while
return two of each edge in T'.

---

[2] A collection of sets is called *laminar* if any two sets in the collection are either disjoint or subsets of each other.

7.4 Main Result

In this section, we give an outline of how the parameterized primal-dual algorithm uses the primal-dual subroutine described in the previous section to construct a feasible tour with bounded approximation guarantee. Assume that we have run the primal-dual subroutine with $\Lambda_1 = \lambda_1$ to find a feasible dual solution $(y, \lambda_1, \lambda_2)$, where we may not know the actual value of λ_2, along with an accompanying forest of tight edges. The corresponding potentials will help us give an upper bound on the size of an optimal solution and a lower bound on the size of our returned solution in order to provide an overall approximation guarantee. Let $O^\star$ be the subset of vertices visited by an optimal tour and $F^\star$ be the edges in that tour. Further, let O be the minimal set in $\mathcal{S}^+$ that contains $O^\star$. Since $V \in \mathcal{S}^+$, such a set always exists. In Section 7.5 we prove the following bound on the size of $O^\star$.

Theorem 17.

$$|O^\star| \leq \lambda_1 D + \pi(O).$$

Next, we show how to set λ_1 so that we can construct a tree T_A of tight edges on a subset S_A with cost $\sum_{e \in T_A} c_e \leq \frac{1}{2}D$. Doubling the edges in T_A and shortcutting produces a tour on S_A that has length at most D. Further, if the cost of T_A is $< \frac{1}{2}D$, we show that we can find an incident edge $\bar{e}$ such that adding $\bar{e}$ to T_A creates a tree $\bar{T}$ on a subset $\bar{S}$ with cost $\sum_{e \in \bar{T}} c_e > \frac{1}{2}D$. In other words, adding one more edge to T_A makes doubling it infeasible. If instead the cost of T_A is equal to $\frac{1}{2}D$, then we simply let $\bar{T} = T_A$ and $\bar{S} = S_A$. Let Q be the maximum potential set in $\mathcal{S}^+$ containing $\bar{S}$. Then, we prove the following bound on $|S_A|$.

Theorem 18.

$$|S_A| > \frac{1}{2}\lambda_1 D + \pi(Q) - 1.$$

Thus, the parameterized primal-dual algorithm can find a feasible tour on a subset S_A where $|S_A|$ is bounded using the potential $\pi(Q)$. If $\pi(Q)$ is large, this implies that we have found a large subset our tour can visit. In particular, if $\pi(Q) \geq \pi(O)$, then

$$|S_A| + 1 > \frac{1}{2}[\lambda_1 D + \pi(O)] \geq \frac{1}{2}|O^\star|.$$

We may assume without loss of generality that $|O^\star|$ is even, since we can always make a copy of each vertex that has an edge of cost zero incident to the original. Hence, the above implies that if $\pi(Q) \geq \pi(O)$, then $|S_A| \geq \frac{1}{2}|O^\star|$.

However, it may be that $\pi(Q) < \pi(O)$. Therefore, the parameterized primal-dual algorithm finds all maximal sets in $\mathcal{S}^+$ with potential strictly greater than $\pi(Q)$ and recurses on the graph induced by each such set, returning the largest feasible tour found. If $\pi(Q) < \pi(O)$, then O must be a subset of one of the sets on which we recurse. In other words, one of the subgraphs we recurse on contains the optimal solution. Since none of the subsets on which we recurse contains $\bar{S}$, by definition of Q as the maximum potential set containing $\bar{S}$, we recurse only on strict subgraphs of G, implying that we eventually reach the case that $\pi(O) \leq \pi(Q)$ and $|S_A| \geq \frac{1}{2}O^\star$, yielding the 2-approximation guarantee.

Finally, since $\mathcal{S}$ is a laminar family, we recurse on disjoint sets of vertices, so the set of graphs on which we recurse is also a laminar family and we call the subroutine $O(n)$ times.

Theorem 19. *The parameterized primal-dual algorithm is a 2-approximation for the budgeted prize-collecting traveling salesman problem.*

7.5 Upper Bound

In this section, we provide the proof of Theorem 17. Recall that O is the minimal set in $\mathcal{S}^+$ that contains the optimal subset $O^\star$ and $F^\star$ is an optimal tour on $O^\star$. Assume that we have run the primal-dual subroutine with $\Lambda_1 = \lambda_1$ to obtain a dual solution $(y, \lambda_1, \lambda_2)$. Theorem 17 states that $|O^*| \leq \lambda_1 D + \pi(O)$. The proof relies on the feasibility of the dual solution along with the following lemma.

Lemma 20. *For each $S \subseteq V$, $(2 \sum_{T:T\subseteq S} y_T) \leq |S|$.*

Proof. Any set S can be partitioned into maximal disjoint laminar subsets $S_1, S_2, \ldots, S_c \in \mathcal{S}$. Therefore,

$$2 \sum_{T:T\subseteq S} y_T = 2 \sum_{i=1}^{c} \sum_{T:T\subseteq S_i} y_T \leq \sum_{i=1}^{c} |S_i| = |S|,$$

where the inequality holds since neutral subsets are marked inactive. ■

Proof of Theorem 17. We first note that we can partition the powerset of O into subsets of $O - O^\star$ and subsets that contain vertices in $O^\star$.

$$2 \sum_{T:T\subsetneq O} y_T = 2 \sum_{T:T\subseteq O-O^\star} y_T + 2 \sum_{\substack{T:T\subsetneq O \\ T\cap O^\star \neq \emptyset}} y_T. \tag{7.1}$$

Thus,

$$\begin{aligned}
|O| &= 2 \sum_{T : T \subsetneq O} y_T + \pi(O) \\
&= 2 \sum_{T : T \subseteq O - O_\star} y_T + 2 \sum_{\substack{T : T \subsetneq O \\ T \cap O^\star \neq \emptyset}} y_T + \pi(O) \\
&\leq |O - O^\star| + 2 \sum_{\substack{T : T \subsetneq O \\ T \cap O^\star \neq \emptyset}} y_T + \pi(O),
\end{aligned}$$

where the first line holds by the definition of potentials, the second by Equation (7.1), and the last line by Lemma 20. Rearranging, we get

$$\begin{aligned}
|O^\star| &\leq 2 \sum_{\substack{T : T \subsetneq O \\ T \cap O^\star \neq \emptyset}} y_T + \pi(O) \\
&\leq \sum_{e \in F^\star} \sum_{T : e \in \delta(T)} y_T + \pi(O) \\
&\leq \lambda_1 \sum_{e \in F^\star} c_e + \pi(O) \\
&\leq \lambda_1 D + \pi(O).
\end{aligned}$$

The second line holds since $F^\star$ is a tour on $O^\star$ and so, by the minimality of O, each subset T with $y_T > 0$ that contains a subset of $O^\star$ has at least two edges in its cut $\delta(T)$. The third line holds by the dual feasibility of $(y, \lambda_1, \lambda_2)$, and the last line holds by the primal feasibility of $F^\star$. ■

Setting Λ_1

We now turn our attention to finding λ_1 and constructing a feasible tour. Our goal is to set Λ_1 for the primal-dual subroutine so as to find a tree with cost very close to $\frac{1}{2}D$ and such that the set of spanned vertices has high potential. Note that

Λ_1 controls the cost of the edges, and as Λ_1 increases, edges become more expensive yielding smaller connected components in the primal-dual subroutine. In particular, for $\Lambda_1 = 0$ all edges go tight immediately and for $\Lambda_1 > n/(2 \min_{e:c_e>0} c_e)$ all vertices go neutral before a single non-zero edge goes tight. When edges go tight and subsets go neutral at the same time, we may assume that subset events are considered first. Further, we assume that we break edge/subset ties using some known ordering (e.g., lexicographical).

If a minimum spanning tree on G has cost $\leq \frac{1}{2}D$, then we double this tree to get a feasible and optimal tour. Otherwise, suppose that we have found values l and r ($l < r$) such that when we run PD(l^+) the largest component in T' has cost $\geq \frac{1}{2}D$ and when we run PD(r^-) the largest component in T' has cost $< \frac{1}{2}D$. Here, $x^- = x - \varepsilon$ and $x^+ = x + \varepsilon$, where ε is infinitesimally small.

Lemma 21. *In polynomial time, we can find a threshold value λ_1 such that when we run PD(λ_1^-) the largest component in T' has cost $\geq \frac{1}{2}D$ and when we run PD(λ_1^+) the largest component in T' has cost $< \frac{1}{2}D$.*

Proof. We refer to an edge going tight during the primal-dual subroutine as an edge event and we refer to a subset going neutral as a subset event. Assume we have values l and r such that the first k events are the same when running the subroutine for any Λ_1 between l^+ and r^-. Further, assume that for each subset S we can find values α_S and β_S such that at the end of the first k events $y_S = \Lambda_1 \alpha_S + \beta_S$ for any Λ_1 between l^+ and r^-. Note that this is trivially true for the base case with l and r defined above and $k = 0$ since all y values will be zero.

To find the next event to occur, we need to find the time after the kth event that each subset will go neutral and each edge will go tight. Observe that an

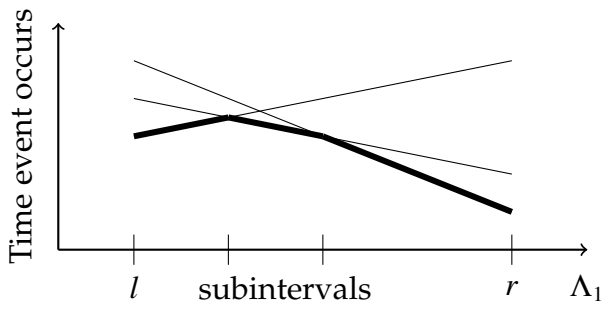


Figure 7.2: Finding the subintervals between l and r where the time of the next event is in bold.

active set S will go neutral at time

$$\frac{1}{2}|S| - \sum_{T \subseteq S} y_T = \frac{1}{2}|S| - \sum_{T \subseteq S} [\Lambda_1 \alpha_T + \beta_T],$$

an edge with exactly one endpoint in an active component will go tight at time

$$\Lambda_1 c_e - \sum_{T : e \in \delta(T)} y_T = \Lambda_1 c_e - \sum_{T : e \in \delta(T)} [\Lambda_1 \alpha_T + \beta_T],$$

and an edge with both endpoints in different active components will go tight at time $\frac{1}{2}$ the above amount. The minimum of these values determines the next event to occur. Since all of these times are affine in Λ_1, we can divide the interval between l^+ and r^- into smaller subintervals such that the first $k+1$ events are identical on these subintervals. See Figure 7.2.

Considering these subintervals, we either identify a threshold point λ_1 or we identify a subinterval between l^+_{new} and r^-_{new} such that when we run PD(l^+_{new}) the largest component in T' has cost $\geq \frac{1}{2}D$ and when we run PD(r^-_{new}) the largest component in T' has cost $< \frac{1}{2}D$. Further, since the time of the $(k+1)$th event is an affine function in Λ_1, we can add this function to the affine function $y(S)$ for each active set S to get the new affine function for this y value, updating the α's and β's accordingly. Thus, the inductive hypothesis holds and eventually we find a threshold point λ_1. ■

We use this threshold point λ_1 to understand the subroutine for PD(λ_1). Consider running the subroutine for λ_1^+ and λ_1^- and comparing event by event. We let y^+ correspond to the y variables when running PD(λ_1^+) and let y^- correspond to the y variables when running PD(λ_1^-).

Lemma 22. *Throughout the two subroutines, the following two properties hold:*

- *All active components in* (V, T) *are the same.*
- *For all* $S \subseteq V$*, the difference between* y_S^+ *and* y_S^- *is infinitesimally small.*

Proof. At the start of the subroutines this is true since all y^+ and y^- variables are zero. Now assume that it is true at some time t into the subroutines. As argued above, the next event to occur depends on the minimum of functions linear in Λ_1. Further, since the current active components are the same, the possible subset and edge events are the same.

In particular, the time for each subset to go neutral in PD(λ_1^+) is $\frac{1}{2}|S| - \sum_{T \subseteq S} y_T^+$, which is infinitesimally different from the time for that subset to go neutral in PD(λ_1^-). Similarly, the time for each edge to go tight is at most infinitesimally different between the two subroutines. Therefore, the next event to occur is only different between the two subroutines if two events occur at the same time for PD(λ_1).

If the next event is the same for the two subroutines, then the active components remain the same and we raise all active components by amounts differing by an infinitesimally different amount. Therefore, the inductive properties continues to hold. Otherwise, suppose the next event is different. We consider four cases:

1. Subset X goes neutral for PD(λ_1^-) and subset Y goes neutral for PD(λ_1^+).
2. Edge e goes tight for PD(λ_1^-) and edge f goes tight for PD(λ_1^+).
3. Edge e goes tight for PD(λ_1^-) and subset X goes neutral for PD(λ_1^+).
4. Subset X goes neutral for PD(λ_1^-) and edge e goes tight for PD(λ_1^+).

In the first case, the times for both X and Y to go neutral must be infinitesimally different and the other subset goes neutral immediately after the first. Therefore, once both X and Y are neutral, the difference of the amounts by which we have raised the y variables in PD(λ_1^-) and in PD(λ_1^+) is infinitesimally small and the current active components are the same. Thus, the two inductive properties continue to hold.

Similarly for the second case, if e and f are not between the same two components, the other edge goes tight immediately after, and the inductive properties continue to hold. Otherwise, e and f are between the same components. Thus, when e goes tight, f is no longer eligible to go tight but the newly merged active component is the same for both subroutines. Again, the inductive properties continue to hold.

In the third case, if edge e has an endpoint in an active component that is not X, then e goes tight immediately after X goes neutral for PD(λ_1^+) and the components remain the same, maintaining the inductive properties. Otherwise, one endpoint of e must be in X and the other endpoint of e is in an inactive component, and right after e goes tight for PD(λ_1^-), the newly merged subset has infinitesimally little potential left and goes inactive immediately thereafter. Again, this maintains the inductive properties.

Lastly, note that the time for a subset to go neutral has a negative slope in Λ_1

and the time for an edge to go tight has a positive slope in Λ_1. Since $\lambda_1^+ > \lambda_1^-$ and the y variables vary by an infinitesimally small amount, the fourth case cannot occur. In all cases, the inductive properties continue to hold, which concludes the proof of the lemma. ■

The proof of Lemma 22 exactly exhibits the differences between the two subroutines. First, there may be subsets that are neutral and marked inactive in PD(λ_1^+) but have infinitesimally small potential in PD(λ_1^-). Second, there may be pairs of edges that went tight between the same components. Lastly, there may be edges in PD(λ_1^-) that do not exist in PD(λ_1^+). However, these edges are between inactive components and components with infinitesimally small potential. Therefore, these edges are pruned in PD(λ_1^-) and do not contribute to the component of size $\geq \frac{1}{2}D$.

Assume that we run PD(λ_1) breaking event ties to behave the same as PD(λ_1^+). Then, the largest component in T' when running PD(λ_1) has cost $< \frac{1}{2}D$. However, we can think about reversing these ties one by one. In particular, consider breaking the first i ties according to PD(λ_1^-) and then the rest by PD(λ_1^+). By the analysis in Lemma 22, reversing these ties changes the y variables only by an infinitesimally small amount and does not at all affect the active components. The only difference occurs when entering the pruning phrase.

Thus, eventually we find the smallest k such that breaking the first k ties according to PD(λ_1^-) yields a component of size $\geq \frac{1}{2}D$. In other words, we have either identified a neutral subset S such that marking S active rather than inactive changes the largest component to have size $\geq \frac{1}{2}D$ or we have identified two edges e and f that tie such that adding e instead of f changes the largest component to have size $\geq \frac{1}{2}D$. From here on, we assume that we always run PD(λ_1) according

to these tie-breaking rules.

Constructing a Tour

Continuing from the previous section, let y be the dual variables at the end of running PD(λ_1), let T' be the set of edges after the pruning phase, and let $\mathcal{S} = \{S : y_S > 0\}$ be defined as before. Lastly, let $\pi(S)$ be the potential of $S \subseteq V$ given y. By construction, the largest component returned by PD(λ_1) has size $\geq \frac{1}{2}D$. Recall that either

1. there exists a neutral subset $X \in \mathcal{S}$ such that if X is marked inactive then the largest component in T' has cost $< \frac{1}{2}D$ or
2. there exist tight edges $e \in T$ and $f \notin T$ such that if we swap e with f in T then the largest component has size $< \frac{1}{2}D$.

In the first case, when X is marked inactive, then a path of neutral subsets $N_1, N_2, \ldots, N_r = X$ is pruned yielding a component S_1 with cost $< \frac{1}{2}D$. Similarly, in the second case, edge e prevented some neutral subsets $N_1, N_2, \ldots, N_r$ from being pruned that had degree > 1. However, by removing e and replacing it with f, these subsets are pruned and we are left with component S_1 with cost $< \frac{1}{2}D$. See Figures 7.3 and 7.4.

For both cases, we use this threshold event to produce a tree T_A on a subset of vertices S_A of cost $\leq \frac{1}{2}D$. In doing so, we also find another tree $\bar{T}$ on a subset of vertices $\bar{S}$ of cost $\geq \frac{1}{2}D$ such that $|S_A| \geq |\bar{S}| - 1$. Then, doubling T_A yields a feasible tour F_A that visits at most one node less than there are in $\bar{S}$. The tree $\bar{T}$ will be helpful in obtaining a lower bound for $|S_A|$.

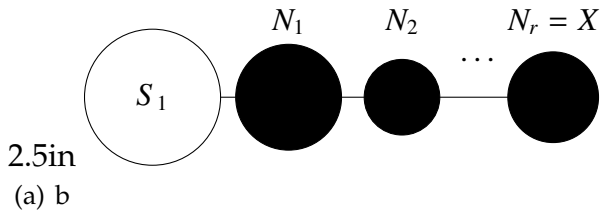


Figure 7.3: Case 1: Marking X as inactive.

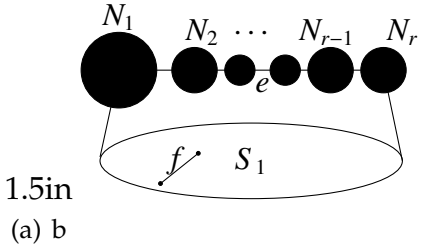


Figure 7.4: Case 2: Replacing e with f.

Figure 7.5: Neutral subsets pruned in each case to yield component S_1 with cost $< \frac{1}{2}D$.

We start by setting T_A to be the edges in T' that span S_1. By construction, these edges have cost $< \frac{1}{2}D$. We then try to grow T_A as much as possible along the path from S_1 to $N_1, N_2, \ldots, N_r$. First, suppose that we can add this full path and the edges that span each N_i to T_A without exceeding cost $\frac{1}{2}D$. Then, we set T_A to be this expanded tree and $S_A = S_1 \cup N_1 \cup \ldots \cup N_r$. Further, we set $\bar{T}$ to be the edges in T' in the corresponding component at the end of PD(λ_1). By construction, the cost of $\bar{T}$ is $\geq \frac{1}{2}D$ and $|S_A| \geq |\bar{S}|$.

Otherwise, we continue to add N_1, N_2, ... to our tree until we reach a component $\bar{X} \in \{N_1, N_2, \ldots, N_r\}$ such that adding the edges that span $\bar{X}$ to T_A implies that $\sum_{e \in T_A} c_e > \frac{1}{2}D$. In other words, we cannot add this whole subset to our tree without going over budget. Let $e = (u, v)$ be the edge that connects $\bar{X}$ to T_A in T'. If adding e to T_A already brings the cost of T_A strictly over $\frac{1}{2}D$, then we stop

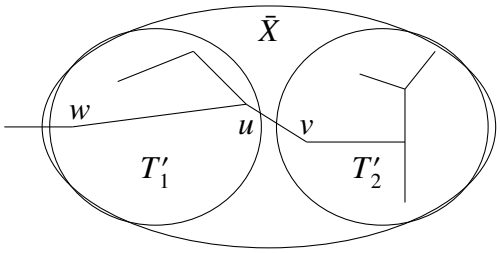


Figure 7.6: Illustration of the pick procedure.

growing T_A and set $\bar{T} = T_A \cup \{e\}$. Otherwise, we add e to T_A and run a procedure **pick**$(\bar{X}, v, T_A)$ that picks a subset of the edges spanning $\bar{X}$ including v.

Specifically, the procedure **pick**(X, w, T_A) adds to T_A a set of edges in T' that span a subset of component X including w. We denote by $X_1, X_2 \in \mathcal{S}$ the two components that merged to form X and by $e' = (u, v)$ the edge that connects X_1 and X_2 in T'. Without loss of generality, $u \in X_1, v \in X_2$. Further, let T_1' and T_2' be the edges in T' with both endpoints in X_1 and X_2, respectively. See Figure 7.6.

If the total cost of edges in $T_A \cup T_1'$ is greater than $\frac{1}{2}D$, then we know we should only add edges in this subtree to T_A and we recursively invoke **pick**(X_1, w, T_A). If instead the total cost of edges in $T_A \cup T_1' \cup \{e'\}$ is less than $\frac{1}{2}D$, then we can feasibly add e' and all edges in T_1' without violating the budget. Thus, the procedure adds all these edges to T_A and recursively invokes **pick**(X_2, v, T_A) to pick the remaining edges in T_2'. Finally, if the cost of edges in $T_A \cup T_1'$ is less than or equal to $\frac{1}{2}D$, but greater than $\frac{1}{2}D - c_{e'}$, then we cannot quite make it to T_2' without going over budget. In this case, the procedure adds all edges in T_1' to T_A and sets $\bar{T} = T_A \cup \{e'\}$.

At the end of the procedure, we produce a tree T_A of cost $\leq \frac{1}{2}D$ that spans a subset S_A along with a tree $\bar{T}$ of cost $\geq \frac{1}{2}D$ that spans a subset $\bar{S}$ where $|\bar{S}| \leq |S_A|+1$. Further, if $|\bar{S}| = |S_A| + 1$, then $\bar{T}$ has cost $> \frac{1}{2}D$. We will use $\bar{T}$ to prove a bound on

$|\bar{S}|$, which in turn will give a bound on $|S_A|$.

Let $Q \in \mathcal{S}^+$ be the maximum potential subset containing $\bar{S}$. Our goal is to prove Theorem 18, which states that

$$|S_A| \geq \frac{1}{2}\lambda_1 D + \pi(Q) - 1.$$

Our proof relies, informally speaking, on the vertex $\bar{v}$ at which the pick routine stopped. We define $\bar{v}$ as the (unique) vertex in $\bar{S}$ that has fewer edges incident in $\bar{T}$ than it does in T; though many vertices in the graph have fewer incident edges in $\bar{T}$ than in T, the definition of the pick routine, combined with the characterization in Figures 7.3 and 7.4 guarantees that only one of them is in $\bar{S}$. Further, if $|\bar{S}| > |S_A|$, then $\bar{v}$ is exactly the one node in $\bar{S}$ that is not in S_A. Before proving Theorem 18, we first state the following useful lemma, the proof of which we delay to the end of the section as it closely closely resembles that of Goemans and Williamson [1995] for the Prize-Collecting Steiner Tree Problem.

Lemma 23.

$$\sum_{e \in \bar{T}} \sum_{S : e \in \delta(S)} y_S \leq 2 \sum_{\substack{T : T \cap \bar{S} \neq \emptyset \\ \bar{v} \notin T}} y_T. \tag{7.2}$$

Proof of Theorem 18. By construction of $\bar{T}$, vertices in $Q - \bar{S}$ are either (i) in a single set $N \subset Q - \bar{S}$ that denotes the union of pruned subsets and sets N_i in Q that were not reached by the pick procedure or (ii) in the set $\bar{X} \in \{N_1, N_2, \ldots, N_p\}$ set on which we started our pick routine. Thus, we may partition $Q = N \cup (\bar{X} - \bar{S}) \cup \bar{S}$. Notice that N is a union of neutral subsets and thus itself neutral, that is,

$$|N| = 2 \sum_{T : T \in \mathcal{S}_N} y_T.$$

Similarly, let $\mathcal{S}_X$ be all subsets in $\mathcal{S}$ that are subsets of $\bar{X}$ and contain vertices in $\bar{X} - \bar{S}$. We may partition the subsets of $\bar{X}$ into sets that contain nodes in $\bar{X} - \bar{S}$

and sets that do not; we then find (since $\bar{X}$ is neutral)

$$|\bar{X}| = 2 \sum_{T : T \in \mathcal{S}_X} y_T + 2 \sum_{T : T \subseteq \bar{X} \cap \bar{S}} y_T \leq 2 \sum_{T : T \in \mathcal{S}_X} y_T + |\bar{X} \cap \bar{S}|$$

where the inequality follows by Lemma 20. Thus, $|\bar{X} - \bar{S}| \leq 2 \sum_{T : T \in \mathcal{S}_X} y_T$.

The definition of $\bar{v}$ guarantees that any subset in $\mathcal{S}$ that contains vertices in $\bar{S}$ and $\bar{X} - \bar{S}$ also contains $\bar{v}$. Therefore, subsets T that intersect with $\bar{S}$ but do not contain $\bar{v}$ are neither in $\mathcal{S}_N$ nor in $\mathcal{S}_X$. It follows that

$$\begin{aligned}
|Q| &= 2 \sum_{T : T \subseteq Q} y_T + \pi(Q) \\
&\geq 2 \sum_{\substack{T : T \cap \bar{S} \neq \emptyset \\ \bar{v} \notin T}} y_T + 2 \sum_{T : T \in \mathcal{S}_N} y_T + 2 \sum_{T : T \in \mathcal{S}_X} y_T + \pi(Q) \\
&\geq 2 \sum_{\substack{T : T \cap \bar{S} \neq \emptyset \\ \bar{v} \notin T}} y_T + |N| + |\bar{X} - \bar{S}| + \pi(Q) \\
&= 2 \sum_{\substack{T : T \cap \bar{S} \neq \emptyset \\ \bar{v} \notin T}} y_T + |Q - \bar{S}| + \pi(Q)
\end{aligned}$$

Rearranging,

$$|\bar{S}| \geq 2 \sum_{\substack{T : T \cap \bar{S} \neq \emptyset \\ \bar{v} \notin T}} y_T + \pi(Q) \geq \sum_{e \in \bar{T}} \sum_{S : e \in \delta(S)} y_S + \pi(Q) = \lambda_1 \cdot \sum_{e \in \bar{T}} c_e + \pi(Q).$$

The second inequality follows from Lemma 23, whereas the final equality is due to the fact that the dual constraints are tight for all edges obtained by the primal-dual subroutine. If $|\bar{S}| = |S_A|$, this completes the proof of the theorem. Else, suppose that $|\bar{S}| = |S_A| + 1$. Then, $\sum_{e \in \bar{T}} c_e > \frac{1}{2} D$, and the asserted inequality still holds. ■

Proof of Lemma 23. Consider a single iteration of the algorithm, and let C be the current set of components C such that $|\delta(C) \cap \bar{T}| \geq 1$. In other words, these are the components incident to edges in $\bar{T}$. We can partition C into active components

C_A and inactive components C_I. Further, let $C_{\bar{v}}$ be the component that contains $\bar{v}$. We first show that active components are not incident to too many edges in $\bar{T}$ on average.

Starting with the graph $(V, \bar{T})$, we consider the graph obtained by shrinking each component in C to a single vertex and removing all vertices not in a component in C. The remaining edges are a subset of $\bar{T}$ and form a tree on C. Further, the degree of each vertex v in this tree is $d_v = |\delta(C) \cap \bar{T}|$, where C is the corresponding component. We set R to be the set of vertices corresponding to components in C_A and set B be the set of vertices corresponding to components in C_I. Since the edges in $\bar{T}$ form a tree on C,

$$\sum_{v \in R} d_v + \sum_{v \in B} d_v \leq 2|R| + 2|B| - 2.$$

Let C be an inactive component such that $|\delta(C) \cap \bar{T}| = 1$. Note that $|\delta(C) \cap T'| > 1$, where T' is the set of all tight edges returned by the subroutine, since otherwise C would have been pruned. Therefore, the other component incident to C was not chosen to be part of $\bar{T}$. This implies that either C is $\bar{X}$ or C is some X_1 encountered during the picking procedure such that we only chose vertices in X_1 and not X_2. In either case, $\bar{v} \in C$.

If $C_{\bar{v}} \in C_I$, then this is the only component in B of degree 1 and $\sum_{v \in B} d_v \geq 2|B| - 1$. In other words,

$$\sum_{C \in C_A} |\delta(C) \cap \bar{T}| = \sum_{v \in R} d_v \leq 2|R| - 1 = 2|C_A| - 1 \tag{7.3}$$

Otherwise, $C_{\bar{v}} \in C_A$ and all vertices in B have degree at least 2. Thus, $\sum_{v \in B} d_v \geq 2|B|$. This implies that

$$\sum_{C \in C_A} |\delta(C) \cap \bar{T}| \leq 2|R| - 2 = 2|C_A| - 2. \tag{7.4}$$

We now prove the main claim in Equation (7.2) using a proof by induction. At the start of the algorithm, the LHS and RHS are both equal to 0. On each iteration, let C_A be the set of active components C such that $|\delta(C) \cap \bar{T}| \geq 1$ and let $C_{\bar{v}}$ be the component containing $\bar{v}$. Suppose that we raise y_{C_i} by ε for every active component C_i. The LHS of the claim is raised by $\sum_{C \in C_A} |\delta(C) \cap \bar{T}|\varepsilon$. If $C_{\bar{v}} \in C_I$, then the RHS is raised by exactly $2|C_A|\varepsilon$. By Equation (7.3), the LHS < RHS. If, on the other hand, $C_{\bar{v}} \in C_A$, then the RHS is raised by exactly $2(|C_A| - 1)\varepsilon$. By Equation (7.4), the LHS < RHS. In both cases, the inductive statement continues to hold. ■

7.6 Extensions

In this section, we show that the parameterized primal-dual algorithm extends to the weighted version and the minimum spanning tree problem. For the weighted case, we assume that each vertex v is associated with an integer weight $p_v \geq 0$ and the goal is to find a feasible tour F that visits a subset $S \subseteq V$ to maximize $\sum_{v \in S} p_v$. We can imagine transforming this problem to an equivalent unweighted version by creating copies of each vertex v with zero cost edges to v. Then in the primal-dual subroutine, all these added edges go tight instantaneously, yielding a weighted "cluster" with potential equal to p_v. Thus, we don't even need to perform the transformation and can actually just begin the algorithm with these weighted clusters as our initial active sets with potential equal to the weight of v. All proofs continue to hold through the equivalence to the unweighted version.

Theorem 24. *The parameterized primal-dual algorithm is a 2-approximation for the budgeted prize-collecting traveling salesman problem with weights.*

Second, we discuss the extension of the algorithm to the budgeted prize-collecting minimum spanning tree problem. The corresponding linear programming relaxation for the budgeted prize-collecting minimum spanning tree problem is given below.

$$\begin{aligned}
\text{maximize} \quad & \sum_{S \subseteq V} |S| z_S \\
\text{subject to} \quad & \sum_{e : e \in \delta(S)} x_e \geq \sum_{T : S \subsetneq T} z_T \quad \forall S \subsetneq V \\
& \sum_{e \in E} c_e x_e \leq D \\
& \sum_{S \subseteq V} z_S \leq 1 \\
& z_S, x_e \geq 0
\end{aligned}$$

The only difference is the removal of the factor of 2 from the first constraint which changes the dual constraint associated with each subset S to become

$$\sum_{T : T \subsetneq S} y_T + \Lambda_2 \geq |S| \quad \forall S \subseteq V.$$

Based on this change, we redefine the potential of a set to be

$$\pi(S) = |S| - \sum_{T : T \subsetneq S} y_T$$

and a set to be neutral if $\sum_{T : T \subseteq S} y_T = |S|$. Given these new definitions, the primal-dual subroutine runs exactly as it did before. Finally, we set the threshold value λ_1 such that the largest component in T' after running PD(λ_1^-) has cost $\geq D$ and the largest component in T' after running PD(λ_1^+) has cost $< D$. We then run the pick procedure to find a tree just within budget D. It is easily verified that all proofs continue to hold for this variant.

Theorem 25. *The parameterized primal-dual algorithm is a 2-approximation for the budgeted prize-collecting minimum spanning tree problem.*

7.7 Computational Experiments

In this section, we complete computational experiments in order to better understand the performance of our algorithm in practice. The primal-dual algorithm as detailed in this section was implemented in C++11 using binary search to find λ_1. The experiments were conducted on a Dell R620 with two Intel 2.70GHz 8-core processors and 96GB of RAM.

The first set of graphs we used for the experiments are the 37 symmetric TSP instances with at most 400 nodes in the TSPLIB data set (Reinelt [1991]). The second set of graphs are 37 weighted instances constructed using the Citi Bike network of bikesharing stations in New York City. Each instance corresponds to a week of usage data at these stations, and the weight of a vertex corresponds to the number of broken docks at that station during that week. The number of broken docks was estimated from usage data with a probabilistic method similar to that of Kaspi et al. [2016]. Details about both types of constructed instances are given in Table 7.1.

For each test graph G, we first found an upper bound on the cost of a tour by computing 2 times the cost of a minimum spanning tree in G. We then set the budget for our tour to be $f = 25\%$, 50%, or 75% of this upper bound. W denotes the total weight of the vertices; for TSPLIB instances, this is the number of vertices. After finding our solution of weight A, we compute an upper bound on the weight of visited vertices $U = \min(\lambda_1 D + \max_{S \in \mathcal{S}} \pi(S), W)$ and record the

Instance	$\|V\|$	$\|E\|$	Total Vertex Weight
TSPLIB	158.14	15658.43	158.14
Bike	319.54	4634.77	1302.51

Table 7.1: Graph statistics for each group of graphs averaged over all instances.

Instance	f	Time (s)	Recursions	% Opt. Gap	% Weight	% Budget
TSPLIB	0.25	74.16	0.59	46.67	33.06	77.38
TSPLIB	0.5	72.61	0.14	41.89	58.08	69.89
TSPLIB	0.75	71.24	0.22	18.62	81.38	68.80
Bike	0.25	25.15	0.28	45.74	43.37	66.90
Bike	0.5	33.21	0.28	25.89	74.01	67.13
Bike	0.75	30.46	0.05	8.29	91.68	67.37

Table 7.2: Computational results of the primal-dual algorithm for each group of graphs and budget with results averaged over all instances.

percent optimality gap as $100 \times (U - A)/U$. Results are given in Table 7.2. The column headed % Weight gives the percentage of the total weight W captured by the constructed tour, and the one headed % Budget gives the percentage of the distance budget used after shortcutting the tree.

We report several interesting structural results. First, the average time seems to be heavily influenced by the number of edges; the bike instances were quicker to complete even though the average number of nodes was higher. However, the average time does not seem to grow with the budget (and hence with the size of the computed solution) since most of the time is spent finding λ_1. The average optimality gap, on the other hand, does improve with the budget. This is likely due to the fact that for larger budgets the upper bound is given by W rather than $\lambda_1 D + \max_{S \in \mathcal{S}} \pi(S)$. Also of interest is that $\max_{S \in \mathcal{S}} \pi(S)$ contributed

f	0.25	0.5	0.75
% Opt. Gap without Virtual Budgets	45.74%	25.89%	8.29%
% Opt. Gap with Virtual Budgets	27.96%	11.87%	0.17%

Table 7.3: Improvement of Optimality Gap using Virtual Budgets.

little to our upper bound U. As a result, our optimality gaps depend mostly on the value of λ_1, rather than the potentials, and might be far from tight. However, the fact that on average we only use around $2/3$ of the distance budget implies that the solutions could be improved as well. To ensure that we use a larger part of the budget, we ran further experiments on the Citi Bike instances; in these, we ran binary search over possible *virtual budgets* in the input until finding one for which the resulting tour uses at least 90% of the actual budget. This significantly reduced our optimality gaps as shown in Table 7.3.

CHAPTER 8

INDUSTRY IMPACT

Will: How much of what you're saying do you believe right now?

Charlie: 60%

Will: I thought it was in the mid-80s. You pulled it off.

Charlie: Experience.

The work in this part has seen widespread use in bike-sharing systems across the United States. Here, we briefly summarize aspects that have been implemented and ones that have not – for the latter, we highlight some of the obstacles we faced in implementing them.

Allocation of Capacity. The analysis we introduced in Chapter 4 has affected the design of several BSSs operated by Motivate. As described in Section 4.5, a pilot in New York City involved the movement of 34 docks. Since then, the Divvy system in Chicago moved a total of 200 docks. Further, all systems operated by Motivate, including D.C. and Boston, now run the analyses biannually to ensure they capture the potential of underutilized docks as a sustainable way to improve service delivery. Motivate's own analysis, based on the cost of reallocating capacity, the cost of rebalancing bikes, and the reduction in out-of-stock events due to reallocated capacity, indicates that the cost of reallocating amortizes within just 2 weeks! It thereby exemplifies the value of a data-driven system design in practice.

Incentive Scheme. The scope of the *Bike Angels* program in New York City has continuously increased since we first piloted it in October 2015. As of early 2017 it accounted for 10% of Citi Bike's rebalancing (Grabar [2017]) at a much

lower cost than other forms of rebalancing. In fact, its success has since also led to its adoption by Ford GoBike in the San Francisco Bay Area. In practice, the program is more complicated than the focus of Chapter 5, allowing participants to earn duplicate points etc. Nevertheless, the UDFs drive the underlying analytics based on essentially the exact logic described in the chapter. Furthermore, it was our analysis that drove the decision to adopt a dynamic incentive scheme.

Rebalancing. In chapter 6, we provided models and solutions for three different optimization problems that arise when rebalancing bike-share systems. Of the three methods discussed, only the corral plan was implemented in its entirety:it was in place during the summer of 2016. The trailer formulation was never fully implemented, but it did identify pairs of stations that had trailers added after discussions with Citi Bike. Furthermore, our corral analysis demonstrated that even just a few corrals can have significant improvement for users' ability to access the system. The overnight rebalancing failed to become a part of Citi Bike's operations (beyond the pilots), mostly due to the complexity of the real-world operations discussed in Section 6.2. In a subsequent research project, two Cornell undergraduates (Shangdi Yu and Ellen Chen) built a web-tool that might yet be adopted for operations. The main reason this has not yet occurred is due to the existing decision aid, based on O'Mahony [2015] that is already in place. In this case, one could claim that good was the enemy of great: the existence of a well-working decision aid hindered the adoption of a more sophisticated routing tool. This is despite the routing formulation improving 20% on average.

Maintenance. The contribution of Chapter 7 is mostly of theoretical nature. In practice, maintenance scheduling involves more than one repairman and

would require solving a rooted version for each one; also, rather than only focusing on dissatisfied users, service-level agreements between operators and cities force high priority on docks that have been explicitly reported by customers. That being said, the model was originally inspired by the development of the analytics we developed for Citi Bike to identify broken docks (cf. Figure 1.4). These are still in use with an accuracy of more than 90% in identifying docks that are either defective or about to break.

Part II

Queuing Network Models for Shared Vehicle Systems

Far outpacing bike-sharing systems, ride-sharing systems, such as Uber and Lyft, are fast becoming essential components of the urban transit infrastructure. In such systems, passengers can request rides (subject to vehicle availability) between a large number of source-destination locations. In contrast to bike-sharing systems, vehicles in ride-sharing systems are operated by independent drivers (units).

Similar to bike-sharing systems, ride-sharing systems experience inefficiencies due to *limited supply* (drivers) and *demand heterogeneity* across time and space. These inefficiencies, however, can often be greatly reduced through operational controls. Pricing has traditionally been the main tool to balance demand and supply in settings with limited supply and heterogeneous demand: for instance, in limited-item auctions or airplane/hotel reservations, see Hartline [2016] and Gallego and Van Ryzin [1994]. In contrast to bike-sharing systems, ride-sharing platforms have also long utilized dynamic pricing. Ride-sharing systems can also balance demand and supply through other controls, such as repositioning empty vehicles (idle drivers), or redirecting customers to nearby locations (cf. Sections 9.1, 10.1). Moreover, these tools can be used towards achieving different objectives – examples include revenue/welfare/throughput maximization, and multi-objective settings such as Ramsey pricing.

In contrast to other limited supply settings, however, shared vehicle settings are more challenging due to two unique features. The first is the presence of *spatial and temporal supply externalities*: whenever a customer engages a vehicle, this not only decreases the instantaneous availability at the source location, but also affects the future availability at all other locations in the system. The other distinguishing feature is the high frequency of events (passenger arrivals/rides) in

such settings: in New York City, 10017 Citi Bikes were used for an average 62,605 trips/day in September 2017 (NYCBS [2017c]), whereas in October 2016, Uber averaged 226,046 rides per day delivered by 46,000 drivers (Hu [2017]) – with each driver working at most 12 consecutive hours (Hawkins [2016]). These distinctions necessitate treating the problem as an infinite-horizon control problem, as opposed to finite-horizon dynamic programming approaches used in traditional transportation and revenue management settings (Gallego and Van Ryzin [1994], Adelman [2007]). The high frequency of events tends to drive such systems into operating under a dynamic equilibrium state, jointly determined by the demand and supply characteristics as well as the chosen controls; the performance of any pricing/control policy is determined by this equilibrium.

All the above features (limited supply, demand heterogeneity, supply externalities and fast operational timescales) are well captured by *closed queueing network models* (cf. Serfozo [1999], Kelly [2011]), which are thus widely adopted in recent research on ride-sharing systems (George [2012], Zhang and Pavone [2016], Waserhole and Jost [2014], Braverman et al. [2016], Ozkan and Ward [2016]). These models use a Markov chain to track the number of vehicles across locations. Each location experiences a stream of arriving customers, who engage available vehicles and take them to their desired destination. For example, increasing the price for a ride between a pair of stations decreases the number of customers willing to take that ride, which over time affects the distribution of vehicles across all stations. Even though such models can be well-calibrated, based on demand rates and price elasticities estimated from historical data, the problem of designing good pricing/control policies under such a model is complex due to a combination of high dimensionality and intrinsic non-concavity of the optimization problem (cf. Section 9.2.1). Consequently, previous work

has focused only on a narrow set of objectives (typically, weighted throughput) and is largely based on heuristics or simulation/numerical techniques, with few provable guarantees (cf. Section 9.1 for a discussion). Algorithms for other objectives (e.g. revenue and welfare) as well as more complex constrained settings, have yet to be addressed.

In this context, our work develops *the first efficient algorithms for designing pricing and control policies in closed queueing networks, with approximation guarantees for a large class of objectives*. More generally, we provide a unified framework for designing rebalancing policies, which can incorporate a variety of controls and constraints, including multi-objective settings and incorporating travel time distributions. In all settings, we obtain parameterized performance guarantees, which improve with the number of vehicles in the system, and are near-optimal in the parameter regimes of real systems. Moreover, our guarantees also provide an elementary proof of the asymptotic optimality of our policies under the so-called large-market scaling (see Braverman et al. [2016] and Ozkan and Ward [2016]), without necessitating the derivation of the associated fluid limits. Our main guarantee (cf. Section 9.3) leverages techniques from convex optimization and approximation algorithms, and combines these with a novel infinite projection and pullback technique. Given the widespread use of closed queueing models for a variety of other applications, we anticipate that our framework will prove useful in other areas as well.

Outline

In Section 9.1 we begin with a detailed review of recent research involving stochastic models of ride-sharing systems. Next, we formally define the basic

version of our model in Section 9.2: there, pricing is the only control available and rides experience no transit delays. As a primary example, we focus in Section 9.3 on this basic version with the objective given by throughput; this allows us to highlight our main technical contribution (cf. Section 9.3).

Throughout, we model a shared vehicle system with m vehicles and n stations as a continuous-time finite-state Markov chain (CTMC) that tracks the number of drivers (units) at each station (node), and use this to study a variety of pricing and control problems. A high-level description of our most basic model is as follows (cf. Section 9.2 for details). Each station in the system observes a Poisson arrival of customers. Arriving customers draw a value and a destination from some known distribution. Upon arrival at a station, the customer is quoted a price and one of three scenarios occurs: i) the customer is not willing to pay the price, i.e., the price exceeds her value, and she leaves the system; ii) the customer is willing to pay the price, but no unit is available at the node; therefore she again leaves the system; or iii) the customer is willing to pay the price, and a vehicle is available. A ride occurs only in the final case with the vehicle moving to the customer's destination; the number of vehicles at the origin is decremented instantaneously, while the state at the destination is incremented either instantaneously (cf. Section 9.3), or more generally, after some random time interval (cf. Section 10.2). This describes the basic dynamics under which we aim to maximize the long-run average performance, measured by the throughput, the social welfare, or the revenue obtained in steady-state. The prices can, in general, depend on the instantaneous state of the system. Thus, the resulting optimization problem is high-dimensional, and moreover, it is non-convex even in basic settings (cf. Section 9.2.1).

In Section 9.3, we propose a simple pricing policy, based on optimizing over a novel convex relaxation, which we term the *elevated flow relaxation*. We adapt the objective by identifying a concave pointwise upper bound which we call *elevated objective*. Furthermore, we introduce additional flow-conservation constraints to capture the network externalities. As the elevated objective is bounded below by the original objective, optimal solutions in the elevated optimization problem are bounded below in value by optimal solutions in the original optimization problem.

In Section 9.3.2, we present our approximation guarantee: we show that the elevated flow relaxation can be efficiently solved to derive a pricing policy which has an approximation ratio of $1 + (n-1)/m$. Even though we consider general state-dependent pricing policies, the policy that achieves our guarantee surprisingly turns out to be state-independent, i.e. the prices do not differ based on the configuration of units across nodes. The idea of the proof is based on the following three steps:

1. First, we notice that, for any state-dependent policy in the m-unit system (and therefore also for the optimal policy), there exists a feasible solution in our relaxation that upper bounds its value. Hence, the elevated value of our policy upper bounds the original objective of the optimal state-dependent policy in the m-unit system.
2. Next, we observe that the elevated objective of our policy is equal to the original objective under an appropriately defined *infinite-supply setting*. In particular, we consider a restricted subset of *state-independent pricing policies*, under which the resulting CTMC has the structure of a so-called *closed Jackson network*, and prove that as the number of vehicles grows to infinity,

the elevated objective collapses to the original objective.

3. Last, we show that the performance of any policy in the m-unit setting approximates its performance in the infinite-supply setting within a factor of $1 + (n-1)/m$. Though the intuition behind this pull-back step can be seen via simple stochastic coupling arguments, we provide a fully algebraic proof based on a combinatorial construction of a biregular graph that relates the state spaces of the m- and $(m-1)$-unit systems.

After formalizing these ideas for the simplest of settings in Section 9.3.2, we show in Chapter 10 how the above framework, comprising of a policy derived via an appropriate elevated flow relaxation, and the three-step process to prove its guarantees, can be applied in a number of related settings:

- We first turn our attention to objectives beyond throughput like social welfare and revenue. Next, we consider multiobjective settings in which the goal is to maximize one objective subject to a lower bound on another, the so-called Ramsey pricing problem (Ramsey [1927]): designing a pricing policy to maximize system revenue subject to a lower bound on the system welfare. This is particularly relevant when systems are operated by private companies in close partnership with city governments. For instance, the Citi Bike system in New York City is run by Motivate, a private company, under service-level agreements with the NYC Department of Transportation. Note that the complementary problem (maximizing welfare subject to revenue constraints) is of interest when such systems are managed by non-profit organizations and is considered in other paradigms such as the FCC spectrum auction (Milgrom and Segal [2014]). In this context we demonstrate how our approach can be used to obtain a (γ, γ) bicriteria

approximation guarantee with $\gamma = (1 + \frac{n-1}{m})$.

- Next, we study two other rebalancing controls considered in the literature, and obtain $1 + (n-1)/m$ approximation guarantees for the respective optimization problems. In the first, units can be directed to a new location after ending a trip; in the second, customers can be matched to units at neighboring nodes. In both cases, we recover and strengthen the previous results.
- We conclude Chapter 10 with the study of settings in which rides do not occur instantaneously but instead require some delay (travel time). Our results in this section provide an elementary proof of the so-called large-market optimality (Braverman et al. [2016]) of our algorithms. Further, they characterize an interesting dichotomy between settings in which convergence can happen at a rate no more than the square-root of the vehicles on the one hand and settings in which convergence happens can happen at a linear rate.

This part is based on Banerjee, Freund, and Lykouris 2017; our results recover and unify many existing results in this area, and provide a general framework for deriving approximation algorithms for many other settings. Moreover, the guarantees we obtain are close to 1 for realistic system parameters. For instance, for the parameters $(m = 10000, n = 600)$ of New York City's Citi Bike system in summer 2016, we obtain an approximation ratio of 1.06.

CHAPTER 9

A BASIC MODEL

"I left my purse up at the office. And I'll need some cash for the cab. And for the cab tomorrow morning." — MacKenzie McHale

9.1 Related work

There is a large literature on characterizing open and closed queueing network models, building on seminal work of Jackson [1963], Gordon and Newell [1967], and al. Baskett et al. [1975]; the books by Kelly [2011] and Serfozo [1999] provide an excellent summary. Optimal resource allocation in open queueing networks also has a long history, going back to the work of Whittle [1985]. However, there is much less work for closed networks, in part due to the presence of a normalization constant for which there is no closed-form (though it is computable in $O(nm)$ time via iterative techniques Buzen [1973], Reiser and Lavenberg [1980]). Most existing work on optimizing closed queueing networks use heuristics, with limited or no guarantees. In contrast, our work focuses on obtaining algorithms with provable guarantees for a wide range of problems.

Three popular approaches for closed queueing network optimization in the literature are: (*i*) using open queueing network approximations, (*ii*) heuristically imposing a 'fairness' property, which we refer to as the demand circulation constraint (cf. Section 9.3.1), and (*iii*) characterizing the fluid limits of closed queueing networks, and obtaining solutions that are optimal in these scaling regimes. We now briefly describe each approach.

The first approach was formalized by Whitt [1984], via the fixed-population-mean (FPM) method, where exogenous arrival rates are chosen to ensure the mean population is m. It has since been used in many applications; for example, Brooks et al. [2013] use it to derive policies for matching debris removal vehicles to routes following natural disasters. Performance guarantees however are available only in restricted settings.

Another line of work is based on heuristics that enforce the demand circulation property (variously referred to as the demand rebalancing, the fairness, or the bottleneck property). In transportation settings, George [2012] used these to optimize weighted throughput, Zhang and Pavone [2016] to minimize rebalancing costs . Most works typically only provide asymptotic guarantees (George et al. [2012]).

More recently, Ozkan and Ward [2016] and Braverman et al. [2016] characterized appropriate fluid (or large-market) limits for closed queueing networks, and used it to study the operations of ride-sharing systems. In contrast to our work, which focuses on optimizing a given finite-m system, these works consider a regime where m and the arrival rates of passengers together scale to ∞, and characterize the optimal policy in the limit. Within this limit, the former studied the assignment of customers to nearby drivers, whereas the latter considered directing drivers at the end of each trip to under-served locations. Our extensions to settings beyond pricing (cf. Section 10.1) are inspired by these works; in particular, we show that similar scaling results can be derived within our framework. Moreover, our work provides guarantees for the resulting policies in the finite case (i.e., before taking the limit), and also against a much more general class of state-dependent policies.

The closest work to ours is that of Waserhole and Jost [2014], who provide a pricing policy for maximizing throughput in closed queueing networks, with the same approximation ratio we obtain. They do this via a different argument wherein they observe that, under the demand circulation property, the Markov chain is doubly stochastic, and hence has a uniform distribution (this was also noted earlier by Whitt [1984]). A simple counting argument then implies that the probability of a station having a vehicle is $m/(m+n-1)$. Moreover, since the maximum throughput under any policy is bounded by the maximum demand circulation, the maximum throughput under demand circulation is within a $m/(m+n-1)$ factor of the optimum. This argument is finely tuned to this particular setting (maximizing throughput via pricing with no delays). In contrast, our approach can accommodate several objectives and rebalancing controls as well as delays.

Finally, we note that there is a parallel line of work which tackles settings with dynamic arrivals and pricing, using techniques from approximate dynamic programming (Adelman [2007], Hampshire et al. [2009]). These typically can deal only with small systems, as their dimensionality scales rapidly with the number of stations; moreover, many of the techniques have no provable guarantees. In contrast, the flavor of results of Levi and Radovanovic [2010] are somewhat similar to ours: they study a setting in which a knapsack constraint exists on used resources, contrasting our network constraints. Similar to our results, they also use a LP relaxation to obtain approximation guarantees for parameter regimes of interest and optimal results for asymptotic regimes.

9.2 Preliminaries

In this section, we first formally define our model of shared vehicle systems and formulate the optimal pricing problem. To capture the complex network externalities of the system, we define a probabilistic model of customer arrivals, which we analyze in steady state. Subsequently, we introduce known results from the queuing literature that provide the technical background upon which our analysis relies. Finally, we present an example that shows that even in the restricted sets of pricing policies, that are independent of the configuration of vehicles across the system, the optimization problems we consider are non-convex.

9.2.1 Basic setting

We consider a system with m units (corresponding to vehicles) and n nodes (corresponding to stations). Customers traveling between nodes i and j arrive at node i according to a Poisson process of rate ϕ_{ij}. Each customer traveling from i to j has a value drawn independently from a distribution $F_{ij}(\cdot)$. We assume that F_{ij} has a density and that all values are positive with some probability, i.e. $F_{ij}(0) < 1$. Upon arrival at i, a customer is quoted a price p_{ij}, and engages a unit if her value exceeds this price, i.e. with probability $1 - F_{ij}(p_{ij})$, and at least one unit is available at node i; else she leaves the system.

As is common with pricing, the related optimization problems are often more easily framed in terms of the *inverse demand* (or *quantile*) function associated with the user as $q_{ij} = 1 - F_{ij}(p_{ij})$. For ease of presentation we assume that the

density of F_{ij} is positive everywhere in its domain, implying that there is a 1-1 mapping between prices and quantiles. As F_{ij} is therefore invertible, we can write $p_{ij} = F_{ij}^{-1}(1 - q_{ij})$. This allows us to abuse notation throughout the paper by using prices and quantiles interchangeably.

A continuous-time Markov chain tracks the number of units across nodes. At time $t \geq 0$, the *state* of the Markov chain $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ contains the number of units $X_i(t)$ present at each node i. The state space of the system is denoted by $\mathcal{S}_{n,m} = \left\{(x_1, x_2, \ldots, x_n) \in \mathbb{N}_0^n | \sum_i x_i = m\right\}$. Throughout the paper we use $\mathbf{X}(t), X_i(t)$ to indicate random variables, and $\mathbf{x}, x_i$ to denote specific elements of the state space. Note that the state-space is finite; moreover, $|\mathcal{S}_{n,m}| = \binom{m+n-1}{n-1} = \Omega(m^n)$. Since our focus is on the long-run average performance, i.e. system performance under the steady state of the Markov chain, we henceforth suppress the dependence on t for ease of notation.

For ease of presentation, we assume that rides between nodes occur without delay. In the context of our model, this translates into an instantaneous state transition from $\mathbf{X}$ to $\mathbf{X} - e_i + e_j$ when a customer engages a unit to travel from i to j (where e_i denotes the ith canonical unit vector). We relax this assumption in Section 10.2.

Pricing Policies and Objectives

We consider pricing policies that select point-to-point prices p_{ij} as a *function of the overall state* $\mathbf{X}$. Formally, given arrival rates and demand elasticities $\{\phi_{ij}, F_{ij}(\cdot)\}$, we want to design a pricing policy $\mathbf{p}(\cdot) = \{p_{ij}(\cdot)\}$, where each $p_{ij} : \mathcal{S}_{n,m} \to \mathbb{R} \cup \{\pm\infty\}$ maps the state to a price for a ride between i and j. Equivalently, we want to

select quantiles $\mathbf{q}(\cdot) = \{q_{ij}\}$ where each $q_{ij} : \mathcal{S}_{n,m} \to [0,1]$. For a fixed pricing policy $\mathbf{p}$ with corresponding quantiles $\mathbf{q}$, the *effective demand stream* from i to j (i.e. customers traveling from node i to j with value exceeding p_{ij}) thus follows a state-dependent Poisson process with rate $\phi_{ij} q_{ij}(\mathbf{X})$. This follows from the notion of probabilistic thinning of a Poisson process – the rate of customers wanting to travel from i to j is a Poisson process of rate ϕ_{ij}, and each customer is independently willing to pay p_{ij} with probability $q_{ij} = 1 - F_{ij}(p_{ij})$. State-dependent prices also allow us to capture unavailability by defining $q_{ij}(\mathbf{x}) = 0$ if $x_i = 0$ (i.e. a customer with origin i is always turned away if there are no units at that station; recall we defined $F_{ij}(\infty) = 1$). Thus, a pricing policy $\mathbf{p}$, along with arrival rates and demand elasticities $\{\phi_{ij}, F_{ij}(\cdot)\}$, determines the transitions of the Markov chain. Note that this is a finite-state Markov chain, and furthermore, is irreducible under weak assumptions on the prices and the demand (cf. Appendix 13.1); hence, it has a unique steady-state distribution $\pi(\cdot)$ with $\pi(\mathbf{x}) \geq 0 \; \forall \, \mathbf{x} \in \mathcal{S}_{n,m}$ and $\sum_{\mathbf{x} \in \mathcal{S}_{n,m}} \pi(\mathbf{x}) = 1$.

Our goal is to design a pricing policy $\mathbf{p}$ to maximize the steady-state performance under various objectives. In particular, we consider objective functions that decompose into per-ride reward functions $I_{ij} : \mathbb{R} \to \mathbb{R}$, which correspond to the reward obtained from a customer engaging a ride between stations i and j at price p. The per-ride rewards corresponding to the three canonical objective functions are:

- *Throughput*: the total rate of rides in the system; for this, we set $I_{ij}^T(p) = 1$.
- *Social welfare*: the per-ride contribution to welfare is given by $I_{ij}^W(p) = \mathbb{E}_{V \sim F_{ij}}[V | V \geq p]$.
- *Revenue*: to find the system's revenue rate, we can set $I_{ij}^R(p) = p$.

We abuse notation to define $I_{ij}(q) \triangleq I_{ij}(F_{ij}^{-1}(1-q))$ as a function of the quantile instead of the price. We also define the *reward curves* $R_{ij}(q) := q \cdot I_{ij}(q)$ (analogous to the notion of revenue curves; cf. Hartline [2016]). Our results require the technical condition that $R_{ij}(q)$ are concave in q, which implies that $I_{ij}(q)$ are non-increasing in q (equivalently $I_{ij}(p)$ are non-decreasing in p). We note that this assumption holds for throughput and welfare under all considered distributions, and revenue for regular distributions. For completeness, we prove these observations in Appendix 13.2.

For a given objective, our aim is to select a pricing policy $\mathbf{p}$, equivalently quantiles $\mathbf{q}$, that maximizes the steady-state rate of reward accumulation, given by

$$\textsc{Obj}_m(\mathbf{q}) = \sum_{\mathbf{x} \in \mathcal{S}_{n,m}} \pi(\mathbf{x}) \cdot \Big(\sum_{i,j} \phi_{ij} \cdot q_{ij}(\mathbf{x}) \cdot I_{ij}(q_{ij}(\mathbf{x})) \Big) = \sum_{\mathbf{x} \in \mathcal{S}_{n,m}} \pi(\mathbf{x}) \cdot \Big(\sum_{i,j} \phi_{ij} \cdot R_{ij}(q_{ij}(\mathbf{x})) \Big). \quad (9.1)$$

Intuitively, Equation (9.1) captures that at any node i, customers destined for j arrive via a Poisson process with rate ϕ_{ij}, and find the system in state $\mathbf{x} \in \mathcal{S}_{n,m}$ with probability $\pi(\mathbf{x})$. They are then quoted a price $p_{ij}(\mathbf{x})$ (corresponding to quantile $q_{ij}(\mathbf{x})$), and engage a ride with probability $q_{ij}(\mathbf{x})$. The resulting ride then contributes in expectation $I_{ij}(q_{ij}(\mathbf{x}))$ to the objective function. Recall that unavailability of units is captured by our assumption that $q_{ij}(\mathbf{x}) = 0$ whenever $x_i = 0$.

State-Independent Pricing and Closed Queueing Models

The Markov chain described in Section 9.2.1 has the structure of a *closed queueing network* (cf. Serfozo [1999], Kelly [2011]), a well-studied class of models in applied probability (closed refers to the fact that the number of units remains constant;

in open networks, units may arrive and depart from the system). Our analysis crucially relies on some classical results from the queuing theory literature, which we review in this section. Our presentation here closely resembles that of Serfozo Serfozo [1999]. One particular class of pricing policies is that of *state-independent* policies, wherein we set point-to-point prices $\{p_{ij}\}$ which do not react to the state of the system. As a consequence, the rate of units departing from any node i at any time t when $X_i(t) > 0$ is a constant, independent of the state of the network. The resulting model is a special case of a closed queueing model proposed by Gordon and Newell [1967].

Definition 26. *A* Gordon-Newell network *is a continuous-time Markov chain on states* $\mathbf{x} \in \mathcal{S}_{n,m}$*, in which for any state* $\mathbf{x}$ *and any* $i, j \in [n]$*, the chain transitions from* $\mathbf{x}$ *to* $\mathbf{x} - e_i + e_j$ *at a rate* $\lambda_{ij}\mu_i \mathbb{1}_{\{x_i(t)>0\}}$*, where* $\mu_i > 0$ *is referred to as the* service rate *at node* i*, and* $\lambda_{ij} \geq 0$ *as the* routing probabilities *satisfying* $\sum_j \lambda_{ij} = 1$.

In other words, if units are present at a node i in state $\mathbf{x}$, then departures from that node occur according to a Poisson distribution with rate $\mu_i > 0$; conditioning on a departure, the destination j is chosen according to state-independent routing probabilities λ_{ij}.

The Markovian dynamics resulting from state-independent pricing policies fulfill the conditions of Gordon-Newell networks: fixing a price p_{ij} (with corresponding q_{ij}) results in a Poisson process with rate $\phi_{ij}q_{ij}$ of arriving customers *willing to pay price* p_{ij}. These customers engage a unit only if one is available, else leave the system. Thus, given quantiles $\mathbf{q}$, the time to a departure from node i is distributed exponentially with rate $\mu_i = \sum_j \phi_{ij}q_{ij}$ when $X_i > 0$ and with rate 0 otherwise. Further, conditioned on an arriving customer having value at least equal to the quoted price, the probability that the customer's destination is j, is

$\lambda_{ij} = \phi_{ij} q_{ij} / \sum_k \phi_{ik} q_{ik}$, independent of system state.

One advantage of considering state-independent policies (and drawing connections with Gordon-Newell networks) is that the resulting steady-state distribution $\left\{\pi_{\mathbf{p},m}(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{S}_{n,m}}$ can be expressed in product form, as established by the Gordon-Newell theorem.

Theorem 27 (Gordon-Newell Theorem 1967)**.** *Consider an m-unit n-node Gordon-Newell network with transition rates μ_i and routing probabilities λ_{ij}. Let $\{w_i\}_{i\in[n]}$ denote the invariant distribution associated with the routing probability matrix $\left\{\lambda_{ij}\right\}_{i,j\in[n]}$, and define the* traffic intensity *at node i as $r_i = w_i / \sum_j \phi_{ij}$. Then the stationary distribution is given by:*

$$\pi(\mathbf{x}) = \frac{1}{G_m} \prod_{j=1}^{n} \left(r_j\right)^{x_j}, \tag{9.2}$$

where the Gordon-Newell normalization constant is given by $G_m = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \prod_{j=1}^{n} \left(r_j\right)^{x_j}$.

We now show how the Gordon-Newell theorem can be used to simplify the objective function in Equation (9.1). Recall that for an m-unit system with state-independent policy $\mathbf{p}$ (with corresponding quantiles $\mathbf{q}$), we obtain a Gordon-Newell network with service rate $\sum_j \phi_{ij} q_{ij}$ and routing probabilities $\phi_{ij} q_{ij} / \sum_k \phi_{ik} q_{ik}$ at node i. Let $\{\pi(\mathbf{x})\}_{\mathbf{x}\in\mathcal{S}_{n,m}}$ be the corresponding steady-state distribution. Since $\mathbf{q}$ is no longer a function of the system state, we can no longer set $q_i = 0$ when $X_i = 0$. Instead, we define $A_{i,m}(\mathbf{q}) = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x}) \mathbb{1}_{\{x_i>0\}}$ as the steady-state *availability* of units at node i (i.e. the probability in steady-state that at least one unit is present at node i), and $f_{ij,m}(\mathbf{q}) = A_{i,m}(\mathbf{q}) \cdot \phi_{ij} q_{ij}$ to be the *steady-state rate of units* moving from node i to j. Then, from Equation (9.2), one can derive (see e.g. Proposition 1.33 and Equation 1.31 in Serfozo [1999])

$$A_{i,m}(\mathbf{q}) = (G_{m-1}(\mathbf{q}) / G_m(\mathbf{q})) \cdot r_i(\mathbf{q}). \tag{9.3}$$

Notice that $r_i(\mathbf{q})$ denotes the traffic intensity as defined above. Now, the objective in Equation (9.1) can be written as

$$\text{OBJ}_m(\mathbf{q}) = \sum_i A_{i,m}(\mathbf{q}) \cdot \left(\sum_j \phi_{ij} q_{ij} \cdot I_{ij}(q_{ij}) \right) = \sum_i f_{ij,m}(\mathbf{q}) I_{ij}(\mathbf{q}). \tag{9.4}$$

For ease of notation, we omit the explicit dependence on m when clear from context.

The infinite-unit limit: The stationary distribution described above (for state-independent pricing policies) holds for any finite m; moreover, it can also be used to obtain the limiting distribution when the number of units tends to infinity. This infinite-unit limit is described in detail in Section 3.7 in Serfozo [1999] (and we provide more details in Appendix 13.3). For the purposes of our results, we rely on one particular fact, which we state in the proposition below. Recall first that given $\mathbf{p} = \{p_{ij}\}$, the quantities $w_i(\mathbf{p})$ and $r_i(\mathbf{p})$ are independent of m.

Proposition 28. *Given a policy with quantiles* $\mathbf{q}$*, in the infinite-unit limit, the steady-state availability of each node* i *is given by* $r_i(\mathbf{q}) / \max_j r_j(\mathbf{q})$*; in particular, there exists at least one node* i *with* $A_i(\mathbf{q}) = 1$*.*

The existence of a node with availability 1 essentially captures the fact that in an infinite-unit system, at least one node must have an infinite number of units. For a formal proof of this result, cf. Section 3.7 in Serfozo [1999].

Non-concavity of objective under state-independent pricing

Directly optimizing the finite-unit system is non-trivial as the objective function is not concave in prices (or quantiles); we now demonstrate this in a simple

network ($m = 1$ and $n = 3$), using throughput as the objective. Our example is presented in Figure 9.1. The network comprises of three nodes (A, B, C); the labels on the edges show the effective demand rate $\phi_{ij}(\mathbf{q})$ with which people wanting to move from node i to node j arrive for the corresponding pricing policies $\mathbf{p}$ (and corresponding quantiles $\mathbf{q}$). In particular, the first figure corresponds to setting all prices to 0 (quantiles to 1), while in the second and third figures, we increase the price between B and C to set quantile $q_{BC} = (1 + \epsilon)/2$ in figure II, and $q_{BC} = \epsilon$ in figure III. Note that the demand in network II is the average of the demands in networks I and III. To prove that this is non-concave with respect to the demand rates we now demonstrate that the throughput in network II is less than half of the sum of its value in networks I and III. To compute the throughput in each network, note that the expected waiting time at a node is inversely proportional to the total effective demand at each node. Furthermore, the unit makes exactly two rides between consecutive visits to node B. Thus, the expected throughput is twice the expected rate of return to node B. This holds because, starting from node B, the expected time for the first $2k$ rides (for any positive integer k) is k times the expected return time to node B. The expected return-time to B in the three networks can be computed as follows, where we use that the total expected waiting time can be computed as the sum of the expected waiting time at B, the probability of waiting at A times the expected waiting time at A (1), plus the probability of waiting at C times the expected waiting time at C (ϵ).

$$\begin{aligned}
\text{Network I}: &\quad 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{\epsilon} = \frac{1 + 2\epsilon}{2\epsilon} = \Omega\left(\frac{1}{\epsilon}\right) \\
\text{Network II}: &\quad 1 \cdot \frac{1}{1 + \frac{1+\epsilon}{2}} + \left(\frac{2}{3+\epsilon}\right) \cdot 1 + \frac{1+\epsilon}{3+\epsilon} \cdot \frac{1}{\epsilon} = \frac{5\epsilon + 1}{\epsilon \cdot (3+\epsilon)} = \Omega\left(\frac{1}{\epsilon}\right) \\
\text{Network III}: &\quad 1 \cdot \frac{1}{1+\epsilon} + \left(\frac{1}{1+\epsilon}\right) \cdot 1 + \left(\frac{\epsilon}{1+\epsilon}\right) \cdot \frac{1}{\epsilon} = \frac{3}{1+\epsilon} = O(1).
\end{aligned}$$

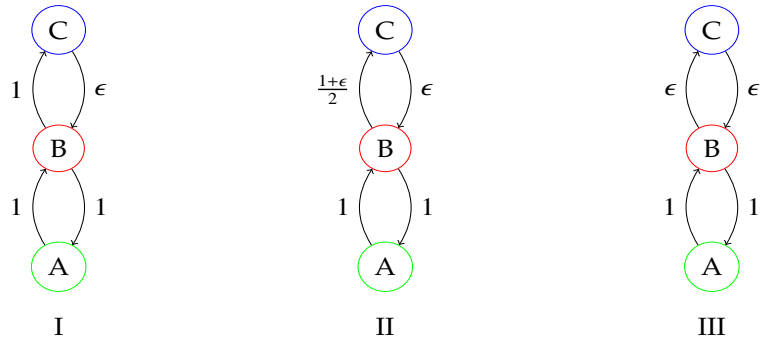


Figure 9.1: Example for non-concavity of throughput for finite units ($m = 1, n = 3$)

Thus, the throughput in I and II is $O(\epsilon)$, whereas it is constant in III, so the throughput is non-concave in the demand-rates (quantiles).

9.3 Pricing in the Vanilla Case

In this section, we present our algorithm for the simplest pricing problem in which the objective is throughput, i.e., $I_{ij}^{T}(p) = 1 \forall p$. Section 9.2.1 demonstrates that the state-independent pricing problem is non-convex; moreover, this non-convexity appears in both the objective and the constraints. We circumvent this via a novel convex relaxation, based on two separate interventions, that alleviates the technical hurdles. Surprisingly, the resulting pricing policy, which we derive in Section 9.3.1, has strong performance guarantees even with respect to state-dependent policies, as we prove in Section 9.3.2.

We develop our pricing policy in this section by first dealing with the non-convexity in the objective, then with ones in the constraints. Then, at the end of the section, we formally state the algorithm.

Elevated Objective Function

Recall from Equation (9.4) that our objective can be written as

$$\textsc{Obj}_m(\mathbf{q}) = \sum_{i,j}\Big(f_{ij,m}(\mathbf{q}) \cdot I_{ij}(q_{ij})\Big).$$

Let $\widehat{q}_{ij} = f_{ij,m}(\mathbf{q})/\phi_{ij} = A_{i,m}(\mathbf{q}) \cdot q_{ij}$; note that $\widehat{q}_{ij} \leq q_{ij}$, and moreover, unlike the quantiles q_{ij} which are in one-to-one correspondence to prices, there is no straightforward way to derive $\widehat{q}_{ij}$ from prices. Since we assume that the per-ride rewards $I_{ij}(\cdot)$ are non-increasing on the quantile space, we have $I_{ij}(q_{ij}) \leq I_{ij}(\widehat{q}_{ij})$. We now define the *elevated objective function* as

$$\widehat{\textsc{Obj}}(\widehat{\mathbf{q}}) = \sum_{i,j} \phi_{ij}\widehat{q}_{ij}I_{ij}(\widehat{q}_{ij}) = \sum_{i,j} \phi_{ij}R_{ij}(\widehat{q}_{ij}). \tag{9.5}$$

The elevated objective has two useful properties: *i*) for all m and $\mathbf{q}$, the elevated objective upper bounds the true objective function, i.e. $\widehat{\textsc{Obj}}(\mathbf{q}) \geq \textsc{Obj}_m(\mathbf{q})$, and *ii*) it is a concave function of $\widehat{q}$ (since we focus on objectives corresponding to concave reward curves $R_{ij}(\cdot)$).

The Flow Polytope

We now turn our attention to the constraints of our pricing problem. As we discussed above, each pricing policy (with corresponding quantiles $\mathbf{q}$) realizes steady-state flows (steady-state rates of units) $f_{ij,m}(\mathbf{q}) = A_{i,m}(\mathbf{q})\phi_{ij}q_{ij}$. As before, we define the change of variables $\widehat{q}_{ij} = f_{ij,m}(\mathbf{q})/\phi_{ij}$. Note that while it is not the case that all flows obeying natural flow constraints can be realized as steady-state flows $\left\{f_{ij,m}(\mathbf{q})\right\}$ under some policy $\mathbf{q}$, all realized flows do have to obey flow conservation and capacity constraints. This motivates the following relaxation

$\left\{\widehat{q}_{ij}\right\}$ of the set of possible steady-state flows under any policy $\mathbf{q}$ and for any number of units m.

A natural capacity constraint arises since prices only decrease demand; the steady-state flow of units between a pair of nodes is thus bounded above by the rate of customers wanting to travel between the nodes. We refer to this constraint as *demand bounding*. Formally, for every pair (i, j), we have $f_{ij,m}(\mathbf{q}) \leq \phi_{ij}$ and hence

$$\widehat{q}_{ij} \in [0, 1].$$

Next, any steady-state flow must obey a natural flow conservation constraint, wherein the rate of incoming units at each node must equal the rate of outgoing units. We refer to this constraint as *supply circulation*. Formally, at any node i, we have $\sum_k f_{ki,m}(\mathbf{q}) = \sum_j f_{ij,m}(\mathbf{q})$, and hence

$$\sum_k \phi_{ki}\widehat{q}_{ki} = \sum_j \phi_{ij}\widehat{q}_{ij}.$$

Note that the above two constraints hold for every finite m and every $\mathbf{q}$; indeed, if they did not hold and the rate of incoming units to node i was larger than the rate of outgoing units then after letting the system run in steady-state for long enough, the number of units in i would be larger than m. Moreover, the constraints are also true for the infinite-unit limit (cf. Appendix 13.3). We refer to the set of flows defined by the above (linear) constraints as the *flow polytope*.

9.3.1 Pricing via the Elevated Flow Relaxation

Combining the elevated objective and the flow polytope, we obtain the *elevated flow relaxation program* (cf. Algorithm 2). Note that this is a convex optimization

problem since the objective function is concave while the polytope is linear; hence it can be efficiently maximized.

Algorithm 2: The Elevated Flow Relaxation Program

Require: arrival rates ϕ_{ij}, value distributions F_{ij}, reward curves R_{ij}.

1: Find $\left\{q_{ij}\right\}$ that solves the following relaxation:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{(i,j)} \phi_{ij} R_{ij}(\widehat{q}_{ij}) & \\ \textstyle\sum_k \phi_{ki} \widehat{q}_{ki} &= \textstyle\sum_j \phi_{ij} \widehat{q}_{ij} & \forall\ i \\ \widehat{q}_{ij} &\in [0,1] & \forall\ i, j. \end{aligned}$$

2: Output *state-independent* prices $p_{ij} = F_{ij}^{-1}(1 - q_{ij})$.

Note that the prices (quantiles) returned by Algorithm 2 impose the flow conservation not only on the units (supply) but also on the customers (demand); we henceforth refer to this property as *demand circulation*.

9.3.2 Approximation Framework

In this section, we provide the main approximation framework of the paper to bounds the performance of Algorithm 2 with respect to the optimal state-dependent pricing policy. This is formalized in the following theorem

Theorem 29. *Consider optimizing throughput for the m-unit system and let* $\widetilde{\mathbf{p}}$ *be the pricing policy returned by Algorithm 2 and* $\textsc{Opt}_m$ *be the value of the objective of the optimal state-dependent pricing policy in the m-unit system. Then*

$$\textsc{Obj}_m(\widetilde{\mathbf{p}}) \geq \frac{m}{m+n-1} \textsc{Opt}_m. \tag{9.6}$$

The proof is based on the following three lemmas. First (Lemma 30), we show that the objective of the optimal state-independent policy is upper bounded by the elevated objective of the policy $\widetilde{\mathbf{p}}$ returned by the Elevated Flow relaxation (Algorithm 2). Next (Lemma 31), we show that the elevated objective of $\widetilde{\mathbf{p}}$ is equal to its objective in the infinite-unit system. Finally (Lemma 32), we show that for any pricing policy (and so in particular for $\widetilde{\mathbf{p}}$), the objective in the m-unit system is within a factor of $\frac{m}{m+n-1}$ of the objective in the infinite-unit system. ■

Lemma 30. *For throughput, the value of the objective function of the optimal state-dependent policy is upper bounded by the value of the elevated objective function of the pricing policy $\widetilde{\mathbf{p}}$ returned by Algorithm 2*

$$\widehat{\textsc{Obj}}(\widetilde{\mathbf{p}}) \geq \textsc{Opt}_m.$$

Lemma 31. *The value of the elevated objective function of the pricing policy $\widetilde{\mathbf{p}}$ returned by Algorithm 2 is equal to the value of its objective function in the infinite-unit system*

$$\textsc{Obj}_\infty(\widetilde{\mathbf{p}}) = \widehat{\textsc{Obj}}(\widetilde{\mathbf{p}}).$$

Lemma 32. *For any state-independent pricing policy $\mathbf{p}$, the value of the objective of the policy $\mathbf{p}$ in the m-unit system is at least $m/(m+n-1)$ times the value of the objective of the same policy in the infinite-unit system.*

$$\textsc{Obj}_m(\mathbf{p}) \geq \frac{m}{m+n-1}\textsc{Obj}_\infty(\mathbf{p}).$$

We remark that one could prove Theorem 29 more easily by showing that Lemma 32 holds when $\mathbf{p}$ is a demand circulation. This has been known since the 1980s Whitt [1984] and, in fact, this is exactly the main theorem (and proof) of Waserhole and Jost [2014]. However, in later section we consider scenarios under which no demand circulation is optimal/feasible. For these, the stronger statement of

Lemma 32 is required. In the remainder of this section, we prove these three lemmas.

From finite-unit state-dependent to the elevated flow relaxation

Lemma 33 (Lemma 30 restated)**.** *For throughput, the value of the objective function of the optimal state-dependent policy is upper bounded by the value of the elevated objective function of the pricing policy* $\widetilde{\mathbf{p}}$ *returned by Algorithm 2*

$$\widehat{\text{OBJ}}(\widetilde{\mathbf{p}}) \geq \text{OPT}_m.$$

Proof. Let $\mathbf{q}^\star(\mathbf{X})$ denote the quantiles of the optimal state dependent policy and $\pi^\star(\mathbf{X})$ denote the steady-state distribution it induces. Then, since throughput is our objective, OPT_m can be written as

$$\sum_{\mathbf{X} \in \mathcal{S}_{n,m}} \pi^\star(\mathbf{X}) \sum_{i,j} \phi_{ij} q^\star_{ij}(\mathbf{X}).$$

We define $\widehat{\mathbf{q}}$ via

$$\widehat{q}_{ij} = \sum_{\mathbf{X} \in \mathcal{S}_{n,m}} \pi^\star(\mathbf{X}) q^\star_{ij}(\mathbf{X}).$$

Then $\text{OPT}_m = \sum_{i,j} \phi_{ij} \widehat{q}_{ij}$. Note that, by definition, $q^\star_{ij}(X) = 0$ when $X_i = 0$. Therefore, $\widehat{\mathbf{q}}$ satisfies the demand circulation and demand bounding constraints in the elevated flow relaxation program; this is due to (i) the state-dependent supply circulation property and (ii) $q^\star_{ij}(\mathbf{X}) \leq 1 \forall \mathbf{X}, i, j$. Hence $\widehat{\mathbf{q}}$ is a feasible solution to the elevated flow relaxation program and the result follows. ■

From the elevated flow relaxation to infinite-unit state-independent

Lemma 34 (Lemma 31 restated)**.** *The value of the elevated objective function of the pricing policy* $\widetilde{\mathbf{p}}$ *returned by Algorithm 2 is equal to the value of its objective function in*

the infinite-unit system

$$\textsc{Obj}_\infty(\widetilde{\mathbf{p}}) = \widehat{\textsc{Obj}}(\widetilde{\mathbf{p}}).$$

Proof. The pricing policy $\widetilde{\mathbf{p}}$ satisfies the demand circulation property since it is a feasible solution to the elevated flow relaxation program. By Lemmas 35 and Proposition 28, the availabilities at all nodes is equal to 1. This means that (i) the value of the objective function in the infinite-unit limit for pricing policy $\widetilde{\mathbf{p}}$ is equal to its elevated value (since no term was increased), and (ii) the flow of customers on each edge is equal to $\phi_{ij}\widetilde{q}_{ij}$. ■

Lemma 35. *For any m (including ∞) if state-independent quantiles $\mathbf{q}$ satisfies the demand circulation property then, at all nodes i, the availabilities $A_{i,m}(\mathbf{q})$ are equal.*

Proof. Consider $i^* \in \operatorname{argmax} A_{i,m}(\mathbf{q})$. Then the demand circulation and supply circulation properties imply

$$A_{i^*,m}(\mathbf{q}) \sum_j \phi_{ji^*} q_{ji^*} = A_{i^*,m}(\mathbf{q}) \sum_j \phi_{i^*j} q_{i^*j} = \sum_j A_{j,m}(\mathbf{q}) \phi_{ji^*} q_{ji^*}$$

and thus $\sum_j (A_{i^*,m}(\mathbf{q}) - A_{j,m}(\mathbf{q}))\phi_{ji^*} q_{ji^*} = 0$. By choice of i^*, each summand is nonnegative, so for each j such that $\phi_{ji^*} > 0$ we obtain $A_{j,m}(\mathbf{q}) = A_{i^*,m}(\mathbf{q})$. All availabilities being equal then follows inductively using connectivity of the underlying graph. ■

From finite-unit to infinite-unit state-independent

Lemma 36 (Lemma 32 restated)**.** *For any state-independent pricing policy $\mathbf{p}$, the value of the objective of the policy $\mathbf{p}$ in the m-unit system is at least $m/(m+n-1)$ times the value of the objective of the same policy in the infinite-unit system.*

$$\textsc{Obj}_m(\mathbf{p}) \geq \frac{m}{m+n-1} \textsc{Obj}_\infty(\mathbf{p}).$$

Proof. By Lemma 37, we have:

$$\frac{\mathrm{OBJ}_m(\mathbf{p})}{\mathrm{OBJ}_\infty(\mathbf{p})} = r_{\max}(\mathbf{p}) \cdot \frac{G_{m-1}(\mathbf{p})}{G_m(\mathbf{p})}.$$

In order to uniformly bound the above expression, the essential ingredient is the construction of a particular weighted biregular graph between the states in $\mathcal{S}_{n,m-1}$ and the states in $\mathcal{S}_{n,m}$. In this graph, non-zero edges only exist between neighboring states, i.e. between states $\mathbf{y} \in \mathcal{S}_{n,m-1}$ and $\mathbf{y} + e_i \in \mathcal{S}_{n,m}$; further, the total weight of edges incident to any state in $\mathcal{S}_{n,m}$ is equal to 1, and the total weight of edges incident to any state in $\mathcal{S}_{n,m-1}$ is equal to $\frac{m+n-1}{m}$. We construct such a graph in Lemma 38.

Throughout this proof, we use s for a state in $\mathcal{S}_{n,m-1}$ and t for one in $\mathcal{S}_{n,m}$. The weight of the edge (s,t) in the bipartite graph constructed in Lemma 38 is denoted by ω_{st}.

$$\begin{aligned}
\frac{\mathrm{OBJ}_m(\mathbf{p})}{\mathrm{OBJ}_\infty(\mathbf{p})} = r_{\max}(\mathbf{p}) \cdot \frac{G_{m-1}(\mathbf{p})}{G_m(\mathbf{p})} &= r_{\max}(\mathbf{p}) \frac{\sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}}{\sum_{t \in \mathcal{S}_{n,m}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{t_j}} \\
&= r_{\max}(\mathbf{p}) \cdot \frac{\sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}}{\sum_{t \in \mathcal{S}_{n,m}} \left(\sum_{s \in \mathcal{S}_{n,m-1}} \omega_{st}\right) \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{t_j}} \\
&= r_{\max}(\mathbf{p}) \cdot \frac{\sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}}{\sum_{(s,t) \in \mathcal{S}_{n,m-1} \times \mathcal{S}_{n,m}} \omega_{st} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j + (t_j - s_j)}} \\
&\geq \frac{\sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}}{\sum_{s \in \mathcal{S}_{n,m-1}} \left(\sum_{t \in \mathcal{S}_{n,m}} \omega_{st}\right) \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}} \\
&= \frac{\sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}}{\left(\frac{m+n-1}{m}\right) \sum_{s \in \mathcal{S}_{n,m-1}} \prod_{j=1}^n \left(r_j(\mathbf{p})\right)^{s_j}} = \frac{m}{m+n-1}
\end{aligned}$$

The third equality holds as $\sum_s \omega_{st} = 1$, while the second-to-last follows from $\sum_t \omega_{st} = \frac{m+n-1}{m}$. Crucially, $\omega_{st} > 0$ only holds for neighboring states s and t, which implies the inequality. ∎

Lemma 37. *For any state-independent pricing policy* $\mathbf{p}$*, let* $A_m(\mathbf{p}) = \max_i(A_{i,m}(\mathbf{p}))$ *denote the maximum steady-state availability across all nodes. Then the objective function of* $\mathbf{p}$ *in the m-unit system is related to the infinite-limit objective as*

$$\frac{\text{OBJ}_m(\mathbf{p})}{\text{OBJ}_\infty(\mathbf{p})} = r_{\max}(\mathbf{p}) \cdot \frac{G_{m-1}(\mathbf{p})}{G_m(\mathbf{p})} = A_m(\mathbf{p}).$$

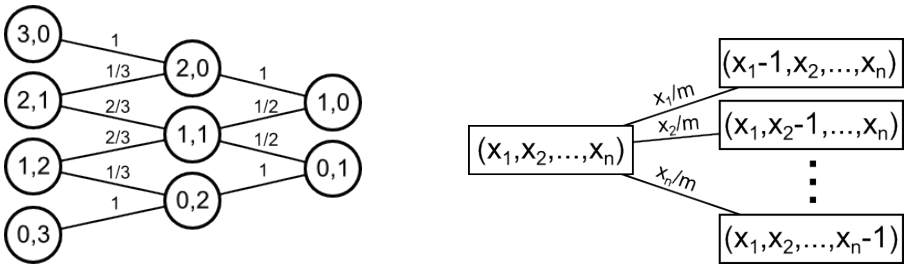
Proof. Let $B_i(\mathbf{p}) = \sum_j \phi_{ij} q_{ij} \cdot I_{ij}(q_{ij})$ denote the contribution of node i to the objective per unit of time in which station i is available. By substituting $A_{i,m}(\mathbf{p}) = (G_{m-1}(\mathbf{p})/G_m(\mathbf{p})) \cdot r_i(\mathbf{p})$, $A_{i,\infty}(\mathbf{p}) = r_i(\mathbf{p})/r_{\max}(\mathbf{p})$, and B_i into the definition of the objectives in Equation 9.4, we obtain

$$\frac{\text{OBJ}_m(\mathbf{p})}{\text{OBJ}_\infty(\mathbf{p})} = \frac{\sum_i A_{i,m}(\mathbf{p}) B_i(\mathbf{p})}{\sum_i A_{i,\infty}(\mathbf{p}) B_i(\mathbf{p})} = \frac{\frac{G_{m-1}(\mathbf{p})}{G_m(\mathbf{p})} \cdot \sum_i r_i(\mathbf{p}) B_i(\mathbf{p})}{\frac{1}{r_{\max}(\mathbf{p})} \cdot \sum_i r_i(\mathbf{p}) B_i(\mathbf{p})} = r_{\max}(\mathbf{p}) \cdot \frac{G_{m-1}(\mathbf{p})}{G_m(\mathbf{p})} = A_m(\mathbf{p}),$$

where the last equality follows from the characterization of the availabilities in Equation (9.3). Note that the argument relies on $OBJ_\infty(\mathbf{p}) > 0$ which is the case for all policies/settings we consider. ■

Lemma 38. *We call* $\mathbf{y} \in \mathcal{S}_{n,m-1}$ *a neighbor of* $\mathbf{y} + e_i \in \mathcal{S}_{n,m} \forall i$*. There exists a weighted biregular graph on* $\mathcal{S}_{n,m-1} \cup \mathcal{S}_{n,m}$ *such that i) an edge has non-zero weight only if it is connecting neighboring states, ii) for any vertex corresponding to a state in* $\mathcal{S}_{n,m-1}$ *the total weight of incident edges is equal to* $\frac{m+n-1}{m}$*, and iii) for any vertex corresponding to a state in* $\mathcal{S}_{n,m}$ *the total weight of incident edges is equal to* 1*.*

Proof. Our construction is shown in figure 9.2. Each state $\mathbf{x} \in \mathcal{S}_{n,m}$ is adjacent to $\mathbf{x} - e_i \in \mathcal{S}_{n,m-1}$ for all i with $x_i > 0$. On these edges, the weight is $\frac{x_i}{m}$. Thus, the total weight incident to $\mathbf{x}$ is $\sum_i \frac{x_i}{m} = 1$. On the other hand, each state $\mathbf{y} \in \mathcal{S}_{n,m-1}$ is adjacent to the states $\mathbf{y} + e_i \ \forall i \in [n]$. The respective weight on these edges is $\sum_i \frac{y_i+1}{m} = \frac{m-1+n}{m}$. Finally, there is only weight on edges between neighboring states. This concludes the proof of the lemma. ■



(a) Graph between $\mathcal{S}_{2,3}, \mathcal{S}_{2,2}$ and $\mathcal{S}_{2,1}$

(b) Construction for general n, m

Figure 9.2: **Biregular graph construction** as described in Lemma 38. Fig. 9.2(a) shows the construction for ($\mathcal{S}_{2,3}$,$\mathcal{S}_{2,2}$) and ($\mathcal{S}_{2,2}, \mathcal{S}_{2,1}$). Fig. 9.2(b) shows the general construction. Note that the sum of weights of incident edges for any node on the left (i.e. any state in $\mathcal{S}_{n,m}$) is 1, while it is $(m+n-1)/m$ for nodes on the right (i.e. states in $\mathcal{S}_{n,m-1}$).

9.3.3 Multi-objective Pricing

We begin this Section by extending the result of Theorem 29 to objectives other than throughput.

Theorem 39. *Consider optimizing any objective with concave reward curves $R_{ij}(\cdot)$ for the m-unit system and let $\widetilde{\mathbf{p}}$ be the pricing policy returned by Algorithm 2 and $\textsc{Opt}_m$ be the value of the objective of the optimal state-dependent pricing policy in the m-unit system. Then*

$$\textsc{Obj}_m(\widetilde{\mathbf{p}}) \geq \frac{m}{m+n-1}\textsc{Opt}_m. \tag{9.7}$$

The proof only requires replacing Lemma 30 by a Lemma that holds for all objectives with concave reward curves.

Lemma 40. *For objectives with concave reward curves $R_{ij}(\cdot)$, the value of the objective function of the optimal state-dependent policy is upper bounded by the value of the elevated objective function of the pricing policy $\widetilde{\mathbf{p}}$ returned by Algorithm 2*

$$\widehat{\textsc{Obj}}(\widetilde{\mathbf{p}}) \geq \textsc{Opt}_m.$$

Proof. Our proof applies Jensen's inequality to show that $\textsc{Opt}_m$ is bounded above by the elevated objective value of some quantiles $\widehat{\mathbf{q}}$ that form a feasible solution of the elevated flow relaxation program. Since the pricing policy $\widetilde{\mathbf{p}}$ maximizes this mathematical program, the lemma follows.

Let $\mathbf{q}^\star(\mathbf{X})$ denote the quantiles of the optimal state dependent policy and $\pi^\star(\mathbf{X})$ denote the steady-state distribution it induces. Then $\textsc{Opt}_m$ can be written as

$$\sum_{\mathbf{X}\in\mathcal{S}_{n,m}} \pi^\star(\mathbf{X}) \sum_{i,j} \phi_{ij} R_{ij}\Big(q^\star_{ij}(\mathbf{X})\Big).$$

We define $\widehat{\mathbf{q}}$ via

$$\widehat{q}_{ij} = \sum_{\mathbf{X}\in\mathcal{S}_{n,m}} \pi^\star(\mathbf{X}) q^\star_{ij}(\mathbf{X}).$$

Since the price-setting reward curve is concave, Jensen's inequality implies that

$$\textsc{Opt}_m \le \sum_{i,j} \phi_{ij} R_{ij}\Big(\widehat{q}_{ij}\Big).$$

By the same argument as in Lemma 30, $\widehat{\mathbf{q}}$ is a feasible solution to the elevated flow relaxation program and the result follows. ■

Bicriterion Approximations

We now discuss how to derive bicriterion approximations in multi-objective optimization settings, in which one objective is maximized subject to a lower bound on another. For ease of presentation, we restrict ourselves to pricing. Formally, the problem is as follows: we are given a m-unit system, a requirement $c \ge 0$, and objectives $\Phi_m(\cdot)$ and $\Psi_m(\cdot)$; the goal is to maximize $\Phi_m(\mathbf{q})$ subject to $\Psi_m(\mathbf{q}) \ge c$. We again assume that both objectives can be decomposed into per-ride rewards with associated concave reward curves $\left\{R^\Psi_{ij}\right\}$ and $\left\{R^\Phi_{ij}\right\}$.

Similarly to Equation (9.5), we first elevate both objectives to obtain $\widehat{\Phi}(\widehat{\mathbf{q}}) = \sum_{i,j} \phi_{ij} R^{\Phi}_{ij}(\widehat{q}_{ij})$ and $\widehat{\Psi}(\widehat{\mathbf{q}}) = \sum_{i,j} \phi_{ij} R^{\Psi}_{ij}(\widehat{q}_{ij})$. Since per-ride rewards are non-increasing on the quantiles, this can only increase the values of the objectives. We then impose the supply circulation and demand bounding constraints to create the flow polytope constraints. This mathematical program (Algorithm 3) is the elevated flow relaxation for our multi-objective setting; we argue below that this is indeed a relaxation. It can be efficiently optimized since the objective is concave while the polytope is convex: the convex combination of any two feasible quantiles is feasible since $\widehat{\Psi}(\cdot)$ is concave.

Algorithm 3: The Elevated Flow Relaxation for the Multi-objective Pricing Problem

Require: arrival rates ϕ_{ij}, value distributions F_{ij}, reward curves R^{Φ}_{ij} and R^{Ψ}_{ij}, requirement c.

1: Find $\left\{q_{ij}\right\}$ that solves the following relaxation:

$$\begin{aligned} \text{Maximize} \quad & \widehat{\Phi}(\widehat{\mathbf{q}}) \\ \textstyle\sum_k \phi_{ki} \widehat{q}_{ki} &= \textstyle\sum_j \phi_{ij} \widehat{q}_{ij} \quad \forall\, i \\ \widehat{q}_{ij} &\in [0,1] \qquad \forall\, i, j. \\ \widehat{\Psi}(\widehat{\mathbf{q}}) &\geq c \end{aligned}$$

2: Output *state-independent* prices $p_{ij} = F^{-1}_{ij}(1 - q_{ij})$.

Theorem 41. *Let Φ_m and Ψ_m be objectives for the m-unit system with concave reward curves. Then the solution $\widetilde{\mathbf{q}}$ returned by Algorithm 3 is a (γ, γ) bicriterion approximation for the multi-objective pricing problem where $\gamma = m/(m+n-1)$, i.e. $\Phi_m(\mathbf{q}^\star) \geq \gamma \textsc{Opt}_m$ and $\Psi_m(\mathbf{q}^\star) \geq \gamma \cdot c$.*

Proof. Let $\mathbf{q}'$ denote the optimal solution of an auxiliary program where we only

elevate objective Φ, i.e. we maximize $\widehat{\Phi}(\cdot)$ subject to $\Psi_m(\cdot) \geq c$ as well as the demand circulation and demand bounding constraints. Moreover, let $\mathbf{q}^\star$ denote the optimal solution of the original (non-elevated) program. Then, for the first guarantee, we have:

$$\Phi_m(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Phi}(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Phi}(\mathbf{q}') \geq \gamma\widehat{\Phi}(\mathbf{q}^\star) \geq \gamma\Phi(\mathbf{q}^\star) = \gamma\textsc{Opt}_m$$

The first inequality is a simple application of Theorem 39. The second inequality holds since any solution of the auxiliary program is a feasible solution of the elevated flow relaxation. In particular, since the elevated objective $\widehat{\Psi}_m(\cdot)$ is pointwise no less than the original objective $\Psi_m(\cdot)$, the corresponding constraint in the auxiliary program is tighter. The third inequality holds as $\mathbf{q}'$ is the optimal solution for the auxiliary program. The last inequality holds as the elevated objective $\widehat{\Phi}(\cdot)$ is pointwise no less than the original objective $\Phi(\cdot)$.

Regarding the second guarantee, we have:

$$\Psi_m(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Psi}(\widetilde{\mathbf{q}}) \geq \gamma c$$

The first inequality is again a simple application of Theorem 39 while the second holds since $\widetilde{\mathbf{q}}$ is a feasible solution of the elevated flow relaxation and therefore satisfies its last constraint. ■

We note that the same approach yields multicriterion approximation algorithms for settings in which more than one constraint of the form $\Psi_m(\cdot) \geq c$ is given.

CHAPTER 10

ADVANCED EXTENSIONS

In this chapter, we relax the restrictions we previously imposed. All of the algorithms and proofs for these extensions make use of our elevated flow relaxation framework of Section 9.3, demonstrating its generality. First, in Section 10.1, we allow the designer to have additional rebalancing controls beyond pricing by redirecting supply and demand. Next, in Section 10.2, we relax our assumption that changes in the state should be instantaneous by allowing travel-times for the trips. Last, in Section 10.3, we consider settings where prices can only depend on the source and where the prices are constrained to come from a discrete set. For all of these results, the proof follows from the same three steps as in Section 9.3.2.

10.1 Other Controls

Pricing is just one of several control levers in shared vehicle systems for balancing supply and demand; we now investigate two other levers, which we refer to as *supply redirection* and *demand redirection*, and show how they fit into our approximation framework. In the former we make a decision at the end of every trip on whether the unit remains at the destination of the trip or moves elsewhere whilst incurring a cost. In the latter, we redirect passengers arriving at a node to take units from nearby nodes. In practice, this would be achieved by pulling units from nearby nodes; for example in ridesharing services, the platform can dispatch a driver from a nearby node. Mathematically, the two are equivalent.

Supply Redirection

We consider a state-dependent policy $\mathbf{r}(\mathbf{X})$ which, for each trip ending at a node i, chooses to redirect the unit to some other node j (leading to state $\mathbf{X} - e_i + e_j$), else allows the unit to stay at i. For a state-independent policy, let $r_{ij} \in [0, 1]$ be the probability that an arriving unit at i is redirected to j. We assume that each redirection from i to j has associated cost c_{ij}, and that units arriving empty (redirected) are not redirected again.

With m units, a fixed pricing policy $\mathbf{p}$ (with corresponding quantiles $\mathbf{q}$), and a fixed redirection policy $\mathbf{r}$, we observe a rate $f_{ij,m}(\mathbf{q}, \mathbf{r})$ of customers traveling from i to j, and a rate of redirected vehicles $z_{ij,m}(\mathbf{q}, \mathbf{r})$ from i to j, i.e. trips with destination i which are redirected to j. For a state-independent policy, since each unit arriving at i is redirected to j with probability r_{ij}, it holds that

$$z_{ij,m}(\mathbf{q}, \mathbf{r}) = r_{ij} \sum_k f_{ki,m}(\mathbf{q}, \mathbf{r}).$$

Similarly to the correspondence between q_{ij} and $f_{ij,m}$, we observe a correspondence between r_{ij} and $z_{ij,m}$, wherein the former are the controls and induce the latter in the objective via the steady-state dynamics. As a result, the objective can be written as

$$\textsc{Obj}_m(\mathbf{q}, \mathbf{r}) = \sum_{i,j} f_{ij,m}(\mathbf{q}, \mathbf{r}) I_{ij}(\mathbf{q}) - c_{ij} z_{ij,m}(\mathbf{q}, \mathbf{r}).$$

In order to define the constraints of the elevated flow relaxation, we write (as in Section 9.3) $\widehat{q}_{ij} = f_{ij,m}(\mathbf{q}, \mathbf{r})/\phi_{ij}$ and $\widehat{z}_{ij} = z_{ij,m}(\mathbf{q}, \mathbf{r})$. We can now write the following relaxed flow polytope:

$$(1)\ \widehat{q}_{ij} \in [0, 1], \qquad (2)\ \sum_k \left(\phi_{ki} \widehat{q}_{ki} + \widehat{z}_{ki}\right) = \sum_j \left(\phi_{ij} \widehat{q}_{ij} + \widehat{z}_{ij}\right), \qquad (3)\ \sum_k \widehat{z}_{ik} \leq \sum_j \phi_{ji} \widehat{q}_{ji}\ \forall\ i.$$

The first constraint is demand bounding, exactly as explained in Section 9.3. The second is a variant of the supply circulation in Section 9.3 to incorporate

redirected vehicles. Finally, the third reflects that only units that are dropping off customers at a node, but not empty ones, can be redirected. Note that these constraints hold for any state-dependent policy as any policy induces such rates $f_{ij,m}$ and $z_{ij,m}$.

Using the reward curves $R_{ij}(\cdot)$ defined in Section 9.3, we obtain an upper bound $\widehat{\text{OBJ}}(\mathbf{q}, \mathbf{r})$ on our desired objective via the Elevated Flow Relaxation with the above constraints; through this, we obtain prices and redirection probabilities in Algorithm 4. Note that the redirection probabilities r_{ij} returned by the algorithm correspond to the rate of redirected units z_{ij} returned by the relaxation over the total incoming rate of (non-empty) units at node i, i.e. $\sum_k \phi_{kj} q_{kj}$. We now derive the equivalent of Theorem 39 to bound the performance of this algorithm.

Theorem 42. *Consider any objective function* OBJ_m *for the m-unit system with concave reward curves* $R_{ij}(\cdot)$. *Let* $\widetilde{\mathbf{p}}$ *and* $\widetilde{\mathbf{r}}$ *be the pricing and redirection policies returned by Algorithm 4, and* OPT_m *be the objective of the optimal state-dependent policies in the m-unit system. Then*

$$\text{OBJ}_m(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}) \geq \frac{m}{m+n-1} \text{OPT}_m.$$

Proof. The proof closely resembles that of Theorem 39. As before, we show the inequality through three intermediate steps: (i) $\widehat{\text{OBJ}}(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}) \geq \text{OPT}_m$, (ii) $\widehat{\text{OBJ}}(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}) = \text{OBJ}_\infty(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}})$, and (iii) $\text{OBJ}_m(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}) \geq \frac{m}{m+n-1} \text{OBJ}_\infty(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}})$. The proof of the first inequality is the same as in Lemma 40, with the relaxation defined in Algorithm 2 replaced by the relaxation defined in Algorithm 4. The second step relies on Lemma 43, which uses Lemma 35 to prove that in the infinite-unit system all availabilities are 1. Based on this claim, similarly to the proof of Lemma 31, we observe that the flow of customers on each edge is $\phi_{ij} \widetilde{q}_{ij}$. The definition of the redirection probabilities in Algorithm 4 then immediately implies that $z_{ij,\infty}(\widetilde{\mathbf{p}}, \widetilde{\mathbf{r}}) = z_{ij}$, i.e. the

Algorithm 4: The Elevated Flow Relaxation Program with Supply Redirection

Require: arrival rates ϕ_{ij}, value distributions F_{ij}, reward curves R_{ij}, rerouting costs c_{ij}.

1: Find $\{q_{ij}, z_{ij}\}$ that solves the the following relaxation:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i,j}\left(\phi_{ij} R_{ij}(\widehat{q}_{ij}) - c_{ij}\widehat{z}_{ij}\right) & \\ \textstyle\sum_k(\phi_{ki}\widehat{q}_{ki} + \widehat{z}_{ki}) &= \textstyle\sum_j\left(\phi_{ij}\widehat{q}_{ij} + \widehat{z}_{ij}\right) & \forall i \\ \textstyle\sum_k \widehat{z}_{ik} &\leq \textstyle\sum_j \phi_{ji}\widehat{q}_{ji} & \forall i \\ \widehat{q}_{ij} &\in [0,1] & \forall i, j \end{aligned}$$

2: Output *state-independent* prices $p_{ij} = F_{ij}^{-1}(1 - q_{ij})$ and redirection probabilities $r_{ij} = z_{ij} / \sum_k \phi_{ki} q_{ki}$

flow of redirected units from i to j is also equal to the value of z_{ij} in the solution of the relaxation. Finally, for the third step, we apply the same proof as in Lemma 32 with just one small modification. In Lemma 32, $B_i(\mathbf{p})$ denotes the contribution of node i per unit of time in which a unit is present at i. Previously, this just captured rides leaving node i. Now, we also charge $B_i(\mathbf{p})$ for the cost incurred through the possible redirection of vehicles traveling from i to j that are redirected to k. Replacing $B_i(\mathbf{p})$ by $\sum_j \phi_{ij} q_{ij}\left(I_{ij}\left(q_{ij}\right) - \sum_k r_{jk} p_{jk}\right)$ formalizes this charging argument – the remainder of the proof is equivalent to that of the Lemma 32. This concludes the proof of the theorem. ■

Lemma 43. *With $\widetilde{\mathbf{p}}$ and $\widetilde{\mathbf{r}}$ as returned by Algorithm 2, all availabilities are equal to one in the infinite-unit system.*

Proof. Denote by $\widetilde{\mathbf{q}}$ the quantiles corresponding to $\widetilde{\mathbf{p}}$. We consider a closed queueing network with the same transition probabilities between states as the one

resulting from $\widetilde{\mathbf{q}}$ and $\widetilde{\mathbf{r}}$. In our hypothetical network, quantiles are all one, there is no redirection, and the demand circulation property holds. Since the hypothetical network does not have redirection and satisfies the demand circulation property, Lemma 35 implies that there the availabilities at all nodes are equal. However, the two networks have the same transition probabilities so they also have the same steady-state distribution. As a result, in the original network all availabilities are also equal and thus, equal to 1 in the infinite-unit limit. We define the demand in the hypothetical network as

$$\bar{\phi}_{ij} = \phi_{ij}\widetilde{q}_{ij}(1 - \sum_k \widetilde{r}_{jk}) + \sum_k \phi_{ik}\widetilde{q}_{ik}\widetilde{r}_{kj}.$$

Observe that transitions occur at the same rate in this network as in the one with $\widetilde{\mathbf{q}}$ and $\widetilde{\mathbf{r}}$. Since quantiles are equal to 1, the demand circulation property says that $\sum_j \bar{\phi}_{ij} = \sum_k \bar{\phi}_{ki}$. To show this property, notice first that the demand at node i is

$$\sum_j \bar{\phi}_{ij} = \sum_j \phi_{ij}\widetilde{q}_{ij} - \sum_j \phi_{ij}\widetilde{q}_{ij}\left(\sum_k \widetilde{r}_{jk}\right) + \sum_j \sum_k \phi_{ik}\widetilde{q}_{ik}\widetilde{r}_{kj} = \sum_j \phi_{ij}\widetilde{q}_{ij}.$$

On the other hand, due to the definition of $\bar{\phi}_{ij}$ (first equality), the definition of $\widetilde{r}_{ij}$ in Algorithm 4 (third equality), and the supply circulation constraint in Algorithm 4 (last equality), the demand of customers traveling to i is

$$\begin{aligned}
\sum_k \bar{\phi}_{ki} &= \sum_k \phi_{ki}\widetilde{q}_{ki} - \sum_{j,k} \phi_{ki}\widetilde{q}_{ki}\widetilde{r}_{ij} + \sum_{j,k} \phi_{kj}\widetilde{q}_{kj}\widetilde{r}_{ji} \\
&= \sum_k \phi_{ki}\widetilde{q}_{ki} - \sum_j \widetilde{r}_{ij}(\sum_k \phi_{ki}\widetilde{q}_{ki}) + \sum_j \widetilde{r}_{ji}(\sum_k \phi_{kj}\widetilde{q}_{kj}) \\
&= \sum_k \phi_{ki}\widetilde{q}_{ki} - \sum_j \frac{z_{ij}}{\sum_k \phi_{ki}\widetilde{q}_{ki}}(\sum_k \phi_{ki}\widetilde{q}_{ki}) + \sum_j \frac{z_{ji}}{\sum_k \phi_{kj}\widetilde{q}_{kj}}(\sum_k \phi_{kj}\widetilde{q}_{kj}) \\
&= \sum_k \phi_{ki}\widetilde{q}_{ki} + \sum_j \left(z_{ji} - z_{ij}\right) = \sum_j \phi_{ij}\widetilde{q}_{ij}.
\end{aligned}$$

■

Demand Redirection

For the control defined in this section, we assume that there exists a graph $G = (V, E)$ on the set of nodes with edges between nodes that are so close that a customer arriving at one node can be served through a vehicle at an adjacent node. We consider a state-dependent policy $\mu(\mathbf{X})$ which, for each customer arriving at node i willing to pay the price quoted, decides from which node in $\{i\} \cup \{j : (i, j) \in E\}$, the customer is served. With m units, fixed quantiles $\mathbf{q}(\mathbf{X})$, and a fixed matching policy $\mu(\mathbf{X})$, we observe a rate $f_{ij,m}(\mathbf{q}, \mu)$ of customers arriving at i that travel to j, potentially after being matched to a unit at k, and a rate $z_{ik,m}(\mathbf{q}, \mu)$ of customers that arrived to travel from i but have been matched to a unit at k. We can write the objective in this setting as $\textsc{Obj}_m(\mathbf{q}, \mu) = \sum_{i,j} f_{ij,m}(\mathbf{q}, \mu) I_{ij}(\mathbf{q})$. We again write $\widehat{q}_{ij} = f_{ij,m}(\mathbf{q}, \mathbf{r})/\phi_{ij}$ and $\widehat{z}_{ij} = z_{ij,m}(\mathbf{q}, \mathbf{r})$ to define the following relaxed flow polytope:

$$(1)\ \widehat{q}_{ij} \in [0, 1], \qquad (2)\ \sum_k \widehat{q}_{ki}\phi_{ki} + \widehat{z}_{ik} = \sum_j \widehat{q}_{ij}\phi_{ij} + \widehat{z}_{ji}\ \forall\ i, \qquad (3)\ \sum_k \widehat{z}_{ki} \le \sum_j \widehat{q}_{ji}\phi_{ji}\ \forall\ i.$$

The first constraint is again demand bounding. The second is a variant of the supply circulation to incorporate matchings to nearby nodes. In particular, the left hand side accounts for the total number of units arriving at node i, which equals all users arriving at i together with all units arriving due to matching from nearby nodes k. Similarly, the right hand side accounts for the total number of units leaving i, which are the users leaving from i together with users from other nodes j that use supply at i. Finally, the third ensures that customers are matched only to units arriving at nearby nodes. Maximizing the elevated objectives over these constraints again yields a $m/(m + n - 1)$ approximation algorithm. We omit the proof, because of its similarity to the one of Theorem 42.

Theorem 44. *Solving for the elevated objective under the constraints defined above*

Algorithm 5: The Elevated Flow Relaxation Program With Matching

Require: arrival rates ϕ_{ij}, value distributions F_{ij}, reward-curves R_{ij}, edges E.

1: Find $\{q_{ij}, z_{ij}\}$ that solves the the following relaxation:

$$\begin{aligned}
\text{Maximize} \quad & \textstyle\sum_{i,j} \phi_{ij} R_{ij}(\widehat{q}_{ij}) & \\
\textstyle\sum_k (\phi_{ki}\widehat{q}_{ki} + \widehat{z}_{ik}) &= \textstyle\sum_j \left(\phi_{ij}\widehat{q}_{ij} + \widehat{z}_{ji}\right) & \forall i \in [n] \\
\textstyle\sum_k \widehat{z}_{ki} &\leq \textstyle\sum_j \phi_{ji}\widehat{q}_{ji} & \forall i \in [n] \\
\widehat{q}_{ij} &\in [0,1] & \forall i,j \in [n] \\
\widehat{z}_{ij} &= 0 & \forall (i,j) \notin E
\end{aligned}$$

2: Output *state-independent* prices $p_{ij} = F_{ij}^{-1}(1 - q_{ij})$ and matching probabilities $\mu_{ij} = z_{ij} / \sum_k \phi_{ik} q_{ik}$

yields a $m/(m+n-1)$ approximation algorithm for pricing and matching.

In Appendix 13.4 we show that the results obtained in this section continue to hold in settings, in which matching and/or redirecting is allowed, but pricing is not. In such scenarios, the optimal solution may not have the demand circulation property. Nevertheless, the same techniques yield $m/(m+n-1)$ approximation algorithms.

10.2 Incorporating travel-times between nodes

We now discuss how to remove the assumption that units move instantaneously by adding travel-times between nodes. We state our result only for pricing; however, our arguments below only depend on properties of the Markov chain, and hence can incorporate the other controls we consider.

A standard way to model travel-times is to assume that each unit takes an i.i.d. random time to travel from node i to j. Formally, we expand the network state to $\mathbf{X} = \left\{X_i(t), X_{ij}(t)\right\}$, where *node queues* $X_i(t)$ track the number of available units at node i, and *link queues* $X_{ij}(t)$ track the number of units in transition between nodes i and j. When a customer engages a unit to travel from i to j, the state changes to $\mathbf{X} - e_i + e_{ij}$ (i.e., $X_i \to X_i - 1$ and $X_{ij} \to X_{ij} + 1$). The unit remains in transit for an i.i.d. random time, distributed exponentially with mean τ_{ij} (this is primarily for ease of notation; our results extend if the travel time is distributed according to some general $G_{ij}(\cdot)$). When the unit reaches its destination, the state changes to $\mathbf{X} - e_{ij} + e_j$. Finally, we assume that pricing policies and passenger-side dynamics remain the same as before; in particular, we assume that the demand characteristics $\left\{\phi_{ij}, F_{ij}\right\}$ and reward-functions $\left\{I_{ij}\right\}$ are independent of the actual transit times (dependence on average transit times τ_{ij} can be embedded in the functions).

The system described above is a generalization of the Gordon-Newell network (Definition 26) referred to as a *BCMP network* (introduced by Baskett et al. [1975]; cf. Serfozo [1999], Section 3.3; also see Zhang and Pavone [2016] for the use of such a model for vehicle sharing). It is also a special case of a closed migration process; our presentation here follows Kelly and Yudovina Kelly and Yudovina [2014] (Chapter 2).

Definition 45. *A closed migration process on states $\mathcal{S}_{n^2,m}$ is a continuous-time Markov chain in which transitions from state $\mathbf{X}$ to state $\mathbf{X} - e_i + e_j$ occur at rate $\lambda_{ij}\mu_i(X_i)$ when $X_i > 0$ and at rate 0 otherwise. The λ_{ij} again form routing probabilities with $\sum_k \lambda_{ik} = 1, \lambda_{ij} \geq 0 \ \forall i, j$. Notice that $\mu_i(X_i)$ is a function of X_i only, whereas λ_{ij} are independent of the state alltogether.*

Given quantiles $\mathbf{q}$, the above-described process is a closed migration process with $\lambda_{i,ij} = \phi_{ij}q_{ij}/\sum_k \phi_{ik}q_{ik}$ and $\lambda_{ij,j} = 1$ for every i and j. Further, the service rate $\mu_i(X_i) = \sum_k \phi_{ik}q_{ik}$ when $X_i > 0$ for node queues and $\mu_{ij}(X_{ij}) = X_{ij}/\tau_{ij}$ for link queues. Intuitively, the latter captures the idea that each of the X_{ij} units has an exponential rate of $1/\tau_{ij}$ and therefore the rate until the first is removed from the link queue is X_{ij}/τ_{ij}. The stationary distribution can then be obtained as follows.

Theorem 46 (Theorem 2.4 in Kelly and Yudovina [2014])**.** *For a closed migration process as described in Definition 45, let $\{w_i\}_{i\in[n^2]}$ denote the invariant distribution associated with the routing probability matrix $\left\{\lambda_{ij}\right\}_{i,j\in[n]}$. Then the equilibrium distribution for a closed migration process is*

$$\pi(\mathbf{x}) = \frac{1}{G_m}\prod_{i=1}^{n^2}\frac{w_i^{x_i}}{\prod_{y=1}^{x_i}\phi_i(y)},$$

where $G_m = \sum_{\mathbf{x}}\prod_{i=1}^{n^2}\frac{w_i^{x_i}}{\prod_{y=1}^{x_i}\phi_i(y)}$ is a normalizing constant.

This implies for our setting, with $\mathbf{w}$ denoting again the invariant distribution of the routing matrix.

$$\pi_{\mathbf{x},m}(\mathbf{q}) = \frac{1}{G_m(\mathbf{q})}\left[\prod_{i\in[n]}\left(\frac{w_i(\mathbf{q})}{\sum_k \phi_{ik}q_{ik}}\right)^{x_i}\right]\left[\prod_{i,j\in[n]^2}\frac{\left(\tau_{ij}w_{ij}(\mathbf{q})\right)^{x_{ij}}}{x_{ij}!}\right]. \tag{10.1}$$

We remark that in comparison to the invariant distribution $\mathbf{w}^I$ when rides occur instantaneously, $\mathbf{w}^D$ with delays would be $w_i^D = w_i^I/2$ for node queues and $w_{ij}^D = \frac{w_i^D\phi_{ij}q_{ij}}{\sum_k \phi_{ik}q_{ik}}$ for link queues.

One consequence of the above characterization is that the resulting flows $f_{ij,m}(\mathbf{q})$ continue to satisfy demand bounding and supply circulation – consequently, the Elevated Flow Relaxation (cf. Algorithm 2) continues to provide an upper bound. Moreover, adding link queues does not affect the optimization

problems we consider in the infinite-unit system; in particular, Lemma 31 also continues to hold in this setting. Finally, from Lemma 37, we know that the ratio of objectives between the infinite-unit system and the finite-unit system equals the maximum availability, among all nodes, in the finite-unit system, i.e. $\frac{\text{OBJ}_m(\mathbf{q})}{\text{OBJ}_\infty(\mathbf{q})} = \max_i A_{i,m}(\mathbf{q})$. In order to obtain an approximation ratio, we now need to understand how $\max_i A_{i,m}(\mathbf{q})$ changes when link queues are added.

Let M denote the random variable corresponding to the steady-state number of *available* (i.e. not in transit) units across all nodes, and define $A_m(\mathbf{q}|M) \triangleq \max_{i\in[n]} \mathbb{P}[\mathbb{1}_{\{X_i>0\}}|M]$, $A_m(\mathbf{q}) = \max_{i\in[n]} A_{i,m}(\mathbf{q})$. Now we have the following

Lemma 47. *Conditioned on M, the distribution of $\{X_i\}_{i\in[n]}$ in the network with travel-times is identical to an n-node M-unit Gordon-Newell network with the same quantiles and arrival rates.*

This follows directly from the product-form nature of the steady-state distribution in Equation (10.1). Using this, we now obtain the following bound for the m-unit system availability.

Lemma 48. *For any network with parameters $\left\{\phi_{ij}, F_{ij}(\cdot), \tau_{ij}\right\}$ if $m \geq 100$ and quantiles $\mathbf{q}$ satisfy $\sum_{ij} \phi_{ij}\tau_{ij}q_{ij} \leq m - 2\sqrt{m\ln(m)}$ then*

$$A_m(\mathbf{q}) \geq \left(1 - \frac{3}{\sqrt{m}}\right)\left(\frac{\sqrt{m\ln m}}{\sqrt{m\ln m} + n - 1}\right).$$

Note that the above converges to 1 as $m \to \infty$.

Proof. First, for any given policy $\mathbf{q}$, as before we have the realized flows $f_{ij,m}(\mathbf{q}) = q_{ij}\phi_{ij}A_{i,m}(\mathbf{q})$; moreover, this is the expected rate of units entering link queue X_{ij}. Let $D = m - M$ be the number of units which are in transit. Now, by Little's law

(cf. Kelly [2011] or Serfozo [1999]), we have that the expected number of units in link queues is given by $\sum_{i,j} A_{i,m}(\mathbf{q})\phi_{ij}q_{ij}\tau_{ij}$.

Note that the link queues $\{X_{ij}\}$ are stochastically dominated by independent $M/M/\infty$ queues with input rate $\phi_{ij}q_{ij}$ and average transition time τ_{ij}. This follows from a simple coupling argument, where incoming customers follow an independent Poisson process of rate $\phi_{ij}q_{ij}$ and enter the link queue with a *virtual unit*, irrespective of whether the customer engages a unit or not in the real system. Thus D is stochastically dominated by $\widetilde{D} = Poi(\sum_{i,j} \phi_{ij}q_{ij}\tau_{ij})$. Further, since D is bounded above by m, D is also stochastically dominated by $\widehat{D} = \min\{\widetilde{D}, m\}$.

Next, from Lemma 47, we know that conditioned on there being M available units in the steady-state system, the distribution of units in node queues is identical to that of an n-node M-unit Gordon-Newell network; moreover, from Lemma 32, we have that for any n-node, m-unit Gordon-Newell network, $A_m(\mathbf{q}|M) \geq M/(M+n-1)$. Since $M = m - D$ and $(m-x)/(m+n-1-x)$ is decreasing in x for $x \leq m$, it follows that

$$A_m(\mathbf{q}) \geq \mathbb{E}\left[\frac{m-D}{m+n-1-D}\right] \geq \mathbb{E}\left[\frac{m-\widehat{D}}{m+n-1-\widehat{D}}\right]. \tag{10.2}$$

Further, by definition of $\widehat{D}$ we observe that $\mathbb{P}\left[\widehat{D} > m\left(1-\sqrt{\frac{\ln m}{m}}\right)\right] = \mathbb{P}\left[\widetilde{D} > m\left(1-\sqrt{\frac{\ln m}{m}}\right)\right]$. We can now apply a standard Chernoff bound for the Poisson random variable $\widetilde{D}$ (cf. from Lemma 70 in Appendix 13.6), using the assumption that $m - 2\sqrt{m\ln(m)} \geq \sum_{ij}\phi_{ij}\tau_{ij}q_{ij} = \mathbb{E}[\widetilde{D}]$. In particular, we may bound

$\mathbb{P}\left[\widehat{D} > m\left(1 - \sqrt{\frac{\ln m}{m}}\right)\right]$ by

$$\mathbb{P}\left[\widetilde{D} > m\left(1 - \sqrt{\frac{\ln m}{m}}\right)\right] \leq \exp\left(\frac{-m \ln m\left(m - 3\sqrt{m \ln m}\right)}{2(m - 2\sqrt{m \ln m})^2}\right) \tag{10.3}$$

$$= \exp\left(\frac{-\ln m \cdot \left(1 - 3\sqrt{\ln m/m}\right)}{2\left(1 - 4\sqrt{\ln m/m}\left(1 - \sqrt{\ln m/m}\right)\right)}\right)$$

$$\leq \exp\left(\frac{-\ln m\left(1 - 3\sqrt{\ln m/m}\right)}{2\left(1 - 1.5\sqrt{\ln m/m}\right)}\right) \tag{10.4}$$

$$= \exp\left(\frac{-\ln m}{2}\left(1 - \frac{3\sqrt{\ln m/m}}{\left(2 - 3\sqrt{\ln m/m}\right)}\right)\right)$$

(Since $\ln m/m \leq 1/e$, and $4 \cdot (1 - 1/\sqrt{e}) > 1.5$)

$$= \frac{1}{\sqrt{m}} \exp\left(\frac{3 \ln m}{4\sqrt{m/\ln m} - 6}\right) \leq \frac{3}{\sqrt{m}} \quad \text{for } m \geq 100.$$

We can use the above to bound the availability in Inequality (10.2) as

$$A_m(\mathbf{q}) \geq \left(1 - \frac{3}{\sqrt{m}}\right)\left(\frac{m - (m - \sqrt{m \ln m})}{m - (m - \sqrt{m \ln m}) + n - 1}\right) + \frac{3}{\sqrt{m}} \cdot 0.$$

Simplifying, we obtain the result. ■

We are now ready to extend our pricing/control policies to the setting with transit delays. In order to do so, we need to first extend the elevated flow relaxation by adding an extra constraint. The main observation is that in an m-unit system with transit delays, there is an additional *conservation constraint* induced by the fact that the number of units in the link queue can not exceed m. As before, let $f_{ij}^m(\mathbf{q}) = \widehat{q}_{ij}\phi_{ij}$ denote the expected rate of units entering link queue X_{ij}; then by Little's law (cf. Kelly [2011] or Serfozo [1999]), we have that the expected number of units in link queues is given by $\sum_{i,j} \phi_{ij}\widehat{q}_{ij}\tau_{ij}$, which, in an m-unit system, must be bounded by m. To incorporate this, we need to add an additional *rate-limiting constraint* to the elevated flow relaxation wherein we

ensure that $\sum_{i,j} \phi_{ij} \widehat{q}_{ij} \tau_{ij} \leq m$. This gives us the *Rate-Limited Elevated Flow Relaxation Program* in Algorithm 6.

Algorithm 6: The Rate-Limited Elevated Flow Relaxation Program

Require: arrival rates ϕ_{ij}, value distributions F_{ij}, reward curves R_{ij}, scaling parameter ε_m, travel-times τ_{ij}.

1: Find $\left\{q_{ij}\right\}$ that solves the following relaxation:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{(i,j)} \phi_{ij} R_{ij}(\widehat{q}_{ij}) & \\ \textstyle\sum_k \phi_{ki} \widehat{q}_{ki} &= \textstyle\sum_j \phi_{ij} \widehat{q}_{ij} & \forall\ i \\ \textstyle\sum_{i,j} \phi_{ij} \tau_{ij} \widehat{q}_{ij} &\leq m & \\ \widehat{q}_{ij} &\in [0,1] & \forall\ i, j. \end{aligned}$$

2: Set $\widetilde{q}_{ij} = q_{ij} \cdot (1 - \varepsilon_m)$

3: Output *state-independent* prices $\widetilde{p}_{ij} = F_{ij}^{-1}(1 - \widetilde{q}_{ij})$.

Theorem 49. *For any objective function* $\textsc{Obj}_m$ *with concave reward curves* $R_{ij}(\cdot)$ *in the m-unit system, let quantiles* $\widetilde{\mathbf{q}}$ *be the output of Algorithm 6 with input* $\varepsilon_m := 2\sqrt{\ln m / m}$, $\textsc{Opt}_m$ *be the value of the objective function for the optimal state-dependent pricing policy, and* $m \geq 100$. *Then*

$$\frac{\textsc{Obj}_m(\widetilde{\mathbf{q}})}{\textsc{Opt}_m} \geq (1 - \varepsilon_m)\left(\frac{\sqrt{m \ln m}}{\sqrt{m \ln m} + n - 1} - \frac{3}{\sqrt{m \ln m}} \right).$$

Proof. The proof follows a similar roadmap as that of Theorem 39. In particular, we argue that

1. the rate-limited elevated flow relaxation provides an upper bound for any state-dependent policy,

2. the rate-limited elevated flow relaxation solution is achieved by a state-independent policy in the infinite-unit system, and
3. the ratio of the performance of any state-independent policy $\mathbf{q}$ in the infinite-unit and m-unit system is equal to the maximum availability $A_m(\mathbf{q})$.

First, similar to Lemma 40, note that since the realized flows in the m unit system must obey the conservation laws encoded by the rate-limited elevated flow relaxation, hence $\textsc{Opt}_m$ is bounded by the solution to the rate-limited elevated flow relaxation $\sum_{(i,j)} \phi_{ij} R_{ij}(q_{ij})$. Moreover, since per-ride rewards $I_{ij}(\cdot)$ are non-increasing in $\mathbf{q}$, therefore scaling the q_{ij} by $(1 - \varepsilon_m)$ results in an elevated objective value that obeys

$$(1 - \varepsilon_m) \sum_{(i,j)} \phi_{ij} R_{ij}(q_{ij}) \leq \sum_{(i,j)} \phi_{ij} R_{ij}(\widetilde{q}_{ij}),$$

and moreover, $\sum_{i,j} \phi_{ij} \widetilde{q}_{ij} \tau_{ij} \leq m \cdot (1 - \varepsilon_m)$. Now, using similar arguments as in Lemma 31, we can show that using a state-independent policy $\widetilde{\mathbf{q}}$ in the infinite-unit limit gives $\textsc{Obj}_\infty(\widetilde{\mathbf{q}}) = \sum_{(i,j)} \phi_{ij} R_{ij}(\widetilde{q}_{ij})$ (note that we use the same $\widetilde{\mathbf{q}}$ as derived from the m unit rate-limited elevated flow relaxation in the infinite unit limit; in other words, we scale the number of units to infinite, but retain the constraint $\sum_{i,j} \phi_{ij} \tau_{ij} \widehat{q}_{ij} \leq m$ for a fixed m). Next, from Lemma 37, we get that $\textsc{Obj}_m(\widetilde{\mathbf{q}}) = A_m(\widetilde{\mathbf{q}}) \textsc{Obj}_\infty(\widetilde{\mathbf{q}})$. Finally, using Lemma 48, we get the desired bound

$$\frac{\textsc{Obj}_m(\widetilde{\mathbf{q}})}{\textsc{Opt}_m} \geq (1 - \varepsilon_m) \left(\frac{\sqrt{m \ln m}}{\sqrt{m \ln m} + n - 1} - \frac{3}{\sqrt{m}} \right).$$

■

Note that for any fixed n, the theorem shows that the policy returned by the rate-limited elevated flow relaxation is asymptotically optimal as $m \to \infty$ for *any demand rates and transit delays* $\left\{\phi_{ij}, \tau_{ij}\right\}$. In Appendix 13.4 we use this to recover

and give a finite-m characterization for the asymptotic results in Braverman et al. [2016], Ozkan and Ward [2016].

It is now natural to wonder whether the guarantee in Theorem 49 is tight. In particular, the rate of convergence is only in the square-root of m and as such it converges an order of magnitude slower than our guarantees without travel-times. It turns out that there are two answers to that question: in general, it is not possible to converge to the upper bound provided by the elevated flow relaxation faster than in the square-root of m – this holds for *any policy*. On the other hand, if the optimal solution $\vec{q}$ found in Algorithm 6 is not constrained by the rate-limitation, that is, it fulfills $\sum_{ij} \phi_{ij}\tau_{ij}\widehat{q_{ij}} = (1-c)m$ for some $c > 0$, then an asymptotically linear rate of convergence is recovered. We now first give an example in which no solution achieves a convergence faster than in the square-root of the number of vehicles and then prove that if the rate constraint is not tight, a linear rate of convergence is recovered.

Example 50. *Consider a system consisting of just a single node i, in which customers travel from i to itself and $\tau_{ii} = 1$. Suppose our objective is to maximize throughput; in this case, it is easy to see that setting $q_{ii} = 1$ is the optimal policy. Further, suppose that $\phi_{ii} = m$, i.e., the rate of demand is exactly equal to, and scales with, the supply available. In fact, using Equation 10.1 we can write the objective as follows:*

$$\textsc{Obj}_m = mA_{i,m}(\mathbf{q}) = m\left[1 - \left(\frac{\left(\frac{\frac{1}{2}}{m}\right)^0\left(\frac{\left(\frac{1}{2}\right)^m}{m!}\right)}{\sum_{x_i=0}^{m}\left(\frac{\frac{1}{2}}{m}\right)^{x_i}\left(\frac{\left(\frac{1}{2}\right)^{m-x_i}}{(m-x_i)!}\right)}\right)\right] = m\left[1 - \left(\frac{\left(\frac{1}{m!}\right)}{\sum_{x_i=0}^{m}\left(\frac{1}{m}\right)^{x_i}\left(\frac{1}{(m-x_i)!}\right)}\right).\right]$$

In order to prove our claim that no policy can obtain a rate of convergence that is faster than in the square-root of m, we need to bound the subtrahend by $\Omega(\frac{1}{\sqrt{m}})$:

$$\frac{\left(\frac{1}{m!}\right)}{\sum_{x_i=0}^{m}\left(\frac{1}{m}\right)^{x_i-m}\left(\frac{1}{(m-x_i)!}\right)}\frac{1}{m^m} = \frac{m^m}{m!}\frac{1}{\sum_{i=0}^{m}\frac{m^{x_i}}{x_i!}} \geq \frac{m^m}{m!}\frac{1}{\sum_{i=0}^{\infty}\frac{m^{x_i}}{x_i!}} = \frac{e^{-m}m^m}{m!} \in \Omega\left(\frac{1}{\sqrt{m}}\right).$$

Here, the final bound follows from Sterling's approximation since $m!$ scales as $m^{m+\frac{1}{2}}e^{-m}$.

Corollary 51. *Suppose that the optimal solution $\mathbf{q}^\star$ to the flow relaxation in Algorithm 6 is not constrained by the rate-limited constraint, that is, $\sum_{ij}\phi_{ij}\tau_{ij}q^\star_{ij} = (1-2c)m$ for some $c > 0$. Then, the rate of convergence to the upper bound is linear. Specifically, for $m \geq 100$,*

$$\frac{\textsc{Obj}_m(\mathbf{q}^\star)}{\textsc{Opt}_m} \geq \left(1 - e^{-\frac{c^2}{2}m}\right)\left(\frac{cm}{cm+n-1}\right).$$

Proof. By Lemma 70, we know that, with $\widehat{D}$ and $\widetilde{D}$ defined as in the proof of Lemma 48, we have

$$\mathbb{P}\left[\widehat{D} > m(1-c)m\right] = \mathbb{P}\left[\widetilde{D} > m(1-c)m\right] \leq e^{\left(-\frac{(cm)^2}{2(1-2c)m}\left(1-\frac{cm}{(1-2c)m}\right)\right)} = e^{\left(-\frac{c^2m}{2(1-c)}\left(1-\frac{c}{1-2c}\right)\right)}$$

Simplifying, we find that $\left(\frac{c^2}{2(1-c)}\left(1-\frac{c}{1-2c}\right)\right) \geq \frac{c^2}{2}$. Thus, with probability at least $\left(1-e^{-\frac{c^2}{2}m}\right)$, there are at least cm units not in transit. The rest of the proof is equivalent to that in Lemma 48 and Theorem 49. ∎

10.3 Constrained point pricing

In this section, we focus on a special case of the vanilla pricing problem wherein the platform is only allowed to set point prices, i.e. prices based on the origin node, and the value distributions of all customers arriving at a node are identical (i.e. $p_{ij} = p_i$, respectively $q_{ij} = q_i$, and $F_{ij}(\cdot) = F_i(\cdot)$ for all i, j). We provide a simple optimal pricing policy for the infinite-unit system, which involves just one eigenvector computation (for throughput/social welfare) or a concave maximization over a single variable (for revenue).

We then consider the additional constraint that prices are only allowed to come from a discrete price set. Using our infinite-to-finite unit reduction, all

our results are then translated back to the finite unit setting. We emphasize that in the latter restricted settings, there may not be a feasible solution satisfying demand circulation.

Unrestricted price set: We begin by providing the point pricing equivalent to Algorithm 2 and Theorem 39.

Algorithm 7: The Point Pricing Elevated Flow Relaxation Program

---

Require: arrival rates ϕ_{ij}, value distributions F_i, reward curves R_{ij}.

1: Find $\{q_i\}$ that solves the following point price relaxation:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{(i,j)} \phi_{ij} R_{ij}(\widehat{q_i}) & \\ \textstyle\sum_k \phi_{ki} \widehat{q_k} &= \textstyle\sum_j \phi_{ij} \widehat{q_i} & \forall\ i \\ \widehat{q_i} &\in [0,1] & \forall\ i. \end{aligned}$$

2: Output *state-independent* prices $p_i = F_i^{-1}(1 - q_i)$.

Theorem 52. *Consider any objective function* $\textsc{Obj}_m$ *for the m-unit system with concave reward curves* $R_{ij}(\cdot)$*. Let* $\widetilde{\mathbf{p}}$ *be the pricing policy returned by Algorithm 7,* $\textsc{Opt}_m$ *be the value of the objective function for the optimal state-dependent point pricing policy in the m-unit system. Then*

$$\textsc{Obj}_m(\widetilde{\mathbf{p}}) \geq \frac{m}{m+n-1} \textsc{Opt}_m \tag{10.5}$$

Proof. The proof is again based on three steps that compare $\textsc{Obj}_m(\widetilde{\mathbf{p}})$ with $\textsc{Obj}_\infty(\widetilde{\mathbf{p}})$, $\textsc{Obj}_\infty(\widetilde{\mathbf{p}})$ with $\widehat{\textsc{Obj}}(\widetilde{\mathbf{p}})$, and $\widehat{\textsc{Obj}}(\widetilde{\mathbf{p}})$ with $\textsc{Opt}_m$. The application of Jensen's inequality to prove $\widehat{\textsc{Obj}}(\widetilde{\mathbf{p}}) \geq \textsc{Opt}_m$ is the same as in Lemma 40, with the polytope in Algorithm 2 replaced by the one in Algorithm 7. Lemma 31 applies since its proof only relies on $\widetilde{\mathbf{p}}$ fulfilling the demand circulation property, which it does

(cf. Algorithm 7). Thus, $\textsc{Obj}_\infty(\widetilde{\mathbf{p}}) \geq \widehat{\textsc{Obj}}(\widetilde{\mathbf{p}})$. Finally, Lemma 32 implies that $\textsc{Obj}_m(\widetilde{\mathbf{p}}) \geq \frac{m}{m+n-1}\textsc{Obj}_\infty(\widetilde{\mathbf{p}})$, which concludes the proof of the theorem. ■

Notice that the optimization problem in Algorithm 7 has the demand circulation property as a constraint; thus, with the resulting pricing policy, the availability is equal at every node (cf. Lemma 35). Recall from Section 9.2.1 that the availability at each node in the infinite-unit system depends on the traffic intensity at that particular node and the maximum traffic intensity among all nodes. Further, the traffic intensity at each node i depends on (i) the ith coordinate of the eigenvector $\mathbf{w}(\mathbf{q})$ of the routing matrix $\{\phi_{ij}(q_i)/\sum_k \phi_{ik}(q_i)\}_{i,j\in[n]^2}$, and (ii) the rate of arrivals $\sum_k \phi_{ik}$ at i. In particular, $r_i(\mathbf{q}) = w_i(\mathbf{q})/\sum_j \phi_{ij}q_{ij}$. In the setting of point prices however, $\mathbf{w}$ is unaffected by the prices and $r_i(\mathbf{q}) = r_j(\mathbf{q})\forall i, j$ implies that $w_i \sum_k \phi_{jk}q_j = w_j \sum_k \phi_{ik}q_i$ for all i, j. Substituting in the optimization problem for every j

$$q_j = w_j \textstyle\sum_k \phi_{ik}q_i/w_i \sum_k \phi_{jk},$$

we find that the convex optimization problem can actually be written in just one variable. Further, in the case of social welfare, and revenue, it is always the case that $\max_i q_i = 1$ for an optimal solution in the infinite-unit system. Hence, in these cases only one eigenvector computation is needed.

Discrete price set: We now show how the pricing policy from Algorithm 7 can be modified when there is a discrete set of available prices for each node. We handle this case with an extra loss in the objective that depends on how well the prices represent each part of the distribution. In particular, we obtain the pricing policy $\widehat{\mathbf{p}}$ by solving for the unconstrained case as in Algorithm 7 to obtain prices

$\mathbf{p}$ and then setting each $\widehat{p_i}$ to be the lowest available price greater or equal to p_i. We now prove the performance guarantee for $\widehat{\mathbf{p}}$.

Theorem 53. *Let $\left\{p_i^1, \ldots, p_i^{k_i}\right\}$ be the set of available prices for node i in increasing order, $\left\{q_i^1, \ldots, q_i^{k_i}\right\}$ be the corresponding quantiles (in decreasing order), and $\mathbf{p}$, $\widehat{\mathbf{p}}$ be defined as above. Suppose that for all i there exists an available price p_i^ℓ such that $q_i^\ell \leq q_i$, and that there exists α such that for all i and all s, $\alpha \cdot q_i^s \geq q_i^{s+1}$. Then,*

$$\alpha \textsc{Obj}_m(\widehat{\mathbf{p}}) \geq \frac{m}{m+n-1} \textsc{Opt}_m,$$

where $\textsc{Opt}_m$ is the objective of the optimal state-dependent policy for discrete prices in the m-unit system.

Proof. Since $\widehat{\textsc{Obj}}(\mathbf{p})$ is an upper bound on the unrestricted point pricing problem (cf. Theorem 52), it is also an upper bound on $\textsc{Opt}_m$. Lemma 31 implies that $\widehat{\textsc{Obj}}(\mathbf{p}) = \textsc{Obj}_\infty(\mathbf{p})$, since $\mathbf{p}$ fulfills the demand circulation property (cf. Algorithm 7). Further, by Lemma 32, $\textsc{Obj}_m(\widehat{\mathbf{p}}) \geq \frac{m}{m+n-1} \textsc{Obj}_\infty(\widehat{\mathbf{p}})$. Thus, what remains is to bound $\textsc{Obj}_\infty(\widehat{\mathbf{p}})$ with respect to $\textsc{Obj}_\infty(\mathbf{p})$. Since $\widehat{q_i} \leq q_i$ for all i and the per-ride rewards $I_{ij}(\cdot)$ are assumed to be non-decreasing in the quantiles, we only need to bound the changes in the availabilities of the infinite-unit system for each i. Since the w_i are constant under point-pricing, the availabilities are only affected by prices in the denominator, where the change is equal to $\widehat{q_i}/q_i$. Thus, no traffic intensity changes by more than a factor of α and the result follows. ∎

The assumption that the value distributions at each node are identical may seem too restrictive. Notice though that the same analysis also applies to the following setting: for each i, j there exists a base price d_{ij} (e.g., based on geographic distance). This price is multiplied by the (state-dependent) control p_i, which is

the same for all j. The behavioral assumption is now that customers react the same way to the control, regardless of their destination.

CHAPTER 11

CONCLUSION

"What's Next?" – President Bartlet

This thesis began with a description of revolutionary changes that the personal transportation space has experienced in recent years. Associated with this revolution is a plethora of interesting operations research problems related to (i) new kinds of operational requirements and (ii) new opportunities enabled by data that previously had not been available. A few of these problems were tackled in this thesis, but many others are yet unsolved. Below, we list a number of open questions, some of theoretical interest, some of practical interest, that remain unanswered.

11.1 Open Questions

Dock Allocation. The optimization problem tackled in Chapter 4 relies on the user dissatisfaction functions (cf. Chapter 3) and their underlying assumption that demand is exogenous. Though our analysis in Section 4.5 partially moves beyond that assumption, we have yet to develop a formally sufficient condition that guarantees good solutions. Such a condition would provide interesting theoretical underpinnings for our optimization.

Further on the theoretical side, the construction we propose in Appendix 12.2 (to prove that our constrained optimization problem is not multimodular) poses an interesting question about constrained optimization for discrete convex sets. It turns out that, for this particular example, a relabeling of the stations (make

the first station the third and vice versa in $\vec{z}$) yields a multimodular problem even with the proposed constraints. Given that one would (intuitively) expect convexity properties to be invariant under a permutation of the ordering of the coordinates, this exhibits a fascinating property of the constrained set; further investigations of these properties, along the lines of Moriguchi and Murota [2018], will likely give rise to fascinating results.

Incentives. The use of incentives to rebalance bike-sharing system is a novel way of rebalancing that has shown great promise in New York City and San Francisco. Our results investigated, in a data-driven fashion, the tradeoffs between online and offline decision-making for such schemes, but many related questions remain. In the context of loyalty programs, there is a long line of research in the operations management literature studying the optimal reward structure of rewards: given the somewhat orthogonal goals of incentives in bike-sharing systems, it would be of interest to investigate whether the reward structure should be designed in a similar or an orthogonal fashion. A similar connection to the operations management literature on loyalty programs may exist in the context of cannibalization: given that Bike Angels are awarded either 1 or 2 points for rentals/returns at incentivized stations, it is natural to ask how stations with 2 points should be chosen and what the downside is of an approach that treats stations as independent of each other (as the user dissatisfaction functions do).

Rebalancing. There has been a tremendous amount of work on motorized rebalancing in the last several years. Nevertheless, it seems like none of the dock-based bike-sharing systems make use of specialized routing software that include demand characteristics in a non-trivial fashion (cf. de Chardon et al.

[2016]). This begs the question: what would it take to develop a tool that really works?

Budgeted Prize-Collecting TSP. In Chapter 7 we provided a 2-approximation algorithm for the budgeted prize-collecting traveling salesman problem that has at its base a classic primal-dual approach. An obvious open question seeks to improve the approximation guarantee or prove the current guarantee is the best possible. Another interesting extension asks to what extend our approach extends to the rooted case; since Paul et al. [2017] appeared, the authors have extended the same algorithm/analysis to the rooted case, in which a feasible solution is forced to include a particular (root) vertex. Compared to the orienteering literature, this would be the special case of the $s - s$ orienteering problem; an interesting question is whether a similar technique can also be used for the $s - t$ orienteering problem.

Queueing Models. The use of the elevated flow relaxation poses interesting questions about the relationship between combinatorial optimization and stochastic control; though we show how to obtain parameterized approximation guarantees (and thereby asymptotic optimality) for a large class of controls in closed queueing networks, it may be possible to apply similar techniques in other stochastic models.

11.2 Thoughts on Industry and Academia

This thesis would have been a very different document if it had not been for the longstanding collaboration between Cornell's bike-sharing research group and Motivate. Having such a collaboration allowed us to be certain that the

questions we were asking were not only of theoretical interest, but also of practical relevance. While not all research needs to fill both of these boxes, the data and the transportation revolutions that we currently experience provide plenty of problems that do; for academia, this yields a huge opportunity to play an important role in shaping the societal changes caused by these new technologies.

Part III

Appendices

CHAPTER 12

APPENDIX TO CHAPTER 1

12.1 Connections to M-Convex Functions

In this appendix we first provide the definitions of M-convex sets and functions, and then show that our objective with budget constraints is not M-convex. For the definitions, it is useful to denote $supp^+(\vec{x}-\vec{y}) = \{i : x_i > y_i\}$, $supp^-(\vec{x}-\vec{y}) = \{i : x_i < y_i\}$, and $\vec{e}_i$ as the canonical unit vector.

Definition 54 (M-convex set)**.** *A nonempty set of integer points $B \subseteq Z^{2n}$ is defined to be an M-convex set if it satisfies $\forall \vec{x}, \vec{y} \in B, i \in supp^+(\vec{x}-\vec{y}), \exists j \in supp^-(\vec{x}-\vec{y}) : \vec{x} - \vec{e}_i + \vec{e}_j \in B$.*

Definition 55 (M-convex function)**.** *A function f is M-convex if for all $x, y \in dom(f), i \in supp^+(x-y), \exists j \in supp^-(x-y) : f(x)+f(y) \geq f(x-e_i+e_j)+f(y+e_i-e_j)$.*

Kaspi et al. [2017] prove a statement equivalent to $c(\cdot,\cdot)$ being M-convex. Murota [2004] characterized the minimum of a M convex function as follows to show that Algorithm 8 minimizes M-convex functions:

Lemma 56. *Murota [2003] For an M-convex function f and $x \in dom(f)$ we have $f(x) \leq f(y)\ \forall y$ if and only if $f(x) \leq f(x-e_i+e_j) \forall i, j$.*

As our example shows, the restriction of c to the feasible set (with budget constraints) does not guarantee M-convexity, despite both the set and c being M-convex.

Algorithm 8: M-convex function minimization, cf. Murota [2004]

0: Find a vector $x \in dom(f)$

1: Find i, j that minimize $f(x - e_i + e_j)$

2: If $f(x) > f(x - e_i + e_j)$, set $x := x - e_i + e_j$ and go to 2

3: Else, return x

Example 57. *Our example consists of three stations i, j, and k with demand-profiles:*

$$p_i(-1) = \frac{1}{2},\ p_i(+1,-1) = \frac{1}{2};\quad p_j(+1) = \tfrac{1}{2};\quad p_k(+1,-1,-1) = 1.$$

We consider two solutions. In the first, i, j, and k each have a dock allocated with i also having a bike allocated, i.e., $b'_i = d'_j = d'_k = 1$, whereas $d'_i = b'_j = b'_k = 0$ and our budget constraint is $D = 2, B = 1$. Then $c_i(d'_i, b'_i) = \frac{1}{2}$, $c_j(d'_j, b'_j) = 0$, and $c_k(d'_k, b'_k) = 1$. In the second solution, $d^*_i = b^*_k = d^*_k = 1$, whereas $b^*_i = d^*_j = b^*_j = 0$. Thus, we have $c_i(d^*_i, b^*_i) = \frac{1}{2}$, $c_j(d^*_j, b^*_j) = \frac{1}{2}$, and $c_k(d^*_k, b^*_k) = 0$, giving that $1 = c(\vec{d}^*, \vec{b}^*) < c(\vec{d}', \vec{b}') = \frac{3}{2}$. But then the statement of Lemma 56 with $y = (\vec{d}^*, \vec{b}^*)$ and $x = (\vec{d}, \vec{b})$ implies that, if c is M-convex one of $c((\vec{d}'_{-i}, d'_i + 1), c(\vec{d}', (\vec{b}'_{-i,-k}, b'_i - 1, b'_k + 1))$, or $c((\vec{d}'_{-i,-j}, d'_i + 1, d'_j - 1), \vec{b}')$ must be strictly smaller than $c(\vec{d}', \vec{b}')$. Since this is not the case, we find that c restricted to the feasible set is not M-convex, even though the underlying feasible set is M-convex.

12.2 Connections to Discrete Midpoint Convex Functions

In this appendix we show that the constrained optimization problem formulated in Section 4.1 is not multimodular. To do so, we apply an equivalence proven in Murota [2005] that characterizes a function f as multimodular if and only if

there exists a L-natural convex function g such that $f(x_1, x_2, \ldots, x_n) = g(x_1, x_1 + x_2, \ldots, \sum_{i=1}^{n} x_i)$. While we do not state the explicit definition of L-natural convex functions here, it was shown by Fujishige and Murota [2000] that L-natural convex functions fulfill the following discrete midpoint convexity property.

Definition 58. *A function $g : \mathbb{Z}^n \to \mathbb{R}^n \cup \{+\infty\}$ is called discrete midpoint convex if*

$$g(x) + g(y) \geq g(\lceil \frac{x+y}{2} \rceil) + g(\lfloor \frac{x+y}{2} \rfloor).$$

Here, the floor and ceiling refer to component-wise floor and ceiling.

We now argue that the function g corresponding to our (constrained) objective c is not discrete midpoint convex. Consider the current allocation (cf. Section 4.1) $\vec{d} = (0, 1, 0, 1)$ and $\vec{b} = (0, 0, 0, 0)$. As all values for $\vec{b}$ are 0 throughout this construction, we do not restate it from now on. Suppose $z = 1$, that is, only one dock is allowed to be moved. Then the vector $\vec{d} = (1, 0, 1, 0)$ is not feasible given the constraint (as it would involve moving 2 docks). Now, if g was discrete midpoint convex, then the inequality would state that:

$$f(1, 0, 0, 1) + f(0, 1, 1, 0) = g(1, 1, 1, 2) + g(0, 1, 2, 2) \geq$$

$$g(1, 1, 2, 2) + g(0, 1, 1, 2) = f(1, 0, 1, 0) + f(0, 1, 0, 1).$$

However, both terms on the left-hand side are feasible whereas the first term on the right-hand side is not. Thus, the inequality does not hold, g is not discrete midpoint convex, and therefore f is not multimodular.

12.3 Tradeoff between number of reallocated and new docks

In this appendix, we show that the discrete gradient-descent algorithm can be applied, with little overhead, to solve the following adaptation of the earlier

optimization problem: rather than having fixed budgets $D + B$ and $2z$, we have a parameter M that bounds (via an additional parameter $k > 1$) the joint cost of reallocating docks and acquiring new docks.

$$
\begin{aligned}
\texttt{minimize}_{(\vec{d},\vec{b}),z,\bar{D}} \quad & c(\vec{d},\vec{b}) && \\
s.t. \quad & \textstyle\sum_i d_i + b_i && \leq D + B + \bar{D}, \\
& \textstyle\sum_i b_i && \leq B, \\
& \textstyle\sum_i |(\bar{d}_i + \bar{b}_i) - (d_i + b_i)| && \leq 2z + \bar{D}, \\
\forall i \in [n]: \quad & l_i \leq\ d_i + b_i && \leq u_i \\
& z + k\bar{D} && \leq M.
\end{aligned}
$$

For each fixed pair of values of z and $\bar{D}$, the discrete-gradient descent algorithm finds an optimal solution by the analysis in Section 4.2. Furthermore, it is easily observed that for each value of $\bar{D}$, it is optimal to set $z = M - k\bar{D}$. Hence, one way of finding an optimal solution would be to try all $\lfloor \frac{M}{k} \rfloor$ feasible values of $\bar{D}$ (and corresponding values of z) and solve optimally with the corresponding value of z.

A better algorithm to find the optimal solution is based on the following observation: by Theorem 10, the dock-move distance between the optimal allocation for $\bar{D}$ and $z = M - k\bar{D}$ on the one hand and the one for $\bar{D}$ and $z = M - k(\bar{D} + 1)$ on the other is at most $\bar{D}$. Hence, we only need to bound the distance to an optimal solution that has an additional empty dock at its disposal. It is a simple corollary of the analysis in Section 4.2.4 that the dock-move distance from an optimal solution for a given budget to an optimal solution with one additional dock available is in fact bounded by 1.

The reasoning above implies that the gradient-descent algorithm with a minimal adaptation can be used to solve the optimization problem that includes a tradeoff between the cost of new docks and the cost of reallocating docks; however, in practice this tradeoff barely ever arises, since the relative cost of new inventory greatly outweighs that of reallocating existing industry.

CHAPTER 13

APPENDIX TO CHAPTER 2

13.1 Irreducibility of the Priced System

We justify here our assumption from Section 9.2 that the infinite-unit solutions we obtain induce a connected graph; to do so, we first need to assume that the graph created by edges (i, j) on which $\phi_{ij} > 0$ is strongly connected. We then prove that given any solution to the infinite-unit pricing problem, there exists a solution with arbitrarily close objective that also induces a connected graph. Throughout this section we work with the flow $f_{ij,\infty}(\mathbf{p})$ induced by the demands in the infinite-unit system, but suppress all dependencies on ∞ in the notation.

Theorem 59. *Let $\epsilon > 0$. For any non-decreasing objective and any pricing policy $\mathbf{p}$ that induces a supply circulation f_{ij} on k components in the infinite-unit system, there exists a policy $\mathbf{p}'$ inducing a supply circulation f'_{ij} in the infinite-unit system such that the graph with edge-set $E = \{(i, j) : f'_{ij} > 0\}$ is strongly connected and the objective with $\mathbf{p}'$ is at least $(1 - \epsilon)$ times that of $\mathbf{p}$.*

Proof. To prove the theorem we repeatedly add flow to edges (i, j) with $f_{ij} = 0$, but also take flow away from edges $(\bar{i}, \bar{j})$ with $f_{\bar{i}\bar{j}} > 0$. To ensure that edges of the second kind do not have their flow reduced by too much, we set

$$\delta = \frac{\epsilon}{k} \times \min\left\{ \min_{i,j}\left\{ f_{ij} : f_{ij} > 0 \right\}, \min_{i,j}\left\{ \phi_{ij} : \phi_{ij} > 0 \right\} \right\}.$$

Whenever we decrease flow on an edge, this is done by an additive δ amount. Reducing flow at most k times to obtain f'_{ij} we guarantee that $f'_{ij} \geq (1 - \epsilon) f_{ij}$ holds.

As we assume our underlying graph with edge-set $\{(i,j) : \phi_{ij} > 0\}$ to be strongly connected, it must be the case that there exists a minimal sequence of components $C_1, C_2, \ldots, C_d = C_1$, $d > 2$, and nodes $u_\ell, v_\ell \in C_\ell$ such that $\lambda_{u_\ell v_{\ell+1}} > 0$, but $f_{u_\ell v_{\ell+1}} = 0$. In particular, it being minimal implies that no component other than the first appears repeatedly.

Since each u_ℓ, v_ℓ are in the same strongly connected component of the graph with edge-set E, we know that for each ℓ there exists a simple path from u_ℓ to v_ℓ with positive flow on it. We change flows as follows: for all pairs $(u_\ell, v_{\ell+1})$ we increase flow by δ and for each edge along the path from u_ℓ to v_ℓ we decrease flow by δ. At all other edges the flow remains unchanged.

We need to first argue that the new circulation is feasible. Each node along a path within a component has its in-flow and out-flow reduced by δ, whereas at the nodes u_i, v_i both the sum of in-flows and the sum of out-flows has remained the same. At all other nodes, nothing is altered. Thus, flow conservation continues to hold. By choice of δ none of the edge-capacities are violated. Thus, the resulting flow is a circulation with at most $k-1$ distinct components. Applying this procedure $k-1$ times, we obtain a single strongly connected component.

Finally, since $I_{ij}(\cdot)$ are nondecreasing with price and decreasing flow is equivalent to increasing prices, the choice of δ guarantees that the objective on paths from u_ℓ to v_ℓ has been reduced by at most a factor of $(1-\epsilon)$. Since $I_{ij}(\cdot)$ are non-negative, the additional flow on edges from u_ℓ to $v_{\ell+1}$ only increases the total objective. Thus, the pricing policy $\mathbf{p}'$ that induces the circulation f'_{ij} has the desired properties. ■

13.2 Concave Reward Curves

In this section, we investigate conditions under which throughput, social welfare and revenue satisfy the conditions of theorem 39. In particular, we first show that the respective reward curves $R(q) = qI(q)$ are concave. We then prove that the concave reward curves assumption implies the non-increasing (quantiles) per-ride rewards assumption.

Lemma 60. *Revenue (i) satisfies the assumptions of Theorem 39 under regular value distributions, Throughput (ii) and Social Welfare (iii) satisfy the assumptions under any value distribution.*

Proof. We drop the subscripts throughout this proof to simplify notation. We begin by considering (i) revenue, for which the result holds due to the fact that the reward curve is concave if and only if the distribution is regular (cf. Proposition 3.10 in Hartline [2016]). For (ii) throughput, $R(q) = q \cdot I(q) = q$ is a linear function of q for any value distribution and thus concave.

Lastly, for (iii) social welfare, we use the so-called hazard rate $h(y) = \frac{f(y)}{1-F(y)}$ of a distribution F with density f. Given F, denote by $p(q)$ and $q(p)$ a price as a function of its corresponding quantile and vice-versa. Then, by the definition of hazard rate:

$$q(p) = \exp\left(-\int_0^{p(q)} h(y)dy\right) \tag{13.1}$$

Taking logarithms and differentiating, we obtain:

$$-\frac{1}{q(p)} = h(p(q))\frac{dp(q)}{dq} \tag{13.2}$$

Hence, as $R(q(p)) = q(p) \cdot I(q(p))$ and $f(p) = (1 - F(p))h(p) = q(p)h(p)$ we have

$$R(q) = \int_{p(q)}^{\infty} vf(v)dv = \int_{p(q)}^{\infty} vh(v)\exp\left(-\int_0^{v} h(y)dy\right)dv$$

The first derivative $\frac{dR(q)}{dq}$ of $R(q)$ is equal to

$$-p(q)h(p(q))\exp\left(-\int_{y=0}^{p(q)} h(y)dy\right)\frac{dp(q)}{dq} = \frac{p(q)\exp\left(-\int_{y=0}^{p(q)} h(y)dy\right)}{q(p)} = p(q),$$

where the first equality comes from Equation (13.2), the second from (13.1).

The second derivative is then given by

$$\frac{d^2R(q)}{dq^2} = \frac{dp(q)}{dq} = -\frac{1}{qh(p(q))} = -\frac{1-F(p(q))}{f(p(q))q(p)} < 0,$$

which concludes the proof of the Lemma. ■

Lemma 61. *If some objective satisfies the concave reward curves assumption, it also satisfies the non-increasing (in quantiles) per-ride rewards assumption.*

Proof. Suppose an objective has concave reward curves, but does not have non-increasing (in quantiles) per-ride reards. Then there must exist i, j, q_1, q_2 with $0 < q_1 < q_2$ such that $I_{ij}(q_1) < I_{ij}(q_2)$. Let $A = \frac{q_1}{q_2}$. Then

$$\begin{aligned} q_1 I_{ij}(q_2) &= A \cdot q_2 I_{ij}(q_2) = A \cdot q_2 I_{ij}(q_2) + (1-A)\cdot 0 \cdot I_{ij}(0) \\ &\leq (A \cdot q_2 + (1-A)\cdot 0) I(A \cdot q_2 + (1-A)\cdot 0) = q_1 I_{ij}(q_1), \end{aligned}$$

where the inequality follows from Jensen's inequality on since the rewards curve $qI_{ij}(q)$ is a concave function. As $q_1 > 0$, it follows that $I_{ij}(q_2) \leq I_{ij}(q_1)$ and we therefore arrive at a contradiction. ■

13.3 Infinite-unit Limit

In Section 9.2.1 we briefly introduced the infinite-unit limit of the Gordon-Newell network, i.e., the characterization of the limiting Markov chain wherein we

keep all system parameters (ϕ_{ij}, F_{ij}, etc.) constant, and scale $m \to \infty$. We also mentioned that the primary result we use from this characterization is that the steady-state availability of each node i is given by $A_i i, \infty(\mathbf{p}) = r_i(\mathbf{p}) / \max_j r_j(\mathbf{p})$, and that there exists at least one node i with $A_{i,\infty}(\mathbf{p}) = 1$ (cf. Proposition 28). We now describe this limit in a little more detail. Our presentation follows closely that of Serfozo [1999], Section 3.7, which we refer the reader to for more details.

Recall first that given $\mathbf{p} = \{p_{ij}\}$, we can compute quantities $w_i(\mathbf{p})$ and $r_i(\mathbf{p})$, which are independent of m. We define $r_{\max} = \max_i r_i(\mathbf{p})$ and $\widehat{r_i}(\mathbf{p}) = r_i(\mathbf{p})/r_{\max}$. We also define $J = \{i \in [n] \,|\, \widehat{r_i}(\mathbf{p}) = 1\}$ to be the set of *bottleneck nodes* in the network (note that J has at least one element), and $K = [n] \setminus J$ be the remaining nodes. Then as $m \to \infty$, the stationary distribution of the m-unit system (as specified in Equation (9.2)) converges to a limiting distribution (cf. Serfozo [1999] for the specific technical sense in which the steady-state distributions converge to the limit) as $m \to \infty$, with the following properties:

- The bottleneck nodes, i.e., nodes in set J with $\widehat{r_i}(\mathbf{p}) = 1$, all have $A_i(\mathbf{p}) = 1$.
- The bottleneck nodes feed the non-bottleneck nodes in set K, which together form an open Jackson network, with each node behaving as a stable $M/M/1$ queue.
- For all $i \in K$, we have $A_i(\mathbf{p}) = \widehat{r_i}(\mathbf{p}) < 1$.

The above description has the following physical interpretation: in the infinite-unit limit, the bottleneck nodes have an infinite queue of units, and hence always have availability 1. Moreover, the rate of units traveling from one of these nodes i to a non-bottleneck node j is exactly $\phi_{ij}(\mathbf{p})$. Thus from the perspective of a non-bottleneck node j, it appears as if a steady-stream of units (with total rate

$< \phi_j(\mathbf{p})$) arrive from (and depart to), an external node; the number of units in node j therefore behaves according to the dynamics of a stable $M/M/1$ queue.

Lemma 62. *The objective of the elevated flow relaxation for the policy returned by Algorithm 2 upper bounds the objective of any state-independent policy* $\mathbf{p}$ *in the infinite-unit system.*

Proof. This follows if we show that the flows in the infinite-unit limit satisfy supply circulation and demand bounding. The latter is clear from the dynamics of the system (the flow out of a node can not exceed the rate of arriving customers). To see that the former follows from the above listed properties, note that $w_i(\mathbf{p})$ is defined to be the leading left eigenvector of $\{\lambda_{ij}(\mathbf{p})\}_{i,j}$, where $\lambda_{ij}(\mathbf{p}) = \phi_{ij}(\mathbf{p})/\phi_i(\mathbf{p})$. From this we get for all i:

$$\sum_j w_j \frac{\phi_{ji}(\mathbf{p})}{\phi_j(\mathbf{p})} = w_i = w_i \left(\sum_k \frac{\phi_{ik}(\mathbf{p})}{\phi_i(\mathbf{p})} \right) \Rightarrow \sum_j r_j(\mathbf{p}) \phi_{ji}(\mathbf{p}) = \sum_k r_i(\mathbf{p}) \phi_{ik}(\mathbf{p})$$

Dividing both sides by $r_{\max}(\mathbf{p})$ we get that for all nodes i, we have $\sum_j \widehat{r}_j(\mathbf{p}) \phi_{ji}(\mathbf{p}) = \sum_k \widehat{r}_i(\mathbf{p}) \phi_{ik}(\mathbf{p})$. However, as we noted above, $A_{i,\infty}(\mathbf{p}) = \widehat{r}_i(\mathbf{p})$, and hence $f_{ij}^{\infty}(\mathbf{p}) = \widehat{r}_i(\mathbf{p}) \phi_{ij}(\mathbf{p})$. Thus the $f_{ij}^{\infty}(\mathbf{p})$ satisfy flow conservation. ■

Combining with Lemma 31, we get that the elevated flow relaxation solution is tight in the infinite-unit limit.

Lemma 63. *The objective of the elevated flow relaxation for the policy returned by Algorithm 2 is equal to the objective of the optimal state-independent policy in the infinite-unit system.*

13.4 Settings without Prices

In Section 10.1 we discussed how two control levers, redirection of supply and of demand, can be combined with pricing to obtain the same guarantees we obtain for the pure pricing problem. We now show that our technique extends to settings in which only redirection of supply/demand is allowed, but pricing is not. Because demand cannot be modulated in these settings, one may assume that I_{ij} is constant for each i and j, because I_{ij} is not a function of prices. Thus, the elevated objective, defined analogously to Section 9.3, is always equal to the objective now. Further, the interpretation of our results changes slightly.

Similarly to Algorithm 7, we introduce quantiles q_i; unlike Section 10.3 however, we cannot change prices to modulate demand according to these quantiles. We adopt the same notation as in Section 10.1, with the exception that we do not allow for pricing policies and thus everything is just a function of $\mathbf{r}$. The quantiles $\mathbf{q}$ now correspond to the induced availabilities, i.e., $q_i = A_{i,m}(\mathbf{r})$. Observe that the resulting flows are within the following polytope (as in Sections 10.1 and 10.3):

$$(1)\ \widehat{q_i} \in [0,1], \qquad (2)\ \sum_k (\phi_{ki}\widehat{q_k} + \widehat{z_{ki}}) = \sum_j \left(\phi_{ij}\widehat{q_i} + \widehat{z_{ij}}\right), \qquad (3)\ \sum_k \widehat{z_{ik}} \leq \sum_j \phi_{ji}\widehat{q_j}\ \forall\ i.$$

As in Section 10.1, these constraints stem from demand bounding, supply circulation, and the limitation that only non-empty arriving vehicles may be rebalanced. Optimizing the elevated objective over the polytope given by these constraints is a linear program and yields an upper bound on the objective. Consider the redirection policy $\widetilde{\mathbf{r}}$ obtained from the solution of the linear program (cf. Algorithm 9). In the next Lemma, we bound the infinite unit performance of this policy compared to the value of the elevated flow relaxation.

Lemma 64. *Denote by $\widehat{\mathbf{q}}$ the quantiles solved for in the relaxation of Algorithm 9 and*

Algorithm 9: The Elevated Flow Relaxation Program for Redirection without Prices

Require: arrival rates ϕ_{ij}, per-ride rewards I_{ij}, rerouting costs c_{ij}.

1: Find $\{q_i, z_{ij}\}$ that solves the the following relaxation:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i,j}\left(\phi_{ij}\widehat{q_i}I_{ij} - c_{ij}\widehat{z_{ij}}\right) & \\ \textstyle\sum_k(\phi_{ki}\widehat{q_k} + \widehat{z_{ki}}) &= \textstyle\sum_j\left(\phi_{ij}\widehat{q_i} + \widehat{z_{ij}}\right) & \forall i \\ \textstyle\sum_k \widehat{z_{ik}} &\leq \textstyle\sum_j \phi_{ji}\widehat{q_j} & \forall i \\ \widehat{q_i} &\in [0,1] & \forall i \end{aligned}$$

2: Output redirection probabilities $r_{ij} = z_{ij} / \sum_k \phi_{ki} q_{ki}$

by $\widetilde{\mathbf{r}}$ the redirection probabilities returned. Then $\textsc{Obj}_\infty(\widetilde{\mathbf{r}}) \geq \widehat{\textsc{Obj}}(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$.

Proof. Consider first $\textsc{Obj}_\infty(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$, the objective obtained when implementing both the redirection policy $\widetilde{\mathbf{r}}$ and the quantiles $\widehat{\mathbf{q}}$ that Algorithm 9 solves for. By the same argument as in Lemma 43, all availabilities are equal to 1 (and all traffic intensities are equal) in this system, and thus its objective matches $\widehat{\textsc{Obj}}(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$. In order for us to compare $\textsc{Obj}_\infty(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$ with $\textsc{Obj}_\infty(\widetilde{\mathbf{r}})$, consider a node $v \in \arg\max_j \widehat{q}_j$. Increasing each quantile by a factor of $1/\widehat{q_v}$, we obtain quantiles $\bar{\mathbf{q}}$. Notice that in the system with quantiles $\bar{\mathbf{q}}$, the traffic intensity at each node is changed by the same factor, so the traffic intensities are still equal and the availabilities are still equal at every node. In fact, for the relaxation in Algorithm 9, there exists at least one i such that $q_i = 1$, so no quantile changes. Allowing for delays and scaling demand with the number of units, this would not necessarily be the case. Thus, $\textsc{Obj}_\infty(\bar{\mathbf{q}}, \widetilde{\mathbf{r}}) \geq \widehat{\textsc{Obj}}(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$. Thereafter, for each node $j \neq v$, we increase its quantile to 1. Notice that each such change only decreases the traffic intensity at j, so the maximum traffic intensity remains unchanged. The lemma follows because the decrease in the traffic intensity (and thus availability) at each node $j \neq v$ is

exactly balanced by the increased rate of arrivals at j. Formally, we have that $f_{jk,\infty}(\bar{\mathbf{q}}, \widetilde{\mathbf{r}})$ remains unchanged when the jth coordinate of the quantiles is set to 1. Therefore, $\textsc{Obj}_\infty(\widetilde{\mathbf{r}}) = \textsc{Obj}_\infty(\bar{\mathbf{q}}, \widetilde{\mathbf{r}}) \geq \widehat{\textsc{Obj}}(\widehat{\mathbf{q}}, \widetilde{\mathbf{r}})$. ■

Now, using Lemma 64 in place of Lemma 43 in the proof of Theorem 42, we get the following.

Theorem 65. *With $\widetilde{\mathbf{r}}$ defined as above, $\textsc{Obj}_m(\widetilde{\mathbf{r}}) \geq \frac{m}{m+n-1}\textsc{Opt}_m$.*

13.4.1 Delays without prices

Accommodating settings in which we are not allowed pricing, but do have delays, requires an additional idea. This is because the argument in Section 10.2 explicitly relied on pricing to ensure that (on average) not too many units are in transit simultaneously, thereby enabling a lower bound on the maximum availability. Without prices to regulate demand, we can no longer control the maximum availability. Instead, we use the following stochastic dominance characterization for closed-queueing networks.

Lemma 66 (cf. Theorem 3.8 in Chen and Yao Chen and Yao [2013])**.** *In a closed Jackson network, with state-independent service rates, increasing the service rate functions, in a pointwise sense, at any subset of nodes will increase throughput.*

In our context, this is equivalent to saying that increasing quantiles at a subset of nodes only increases throughput. In fact, one can show that throughput also increases locally, i.e., increasing quantiles at one node (which we henceforth refer to as *point quantiles*) does not decrease the rate of units on any edge.

Lemma 67. *Let* $\mathbf{q} = \{q_i\}$ *be a vector of point quantiles, and* $\widetilde{\mathbf{q}}$ *be a vector of point quantiles with* $\widetilde{q}_k \geq q_k \; \forall \, k$*. Then for any pair* (i, j)*, we have* $f_{ij,m}(\mathbf{q}) \leq f_{ij,m}(\widetilde{\mathbf{q}})$*, i.e. the rate of realized trips from* i *to* j *does not decrease when point quantiles are increased.*

Proof. The proof relies on two observations. Note first for $\mathbf{q}$ and $\widetilde{\mathbf{q}}$, we have

$$\frac{\phi_{ij} q_i}{\sum_k \phi_{ik} q_i} = \frac{\phi_{ij} \widetilde{q}_i}{\sum_k \phi_{ik} \widetilde{q}_i} \; \forall \, i, j,$$

and therefore, letting $\mathbf{w}(\mathbf{q}')$ denote the eigenvector of the routing matrix $\{\phi_{ij}(q_i')/\sum_k \phi_{ik}(q_i')\}_{i,j\in[n]^2}$ (cf. Section 10.3), we obtain $w_i(\widetilde{\mathbf{q}}) = w_i(\mathbf{q})$. Define $\Gamma_m(\mathbf{q}) \triangleq G_m(\mathbf{q})/G_{m-1}(\mathbf{q})$. We now have that the ratio of the rates $f_{ij,m}(\mathbf{q})/f_{ij,m}(\widetilde{\mathbf{q}})$ is equal to

$$\frac{f_{ij,m}(\mathbf{q})}{f_{ij,m}(\widetilde{\mathbf{q}})} = \frac{A_{i,m}(\mathbf{q}) q_i \phi_{ij}}{A_{i,m}(\widetilde{\mathbf{q}}) \widetilde{q}_i \phi_{ij}} = \frac{\Gamma_m(\mathbf{q}) r_i(\mathbf{q}) q_i}{\Gamma_m(\widetilde{\mathbf{q}}) r_i(\widetilde{\mathbf{q}}) \widetilde{q}_i} = \frac{\Gamma_m(\mathbf{q}) \frac{w_i(\mathbf{q})}{\sum_k \phi_{ik} q_i} q_i}{\Gamma_m(\widetilde{\mathbf{q}}) \frac{w_i(\widetilde{\mathbf{q}})}{\sum_k \phi_{ik} \widetilde{q}_i} \widetilde{q}_i} = \frac{\Gamma_m(\mathbf{q})}{\Gamma_m(\widetilde{\mathbf{q}})}.$$

Note that the ratio of $f_{ij,m}(\mathbf{q})$ and $f_{ij,m}(\widetilde{\mathbf{q}})$ does not depend on i and j. Moreover, from Theorem 66 we have $\sum_{i,j} f_{ij,m}(\mathbf{q}) \leq \sum_{i,j} f_{ij,m}(\widetilde{\mathbf{q}})$. Combining the two, we get $f_{ij,m}(\mathbf{q}) \leq f_{ij,m}(\widetilde{\mathbf{q}})$. ∎

This allows us to prove the guarantee of Theorem 49 for settings in which prices cannot be used to provide a lower on the maximum availability within the system.

Theorem 68. *Let* $\widetilde{\mathbf{r}}$ *denote the output of Algorithm 10 with* $\varepsilon_m \coloneqq 2\sqrt{\ln m/m}$*,* $\textsc{Opt}_m$ *be the value of the objective function for the optimal state-dependent pricing policy, and* $m \geq 100$*. Then*

$$\frac{\textsc{Obj}_m(\widetilde{\mathbf{r}})}{\textsc{Opt}_m} \geq (1 - \varepsilon_m)\left(\frac{\sqrt{m \ln m}}{\sqrt{m \ln m} + n - 1} - \frac{3}{\sqrt{m \ln m}}\right).$$

Proof. The same proof as in Theorem 49 guarantees that using point prices as given by $\mathbf{q}(1-\epsilon_m)$, where $\mathbf{q}$ comes from the solution of the relaxation in Algorithm

Algorithm 10: The Rate-Limited Elevated Flow Relaxation Program for Redirection w/o Prices

Require: scaling paramter ϵ_m, arrival rates ϕ_{ij}, rewards I_{ij}, rerouting costs c_{ij}, travel-times τ_{ij}.

1: Find $\{q_i, z_{ij}\}$ that solves the the following relaxation:

$$\begin{aligned}
\text{Maximize} \quad & \textstyle\sum_{i,j}\left(\phi_{ij}\widehat{q_i}I_{ij} - c_{ij}\widehat{z_{ij}}\right) & \\
\textstyle\sum_{i,j}\phi_{ij}\tau_{ij}q_i + \widehat{z_{ij}} \quad &\leq \quad m & \\
\textstyle\sum_k(\phi_{ki}q_k + \widehat{z_{ki}}) \quad &= \quad \textstyle\sum_j\left(\phi_{ij}\widehat{q_i} + \widehat{z_{ij}}\right) & \forall i \\
\textstyle\sum_k \widehat{z_{ik}} \quad &\leq \quad \textstyle\sum_j \phi_{ji}\widehat{q_j} & \forall i \\
\widehat{q_i} \quad &\in \quad [0,1] & \forall i
\end{aligned}$$

2: Output redirection probabilities $r_{ij} = z_{ij} / \sum_k \phi_{ki}q_{ki}$

10 yields the required guarantee. Lemma 67 then guarantees that increasing all quantiles to one yields a solution no worse. ■

We remark that with $m \to \infty$, the above theorem recovers the result of Braverman et al Braverman et al. [2016].

Finally, we note that Lemma 67 also yields an alternate proof of Lemma 32: given quantiles $\mathbf{q}$ that do not induce a demand circulation, we consider a system with rates $\widetilde{\phi}_{ij} = \phi_{ij}\frac{\max_k r_k(\mathbf{q})}{r_i(\mathbf{q})}$. We observe that (i) the objectives with rates $\widetilde{\phi}_{ij}$ and rates ϕ_{ij} are the same in an infinite unit system and (ii) that the system with rates $\widetilde{\phi}_{ij}$ obeys the demand circulation property. Thus, the counting argument of Whitt [1984] guarantees an objective within $m/(m+n-1)$ of the infinite unit system in a system with rates $\widetilde{\phi}_{ij}$. However, by (i) the latter was equal to the upper bound on $\textsc{Opt}_m$. Since Lemma 67 implies that the m-unit system with rates ϕ_{ij} has objective no worse than the m-unit system with rates $\widetilde{\phi}_{ij}$, the statement of the

lemma follows.

13.5 Tightness Of Our Guarantees

In this section, we discuss an example of Waserhole and Jost [2014], that proves that the guarantees we prove for our algorithms are tight. Interestingly, this does not require the distinction between state-dependent and state-independent policies, i.e. the objectives obtained through our algorithms can be as far away from the optimal state-independent policy as from the optimal state-dependent polcy.

Proposition 69. *(Waserhole and Jost [2014]) For any number m of units and n of nodes, the objective of the solution returned in Algorithm 2 and the optimal objective may be arbitrarily close to the approximation guarantee* $\frac{m}{m+n-1}$.

Proof. Consider a system of n nodes $\{1, \ldots, n\}$ with demand only occurring from nodes i to $i+1$ and from node n to node 1. In particular, suppose that for some k that is yet to be set, we have $\phi_{12} = \phi_{23} = \ldots = \phi_{n-1\,n} = k$, and $\phi_{n1} = 1$. Further, suppose we are maximizing throughput, though the same construction works for revenue and social welfare. The policy returned by Algorithm 2 sets quantiles $q_{12} = q_{23} = \ldots = q_{n-1\,n} = \frac{1}{k}$ and $q_{n1} = 1$. Given that the availability of each node is then $\frac{m}{m+n-1}$ (cf. Lemma 32 with all inequalities holding tightly) and that there are n nodes from which a ride can occur (at rate 1), the throughput is $\frac{nm}{m+n-1}$. On the other hand, for the solution that sets all quantiles to 1, the throughput converges to n as $k \to \infty$. Intuitively, this is because the expected time between an arrival at node n (triggering that unit to move to node 1) and the expected return time of that unit to node n converges to 0. Thus, for each arrival at node n, occurring at

rate 1, the system observes m rides. The details of this argument can be found in Proposition 3 of Waserhole and Jost [2014]. ■

13.6 Auxiliary lemma

We present a basic Chernoff tail bound for Poisson random variables, which we use in Section 10.2

Lemma 70. *For $X \sim Poisson(\lambda)$, we have for any $0 \leq x \leq \lambda$:*

$$\mathbb{P}[X > \lambda + x] \leq \exp\left(-\frac{x^2}{2\lambda}\left(1 - \frac{x}{\lambda}\right)\right)$$

Proof. Using a standard Chernoff bound argument, we have for any $\theta \geq 0$:

$$\mathbb{P}[X > \lambda + x] = \mathbb{P}[e^{\theta X} > e^{\theta(\lambda+x)}] \leq e^{-\theta(\lambda+x)}\mathbb{E}\left[e^{\theta X}\right] = e^{-\theta(\lambda+x)} \cdot e^{\lambda(e^{\theta}-1)}$$

Now, optimizing over the choice of θ, we get

$$\begin{aligned}
\mathbb{P}[X > \lambda + x] &\leq \exp\left(\inf_{\theta}\left(\lambda\left(e^{\theta} - 1 - \theta\right) - x\theta\right)\right) \\
&= \exp(x - (x+\lambda)\log(1 + x/\lambda)) \quad \text{(Setting } \theta = \log(1 + x/\lambda)\text{)} \\
&\leq \exp\left(x - (x+\lambda)\left(\frac{x}{\lambda} - \frac{x^2}{2\lambda^2}\right)\right) = \exp\left(-\frac{x^2}{2\lambda}\left(1 - \frac{x}{\lambda}\right)\right)
\end{aligned}$$

■

BIBLIOGRAPHY

Daniel Adelman. Price-directed control of a closed logistics queueing network. *Operations Research*, 55(6):1022–1038, 2007.

Eitan Altman, Bruno Gaujal, and Arie Hordijk. Multimodularity, convexity, and optimization properties. *Mathematics of Operations Research*, 25(2):324–347, 2000.

Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. *SIAM Journal on Computing*, 40(2):309–332, 2011.

Sunil Arya and Hariharan Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65(3):117–118, 1998.

G. Ausiello, M. Demange, L. Laura, and V. Paschos. Algorithms for the on-line quota traveling salesman problem. *Information Processing Letters*, 92(2):89–94, 2004.

Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 517–517. ACM, 2017.

Siddhartha Banerjee, Yash Kanoria, and Pengyu Qian. The value of state dependent control in ridesharing systems. *arXiv preprint arXiv:1803.04959*, 2018.

Forest Baskett, K Mani Chandy, Richard R Muntz, and Fernando G Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM (JACM)*, 22(2):248–260, 1975.

Mike Benchimol, Pascal Benchimol, Benoît Chappert, Arnaud De La Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. Balancing the stations of a self service bike hire system. *RAIRO-Operations Research*, 45(1):37–61, 2011.

New York City Bikeshare, 2016.

Bird. Charger – bird, 2018. URL `https://www.bird.co/charger`.

Avrim Blum, Ramamurthy Ravi, and Santosh Vempala. A constant-factor approximation algorithm for the k-MST problem. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 442–448. ACM, 1996.

Anton Braverman, JG Dai, Xin Liu, and Lei Ying. Empty-car routing in ridesharing systems. *arXiv preprint arXiv:1609.07219*, 2016.

James D. Brooks, Koushik Kar, and David Mendona. Dynamic allocation of entities in closed queueing networks: An application to debris removal. In *Proceedings of the 2013 IEEE International Conference on Technologies for Homeland Security*, pages 504–510, 2013.

Teobaldo Bulhões, Anand Subramanian, Güneş Erdoğan, and Gilbert Laporte. The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*, 264:508–523, 2018.

Jeffrey P Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16(9):527–531, 1973.

Capital Bikeshare. 2014 capital bikeshare member survey report, 2014.

Chandra Chekuri and Nitish Korula. Approximation algorithms for orienteering with time windows. *arXiv preprint arXiv:0711.4825*, 2007.

Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 245–253. IEEE, 2005.

Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8 (3):23, 2012.

Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2): 120–146, 2013.

Hong Chen and David D Yao. *Fundamentals of queueing networks: Performance, asymptotics, and optimization*, volume 46. Springer Science & Business Media, 2013.

Ke Chen and Sariel Har-Peled. The orienteering problem in the plane revisited. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 247–254. ACM, 2006.

Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, and Jérémie Jakubowicz Thi-Mai-Trang Nguyen. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 841–852. ACM, 2016.

Hangil Chung, Daniel Freund, and David B. Shmoys. Bike Angels: An Analysis of Citi Bike's Incentive Program . In *Proceedings of the 2018 Conference on Computing and Sustainable Societies*. ACM, 2018.

Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. *Balancing a dynamic public bike-sharing system*, volume 4. Cirrelt, 2012.

Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. Setting inventory levels in a bike sharing network. *Transportation Science*, 2017.

Cyrille Médard de Chardon, Geoffrey Caruso, and Isabelle Thomas. Bike-share rebalancing strategies, patterns, and purpose. *Journal of Transport Geography*, 55:22–39, 2016.

Mauro Dell'Amico, Eleni Hadjicostantinou, Manuel Iori, and Stefano Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. Constraint-based approaches for balancing bike sharing systems. In *International Conference on Principles and Practice of Constraint Programming*, pages 758–773. Springer, 2013.

Güneş Erdoğan, Gilbert Laporte, and Roberto Wolfler Calvo. The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451–457, 2014.

Güneş Erdoğan, Maria Battarra, and Roberto Wolfler Calvo. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245(3):667–679, 2015.

Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1): 21 – 41, 2001. ISSN 0022-0000.

Iris A Forma, Tal Raviv, and Michal Tzur. A 3-step math heuristic for the static

repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological*, 71:230–247, 2015.

Greg N. Frederickson and Barry Wittman. Approximation algorithms for the traveling repairman and speeding deliveryman problems. *Algorithmica*, 62 (3-4):1198–1221, 2012.

Daniel Freund, Ashkan Norouzi-Fard, Alice Paul, Shane G. Henderson, and David B. Shmoys. Data-driven rebalancing methods for bike-share systems. Working paper, 2016.

Daniel Freund, Shane G. Henderson, and David B. Shmoys. Minimizing multimodular functions and allocating capacity in bike-sharing systems. In F. Eisenbrand and J. Koenemann, editors, *Integer Programming and Combinatorial Optimization Proceedings*, volume 10328 of *Lecture Notes in Computer Science*, pages 186–198. Springer, 2017. arXiv preprint arXiv:1611.09304.

Satoru Fujishige and Kazuo Murota. Notes on l-/m-convex functions and the separation theorems. *Mathematical Programming*, 88(1):129–146, 2000.

Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8): 999–1020, 1994.

N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 302–, Washington, DC, USA, 1996. IEEE Computer Society.

Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 396–402. ACM, 2005.

David K George. *Stochastic Modeling and Decentralized Control Policies for Large-Scale Vehicle Sharing Systems via Closed Queueing Networks*. PhD thesis, The Ohio State University, 2012.

David K George, Cathy H Xia, and Mark S Squillante. Exact-order asymptotic analysis for closed queueing networks. *Journal of Applied Probability*, pages 503–520, 2012.

Supriyo Ghosh, Michael Trick, and Pradeep Varakantham. Robust repositioning to counter unpredictable demand in bike sharing systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3096–3102. AAAI Press, 2016.

Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research*, 58:387–430, 2017.

Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

William J Gordon and Gordon F Newell. Closed queuing systems with exponential servers. *Operations research*, 15(2):254–265, 1967.

Henry Grabar. How new yorks bike-share system pays riders to make it run better, 2017. URL `http://www.slate.com/blogs/moneybox/2017/02/09/new_york_s_citi_bike_pays_riders_to_make_it_run_better.html`.

Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In *Proceedings of the*

Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1522–1538. SIAM, 2012.

Bruce Hajek. Extremal splittings of point processes. *Mathematics of operations research*, 10(4):543–556, 1985.

Robert C Hampshire, William A Massey, and Qiong Wang. Dynamic pricing to control loss systems with quality of service targets. *Probability in the Engineering and Informational Sciences*, 23(02):357–383, 2009.

Jason D Hartline. Mechanism Design and Approximation. Manuscript, 2016. URL `http://jasonhartline.com/MDnA/`.

Andrew J. Hawkins. Uber limits drivers in nyc to 12 hour shifts. *The Verge*, 2016. URL `https://www.theverge.com/2016/2/12/10979730/uber-driver-fatigue-deactivation-nyc-twelve-hours`.

Sin C Ho and WY Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.

Yu-Ting Hsu, Lei Kang, and Yi-Hsuan Wu. User Behavior of Bikesharing Systems Under Demand–Supply Imbalance. *Transportation Research Record: Journal of the Transportation Research Board*, pages 117–124, 2016.

Winnie Hu. Yellow Cab, Long a Fixture of City Life, Is for Many a Thing of the Past. *New York Times*, 2017. URL `{https://www.nytimes.com/2017/10/12/nyregion/uber-taxis-new-york-city.html}`.

James R Jackson. Jobshop-like queueing systems. *Management science*, 10(1): 131–142, 1963.

Nanjing Jian and Shane G Henderson. An introduction to simulation optimization. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1780–1794. IEEE Press, 2015.

Nanjing Jian, Daniel Freund, Holly Wiberg, and Shane G. Henderson. Simulation optimization for a large-scale bike-sharing system. In T. M. K. Roeder, P. I. Frazier, R. Szechtman, and E. Zhou, editors, *Proceedings of the 2016 Winter Simulation Conference*, pages 602–613, Piscataway NJ, 2016. IEEE.

David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting Steiner tree problem: theory and practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–769. Society for Industrial and Applied Mathematics, 2000.

Ashish Kabra, Karan Girotra, and Elena Belavina. Bike-share systems: accessibility and availability. Working paper, 2015.

Mor Kaspi, Tal Raviv, and Michal Tzur. Detection of unusable bicycles in bike-sharing systems. *Omega*, 65:10 – 16, 2016. ISSN 0305-0483.

Mor Kaspi, Tal Raviv, and Michal Tzur. Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. *IISE Transactions*, 49(2):144–158, 2017. doi: 10.1080/0740817X.2016.1224960. URL `http://dx.doi.org/10.1080/0740817X.2016.1224960`.

Frank Kelly and Elena Yudovina. *Stochastic networks*, volume 2. Cambridge University Press, 2014.

Frank P Kelly. *Reversibility and Stochastic Networks*. Cambridge University Press, 2011.

Christian Kloimüllner, Petrina Papazek, Bin Hu, and Günther R Raidl. Balancing bicycle sharing systems: an approach for the dynamic case. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 73–84. Springer, 2014.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

Retsef Levi and Ana Radovanovic. Provably near-optimal lp-based policies for revenue management in systems with reusable resources. *Operations Research*, 58(2):503–507, 2010.

Asaf Levin. A better approximation algorithm for the budget prize collecting tree problem. *Operations Research Letters*, 32(4):316–319, 2004.

Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 33. ACM, 2015.

LimeBike. Limebike juicer, 2018. URL `https://web.limebike.com/juicer?TrucksFoT`.

Meghna Lowalekar, Pradeep Varakantham, Supriyo Ghosh, Sanjay Dominik Jena, and Patrick Jaillet. Online Repositioning in Bike Sharing Sys-

tems. Manuscript, 2017. URL http://web.mit.edu/~jaillet/www/general/icaps17-bikeshare.pdf.

Paul Milgrom and Ilya Segal. Deferred-acceptance auctions and radio spectrum reallocation. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 185–186. ACM, 2014.

Satoko Moriguchi and Kazuo Murota. On fundamental operations for multimodular functions. *arXiv preprint arXiv:1805.04245*, 2018.

Satoko Moriguchi, Kazuo Murota, Akihisa Tamura, and Fabio Tardella. Discrete midpoint convexity. *arXiv preprint arXiv:1708.04579*, 2017.

Kazuo Murota. *Discrete convex analysis*. SIAM, 2003.

Kazuo Murota. On steepest descent algorithms for discrete convex functions. *SIAM Journal on Optimization*, 14(3):699–707, 2004.

Kazuo Murota. Note on multimodularity and l-convexity. *Mathematics of Operations Research*, 30(3):658–661, 2005.

Viswanath Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.

Rahul Nair, Elise Miller-Hooks, Robert C Hampshire, and Ana Bušić. Large-scale vehicle sharing systems: analysis of vélib'. *International Journal of Sustainable Transportation*, 7(1):85–106, 2013.

NYCBS. June 2016 monthly report, 2016. URL https://d21xlh2maitm24.cloudfront.net/nyc/June-2016-Citi-Bike-Monthly-Report.pdf?mtime=20160803201017.

NYCBS. Citi bike system data, 2017a. URL https://www.citibikenyc.com/system-data.

NYCBS. Citi bike json feed, 2017b. URL https://gbfs.citibikenyc.com/gbfs/en/station_status.json.

NYCBS. September 2017 monthly report, 2017c. URL https://d21xlh2maitm24.cloudfront.net/nyc/September-2017-Citi-Bike-Monthly-Report.pdf?mtime=20171018154130.

NYCBS. Bike angel program, 2018. URL https://bikeangels.citibikenyc.com.

Eoin O'Mahony. *Smarter Tools For (Citi) Bike Sharing*. PhD thesis, Cornell University, 2015.

Eoin O'Mahony and David B Shmoys. Data analysis and optimization for (citi) bike sharing. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 687–694, 2015.

Eoin O'Mahony, Shane G. Henderson, and David B. Shmoys. (Citi)Bike sharing. working paper, 2016.

Erhun Ozkan and Amy R Ward. Dynamic matching for real-time ridesharing. Submitted, 2016.

Pulkit Parikh and Satish Ukkusuri. Estimation of optimal inventory levels at stations of a bicycle sharing system. In *Transportation Research Board Annual Meeting*, 2014.

Alice Paul, Daniel Freund, Aaron Ferber, David B. Shmoys, and David P. Williamson. Prize-Collecting TSP with a Budget Constraint. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-049-1. doi: 10.4230/LIPIcs.ESA.2017.62. URL `http://drops.dagstuhl.de/opus/volltexte/2017/7837`.

Günther R Raidl, Bin Hu, Marian Rainer-Harbach, and Petrina Papazek. Balancing bicycle sharing systems: Improving a VNS by efficiently determining optimal loading operations. In *International Workshop on Hybrid Metaheuristics*, pages 130–143. Springer, 2013.

Marian Rainer-Harbach, Petrina Papazek, Bin Hu, and Günther R Raidl. Balancing bicycle sharing systems: A variable neighborhood search approach. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 121–132. Springer, 2013.

Frank P Ramsey. A contribution to the theory of taxation. *The Economic Journal*, 37(145):47–61, 1927.

Tal Raviv and Ofer Kolka. Optimal inventory management of a bike-sharing station. *IIE Transactions*, 45(10):1077–1093, 2013.

Tal Raviv, Michal Tzur, and Iris A Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, 2013.

Regional Plan Association. Building transit-friendly communities a design and development strategy for the tri-state metropolitan region, 1997.

Gerhard Reinelt. TSPLIB – a traveling salesman problem library. *ORSA Journal on Computing*, pages 376–384, 1991.

Martin Reiser and Stephen S Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM (JACM)*, 27(2):313–322, 1980.

Carlos Riquelme, Ramesh Johari, and Baosen Zhang. Online Active Linear Regression via Thresholding. In *AAAI*, pages 2506–2512, 2017.

Christian Rudloff and Bettina Lackner. Modeling demand for Bikesharing Systems: Neighboring Stations as Source for Demand and Reason for Structural Breaks. *Transportation Research Record: Journal of the Transportation Research Board*, pages 1–11, 2014.

Syed Moshfeq Salaken, Mohammad Anwar Hosen, Abbas Khosravi, and Saeid Nahavandi. Forecasting Bike Sharing Demand Using Fuzzy Inference Mechanism. In *ICONIP 2015: Proceedings of the 22nd International Conference on Neural Information Processing*, pages 567–574. Springer, 2015.

Robert M Saltzman and Richard M Bradford. Simulating a More Efficient Bike Sharing System. *Journal of Supply Chain and Operations Management*, 14(2):36, 2016.

J. Schuijbroek, R.C. Hampshire, and W.-J. van Hoeve. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992 – 1004, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.08.029. URL `http://www.sciencedirect.com/science/article/pii/S0377221716306658`.

Richard Serfozo. Introduction to stochastic networks, 1999.

Julie Sherman. How divvy rebalancing problems can make commuting less safe for women. https://chi.streetsblog.org/2017/04/27/how-divvy-rebalancing-problems-can-make-commuting-less-safe-for-wom 2017.

Jia Shu, Mabel C Chou, Qizhang Liu, Chung-Piaw Teo, and I-Lin Wang. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research*, 61(6):1346–1359, 2013.

Divya Singhvi, Somya Singhvi, Peter I Frazier, Shane G Henderson, Eoin O'Mahony, David B Shmoys, and Dawn B Woodard. Predicting Bike Usage for New York City's Bike Sharing System. In *AAAI Workshop: Computational Sustainability*, 2015.

WY Szeto, Ying Liu, and Sin C Ho. Chemical reaction optimization for solving a static bike repositioning problem. *Transportation research part D: transport and environment*, 47:104–135, 2016.

Shalabh Vidyarthi and Kaushal K. Shukla. Approximation algorithms for P2P orienteering and stochastic vehicle routing problem. *arXiv preprint arXiv:1501.06515*, 2015.

Patrick Vogel, Bruno A Neumann Saavedra, and Dirk C Mattfeld. A hybrid metaheuristic to solve the resource allocation problem in bike sharing systems. In *International Workshop on Hybrid Metaheuristics*, pages 16–29. Springer, 2014.

Jay Walder. Rolling along: Bicycles, mobility, and the future of cities. http://www.mckinsey.com/business-functions/sustainability-and-resource-productivity/our-insights/

rolling-along-bicycles-mobility-and-the-future-of-cities, 2016.

Ariel Waserhole and Vincent Jost. Pricing in vehicle sharing systems: optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, pages 1–28, 2014.

Ward Whitt. Open and closed models for networks of queues. *AT&T Bell Laboratories Technical Journal*, 63(9):1911–1979, 1984.

Peter Whittle. Scheduling and characterization problems for stochastic networks. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 407–428, 1985.

Jiawei Zhang, Xiao Pan, Moyin Li, and Philip S Yu. Bicycle-sharing system analysis and trip prediction. *arXiv preprint arXiv:1604.00664*, 2016.

Rick Zhang and Marco Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3):186–203, 2016.