©Copyright 2026
Yifan Shen

Pricing and Matching Shared Rides

Yifan Shen

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2026

Reading Committee:

Chiwei Yan, Co-chair

Chaoyue Zhao, Co-chair

Julia Yan

Michael Wagner

Program Authorized to Offer Degree:
Industrial and Systems Engineering

University of Washington

Abstract

Pricing and Matching Shared Rides

Yifan Shen

Co-Chairs of the Supervisory Committee:
Chiwei Yan
Chaoyue Zhao
Industrial and Systems Engineering

Shared rides, which pool individual riders into a single vehicle, are essential for mitigating congestion and promoting more sustainable urban transportation. However, major ridesharing platforms have long struggled to maintain a healthy and profitable shared rides product. The key challenge lies in designing pricing and matching policies that effectively balance platform profitability, rider accessibility, and system efficiency. This research aims to address this challenge through three interconnected research works.

The first work tries to address the problem from the perspective of pricing. To understand why shared rides have struggled, we analyze procedures commonly used in practice to set *static* prices for shared rides, and discuss their pitfalls. We then propose a pricing policy that is *adaptive* to matching outcomes, dubbed *match-based* pricing, which varies prices depending on whether a rider is dispatched alone or to what extent she is matched with another rider. Analysis on a single origin-destination setting reveals that match-based pricing is both profit-maximizing *and* altruistic, simultaneously improving cost efficiency (i.e., the fraction of cost saved by shared rides relative to individual rides) and reducing rider payments relative to the optimal static pricing policy. These theoretical results are validated on a large-scale simulation with hundreds of origin-destinations from Chicago ridesharing data. The improvements in efficiency and reductions in payments are especially noticeable when costs are high and demand density is low, enabling healthy operations where they have historically been most challenging.

Whereas the first part assumes that riders are matched to each other *pre-trip* (i.e., before the rider is dispatched), this neglects a ride's continued matching potential while *on-trip* after she is initially dispatched solo. The second work develops a dynamic, stochastic matching model where the platform makes both pre-trip and on-trip matching decisions, and contrast the behavior of each phase of matching. Using both synthetic and real-world data from Chicago, we find that whereas pre-trip matching is well-suited to dense downtown areas with concentrated demand, on-trip matching is critical in sparser outskirts where demand is spatially dispersed, and manages a trade-off between matching opportunity and value. We also embed the matching model in an outer pricing optimization problem to study the interaction of matching with pricing, and find that the addition of on-trip matching increases profitability and efficiency for the platform and lowers prices for riders. These effects are particularly pronounced in the sparse outskirts, where shared rides have historically been most difficult to sustain.

The third work complements the analytical and computational contributions with a simulation-based teaching game, *Calyber*, which is designed to provide hands-on and engaging experience in developing real-time pricing and matching decisions for shared ride services. Students design and implement dynamic pricing and matching policies using a rich historical ridesharing dataset, competing for top performance on a holdout test set. Through this case, students gain practical insight into stochastic dynamic decision-making within a modern, relevant, and data-driven context. Results from previous class implementations provide strong evidence of enhanced learning and engagement.

Together, the three works offer a comprehensive treatment of shared ride operations in terms of joint decision making in pricing and matching, spanning modeling analysis, computational methodology, and pedagogical practice.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables vi

Chapter 1: Introduction 1
1.1 Background 1
1.2 Motivation 3
1.3 Research Problems and Dissertation Overview 5

Chapter 2: Pricing Shared Rides 8
2.1 Introduction 8
2.2 Literature Review 10
2.3 Models and Preliminaries 12
2.4 Computational Experiments 33
2.5 Conclusion and Discussion 38

Chapter 3: On-Trip Matching and Pricing for Shared Rides 43
3.1 Introduction 43
3.2 Literature Review 46
3.3 Modeling 48
3.4 Computational Experiments 57
3.5 Conclusion and Discussion 70

Chapter 4: Calyber: A Ridesharing Game 71
4.1 Introduction 71
4.2 Related Work 73
4.3 Learning Objectives 75
4.4 Audience, Prerequisites and Position in Course 75
4.5 Teaching Plan 76
4.6 Teaching Experience 77

4.7 Conclusion 82

Chapter 5: Conclusion 85

Appendix A: Supplementary Materials for Chapter 2 101
A.1 Solo, Shared Rides and Sharing Disutility 101
A.2 Computing the Static Pricing Policy 109
A.3 Proofs 111
A.4 Robustness Tests 130
A.5 Additional Results 137

Appendix B: Supplementary Materials for Chapter 3 140
B.1 Modeling Combined Pre-Trip and On-Trip Matching 140
B.2 Proofs 142
B.3 Other Results Based on Chicago Ridesharing Data 148

Appendix C: Supplementary Materials for Chapter 4 151
C.1 Case 151

LIST OF FIGURES

Figure Number — Page

1.1 Timeline of pricing policies tested by Uber and Lyft in the first year. 2
1.2 Ride-hailing trip distributions and match rates in Chicago (2018-2024). . . . 3

2.1 A Markov chain for a shared rides system with a single origin-destination pair. 14
2.2 Pricing in the single origin-destination pair case. 16
2.3 Illustration of Theorem 1 and Theorem 2. 19
2.4 Two Uber products that resemble match-based pricing. 20
2.5 Comparisons between optimal match-based prices and static prices. 23
2.6 Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing). 24
2.7 An example when two requests of the same type i might be held to enable future matches (i, m) and (i, j). 28
2.8 Geographical distribution: quoted prices and disutilities in Chicago 39
2.9 Geographical distribution of payments and realized disutilities in Chicago . . 40
2.10 Geographical distribution of cost efficiency, match rate, and detour rate in Chicago. 41
2.11 Arrival rates and relative differences in quoted price, quoted disutility, payment, realized disutility and detour rate (match-based versus static pricing) in Chicago. 42

3.1 Screenshots of Uber app with both pre-trip and on-trip matching. 43
3.2 Illustrations of incompatibility between a solo rider (j, u) and a new rider of type i. 50
3.3 Illustration of the two-rider-type example, and profit comparison between pre-trip and on-trip matching under varying demand heterogeneity. 59
3.4 Examples of the distributions of the random rider types on the grid network $(N = 1, 5, 10;\ L = 100)$. 60
3.5 Profit comparison between on-trip and pre-trip matching policies under different numbers of rider types on the grid network. 60
3.6 Geographical distribution of arrival rates, trip lengths, and household incomes in Chicago neighborhoods. 62

3.7 Profit comparison between combined and pre-trip matching policies under different waiting periods ($c = 0.7$). 66
3.8 Region split in Chicago and matching outcome distribution of combined matching when $c = 0.9$, waiting period is 1.5 min. 67
3.9 Geographical distribution of performance metrics in Chicago. 68

4.1 Student team profits, compared to Nick's and Thea's engines. 79
4.2 Profit vs. average quoted prices among student teams. 80
4.3 Profit vs. cost per rider among student teams. 81

A.1 Comparisons between optimal match-based conversions and static conversions with $\beta = 0.1$. 104
A.2 Profitable regions and payment/cost comparison of pricing policies with sharing disutility 105
A.3 Risks under different pricing policies for $c = 0.8$ and varying θ. 108
A.4 Risks under different pricing policies for $c = 0.8$, sharing disutility $\beta = 0.1$ and varying θ. 108
A.5 Illustration of the proof of Claim 2. 125
A.6 Static prices under the willingness-to-pay distribution $\mathsf{Beta}(6, 2)$. 131
A.7 Convergence of iterative pricing under different $\mathsf{Beta}(a, b)$ willingness-to-pay distributions. 131
A.8 Comparison between match-based and static prices under different $\mathsf{Beta}(a, b)$ willingness-to-pay distributions. 132
A.9 Comparisons between optimal match-based prices and static prices under the risk-averse model. 134
A.10 Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing). All results are under the risk-averse model. 134
A.11 Comparisons between optimal match-based prices and static prices under the batch window model. 136
A.12 Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing). All results are under the batch window model. 137
A.13 Cost allocation example. 137

B.1 Downtown/outskirts region split and the cost efficiency distribution in downtown versus outskirts across the three policies ($c = 0.7$, waiting period is 1 min). 149
B.2 Matching outcome distribution over downtown/outskirts requests ($c = 0.7$, waiting window = 1 min). 150

C.1 Downtown and outer neighborhoods of Chicago. 156
C.2 Illustration of the components of routes serving two riders. 158
C.3 Demand (riders per minute), cost per rider ($/mile, under Nick's engine), and quoted price ($/mile, under Nick's engine) across different neighborhoods of Chicago, for a typical weekday 7:30-8:30am. 159

LIST OF TABLES

Table Number Page

2.1 Convergence results of Algorithm 1. 18

2.2 Test results on Chicago data ($\beta_0 = 0.05, \beta_1 = 0.675$), averaged over 100 simulation runs using test data. 37

3.1 Test results on Chicago data. 64

4.1 Summary of previous class implementations. 77

4.2 Median KPIs across student teams, compared to those for Nick's and Thea's engines. 78

4.3 Summary of student feedback across cohorts. 83

A.1 Derivatives of $q(\theta)$ on $(0, \infty)$ for $c \in (0, 1)$. 118

A.2 Training results on Chicago data ($\beta_0 = 0.05, \beta_1 = 0.675$). 139

B.1 Training results on Chicago data. 148

C.1 KPIs for Nick's and Thea's pricing and matching engines, by neighborhood type. 157

C.2 Sample of Calyber's data on routes taken to fulfill rider requests. 157

C.3 Sample of Calyber's data on rider sessions. 157

ACKNOWLEDGMENTS

I want to first thank my advisors, Chiwei Yan and Julia Yan. I am grateful for their patience, their generosity, and the many hours they spent on my work. Working with them for four years has been one of the best parts of this journey. Beyond research, they also supported me through my job search, offered guidance on my career that I will carry with me for a long time, and made my visit to UBC possible—the six months I spent in Vancouver are among the most special of my life.

I also want to thank the members of my dissertation committee, Chaoyue Zhao and Michael Wagner, for their time and feedback. I am especially grateful to Chaoyue for her willingness to serve as the committee co-chair.

The three chapters in this dissertation grew out of close collaborations. Chiwei Yan and Julia Yan were partners on all three projects. I am also grateful to my co-author Junlin Chen, who was consistently helpful and a genuine pleasure to work with.

To all the instructors whose classes I took during my PhD—Kamran Moinzadeh, Maryam Fazel, Thomas Rothvoss, James A. Ritcey, Thomas Richardson, Jacques Lawarree, Xu Tan, Chiwei Yan, Shuai Huang, John Choe, Ashis Banerjee, and others—thank you for the teaching.

To the staff of UW ISE—Yuliya, Jennifer, Kelly, Neelu, and so on—thank you for keeping things running and for always being willing to help. To the faculty I worked with as a TA—Patty Buchanan, John Choe, Christina Mastrangelo, Chiwoo Park, and Maggie Ramirez—thank you for the experience and the support.

I also want to thank my previous advisors, Jiangang Jin and Jun Xia. I would not have started down this path without them.

To my fellow PhD and master's students and friends—Xinyue Chen, Ang Xu, Oscar Xu, Ziyan He, Nuoyan Zeng, Qiong Wu, Chenhan Wu, Ryan Warnock (and his mother Glori),

Yanwei Sun, Shiding Huang, and so on—thank you for the company and the support, especially during the stretches when things felt heavy. I am glad I did not have to go through this alone.

To my family: your support has never wavered, and I have never taken that for granted.

And finally, to myself, for keeping going.

Chapter 1
INTRODUCTION

1.1 Background

Urban transportation systems face a long-standing tension. Ride hailing service provides the on-demand, door-to-door mobility to riders, but at the cost of congestion, emissions, and inefficient vehicle utilization. Shared rides—in which a single vehicle transports multiple riders with compatible origins and destinations—have long been proposed as a way to address this tension [1, 44]. If riders heading in the same direction with similar arrival times are pooled together, the same trips can be served with fewer vehicles, fewer vehicle miles, and lower per-rider cost. In principle, every party benefits: platforms serve more riders at lower cost; riders enjoy lower fares driven by lower costs; and cities see less congestion, fewer emissions, and better use of road space.

The launch of UberPool and Lyft Line in 2014 brought this vision into widespread practice. Within a few years, shared rides accounted for 20-30% of all trips on the two platforms [70, 72]. In 2018, Uber expanded its commitment by launching Uber Express Pool, a lower-cost variant in which riders walked to dynamic meeting points [108], and Lyft publicly committed to making 50% of its rides shared by the end of 2020 [72]. Lyft co-founder John Zimmer went so far as to claim that shared rides could bring about "the end of traffic" [61]. For a brief period, it appeared that the two largest ridesharing platforms were on track to make shared rides a mainstream mode of urban transportation.

These expectations were not met. After the launch in 2014, both companies sought to "win" riders in San Francisco, chiefly by keeping prices low—Uber even offered a $5 flat fare.[1] But at the same time, Uber's algorithms struggled to match riders: the *match rate*, or the fraction of riders who get matched with another rider, was reportedly only 8%

[1]See Figure 1.1 for a timeline of the pricing policies that Uber and Lyft experimented with over the first year launching shared rides.

initially at launch. The combined effect was that Uber was essentially offering ultra-cheap solo rides, losing roughly one million dollars per week on UberPool while building density through aggressive subsidies [7]. Over the next few months, Uber improved its matching algorithms and achieved a match rate of almost 25%.[2] But this improvement was still not enough to break even. Attempting to cut its losses, Uber raised prices, which then tanked ridership by 22% [7].

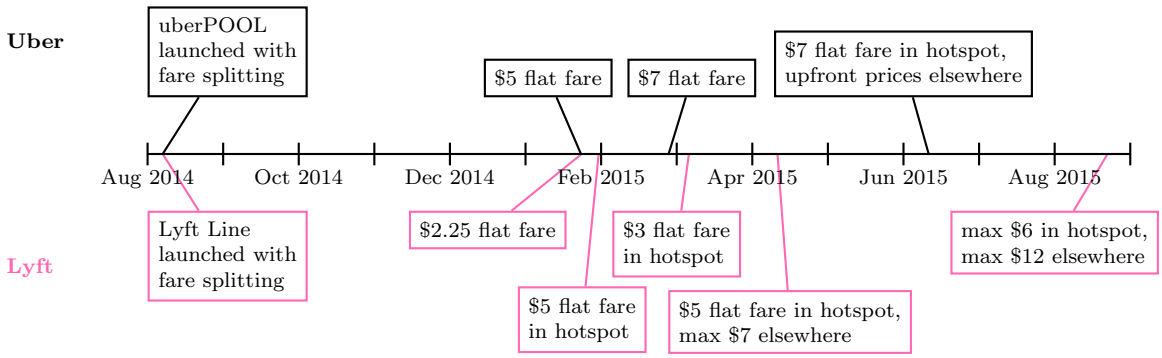
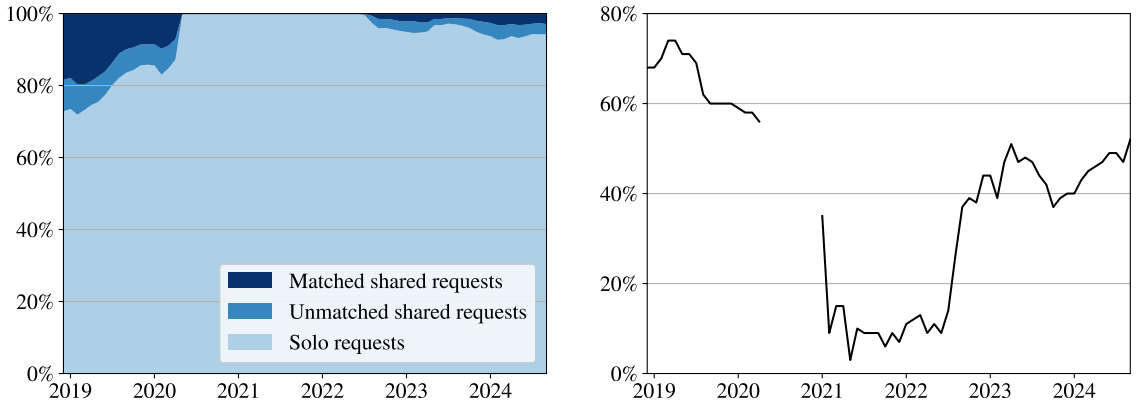


Figure 1.1: Timeline of pricing policies tested by Uber and Lyft in the first year. (Figure reproduced from [7].)

The COVID-19 pandemic compounded these difficulties. Both Uber and Lyft suspended shared rides in 2020. Lyft relaunched in 2021 [73] and Uber followed in 2022 with a re-branded UberX Share [110], but these relaunches struggled to find traction [33]. In 2023, Lyft discontinued shared rides entirely in the face of financial difficulties [29], only to reintroduce them in 2025. Figure 1.2 illustrates the resulting pattern in Chicago: even by 2024, after years of operational refinement, only a small share of ride-hailing trips are matched, and the match rate among requested shared rides recovered only partially from its post-pandemic collapse.

Another challenge reflected is that the success of shared rides relies heavily on local demand density, which directly affects the availability of quality matches [3]. Platforms acknowledge that "matching with a co-rider is dependent on ... the number of ride requests ... available in a given area" [113], and limit their shared ride offerings accordingly. UberX

[2]See Figure 1.2 for the match rates of Uber and Lyft's shared rides in Chicago over recent years.



(a) Ride-hailing trip distributions by class

(b) Match rate of shared requests

Figure 1.2: Ride-hailing trip distributions by class and match rate of shared requests in Chicago (November 2018 - August 2024). Plots are reproduced based on data in `https://toddwschneider.com/dashboards/chicago-taxi-ridehailing-data/`. From April 2020 to April 2021, shared rides were suspended across major ride-hailing platforms due to COVID-19. Lyft relaunched shared rides in May 2021; Uber launched UberX Share in June 2022.

Share is only available in select major cities with high demand [112], and Lyft, after discontinuing shared rides entirely in 2023, pivoted in 2025 to offering shared rides only at major airports where demand is concentrated [71]. Consequently, the benefits of shared rides have not been realized outside of dense urban centers; notably, lower-density, outlying areas also tend to coincide with transit deserts [60, 5]. An analysis of Toronto ridesharing data finds that downtown Toronto accounts for 66% of total matches with a match rate of 62%, whereas the match rate in the outlying areas drops to only 39% [120]. In effect, shared rides have succeeded mainly where they were least needed and failed where they were most needed.

1.2 Motivation

The history above highlights a key challenge in the design of the platform's operational policies—how pricing and matching decisions are made and how they interact. Pricing and matching are two critical levers of the platforms. They each involve a fundamental tension, and their interaction makes the joint problem harder.

The pricing tension. Low prices are needed to attract riders and to build the demand density that enables efficient matching. At the same time, low prices mean the platform may miss out on revenue from riders with higher willingness to pay, and can create losses when the platform fails to match riders and instead dispatches them as discounted solo rides. In trying to strike a balance, platforms can end up setting prices that are both too high to attract riders and too low to operate sustainably. Furthermore, the difficulty is compounded when a price is offered before the matching outcome is known, as the platform bears the risk that no compatible rider arrives, and pays the extra cost of a solo ride at a discounted price.

The matching tension. Matching shared rides involves a tension between matching immediately and waiting. Committing to matching now realizes the cost saving of the current pair, but gives up the possibility of a better match later; waiting for a future match preserves the potential of a better match, but risks outcomes that can each be more costly than the current match, depending on the platforms's mechanism: the rider may abandon or be dispatched solo when her waiting window expires, or she may eventually be matched with a less compatible co-rider. This can be more complicated if on-trip matching is allowed (i.e., matching can happen on the route after a ride is dispatched solo; see Chapter 3 for more details) since the matching value and matching opportunity of an on-trip rider may vary over time.

The coupling of pricing and matching. Pricing and matching are two interconnected levers. Prices determine who requests a shared ride, and therefore shape the demand density for the platform to match. Matching, in turn, determines the per-rider cost depending on its matching efficiency that affects the optimal price to quote. Designing pricing and matching together, rather than in sequence, is therefore a core idea of this dissertation.

These observations motivate the central question of this dissertation: How should ridesharing platforms design pricing and matching policies that make shared rides sustainable, particularly in the low-density settings where they have historically failed? The remainder of this chapter breaks it down into three more concrete research questions, and previews the

contributions of the three subsequent chapters.

1.3 Research Problems and Dissertation Overview

The three works that make up this dissertation share a common motivation to make shared rides operationally sustainable, while addressing three individual problems:

Problem 1: Why have platforms struggled in pricing, and how to design the pricing policy for shared rides? To understand why shared rides have struggled, Chapter 2 analyzes procedures commonly used in practice to set *static* prices for shared rides, and discusses their pitfalls. We then propose a pricing policy that is *adaptive* to matching outcomes, dubbed *match-based* pricing, which varies prices depending on whether a rider is dispatched alone or to what extent she is matched with another rider. Analysis on a single origin-destination setting reveals that match-based pricing is both profit-maximizing *and* altruistic, simultaneously improving cost efficiency (i.e., the fraction of cost saved by shared rides relative to individual rides) and reducing rider payments relative to the optimal static pricing policy. These theoretical results are validated on a large-scale simulation with hundreds of origin-destinations from Chicago ridesharing data. The improvements in efficiency and reductions in payments are especially noticeable when costs are high and demand density is low, enabling healthy operations where they have historically been most challenging. This work is published in *Operations Research* [118].

Problem 2: How to make both pre-trip and on-trip matching decisions, and when on-trip matching is most valuable? Although shared rides have the potential to increase vehicle utilization and reduce congestion and emissions, these benefits depend heavily on ridesharing platforms' ability to match riders effectively. As such, shared rides have seen limited success outside of dense urban areas—the sparse outskirts of greater metropolitan areas remain underserved. In the literature, the dominant matching model involves collecting rider requests in a batch interval and solving a non-bipartite matching problem on the requests. However, this model neglects the ability of a rider to be matched to a future arriving rider even after she is initially dispatched solo; namely, match-

ing is only modeled *pre-trip*, and the value of *on-trip* matching is not explicitly accounted for. Chapter 3 develops a dynamic, stochastic matching model, where the platform makes both pre-trip and on-trip matching decisions, and contrasts the behavior of each phase of matching. Using both synthetic and real-world data from Chicago, we find that whereas pre-trip matching is well-suited to dense downtown areas with concentrated demand, on-trip matching is critical in sparser outskirts where demand is spatially dispersed, and manages a trade-off between matching opportunity and value. We also embed the matching model in an outer pricing optimization problem to study the interaction of matching with pricing, and find that the addition of on-trip matching increases profitability and efficiency for the platform and lowers prices for riders. These effects are particularly pronounced in the sparse outskirts, where operating shared rides—even providing access to any form of transportation—has historically been most challenging.

Problem 3: How to turn the joint pricing-and-matching problem into a teaching case to enhance the education of dynamic, stochastic decision making? Chapter 4 completes the dissertation by turning the joint pricing-and-matching problem into a teaching setting. We introduce *Calyber*, a simulation-based game designed to provide a hands-on and engaging experience in developing real-time pricing and matching decisions for shared ride services. Students design and implement dynamic pricing and matching policies using a rich historical ridesharing dataset, competing for top performance on a holdout test set. Through this case, students gain practical insight into stochastic dynamic decision-making within a modern, relevant, and data-driven context. Results from previous class implementations provide strong evidence of enhanced learning and engagement. This work is published in *INFORMS Transactions on Education* [95, 96].

The remainder of this dissertation proceeds as follows. Chapter 2 studies the pricing problem for shared rides, characterizes the structure of optimal pricing policies, and develops scalable computational approaches for realistic networks. Chapter 3 extends this framework to the on-trip matching and pricing problem, where the platform must jointly decide how to price and match riders in the on-trip regime. Chapter 4 completes the dis-

sertation with Calyber, a simulation-based ridesharing game that builds the pricing and matching problem into a hands-on educational tool, providing evidence that these ideas can be taught effectively to diverse audiences. Chapter 5 concludes the dissertation. Together, these works advance our understanding of shared ride operations in terms of joint pricing and matching decisions, regarding modeling analysis, computational methodology, and pedagogical practice.

Chapter 2

PRICING SHARED RIDES

2.1 Introduction

From their inception, *shared rides* have been envisioned as a means of transporting compatible riders together in a carpool-type service, thereby reducing vehicle congestion and carbon emissions [1, 44]. However, although platforms such as Uber and Lyft have offered shared rides since 2014, these products have long struggled with profitability and service quality [33].

The up-and-down history of shared rides highlights the following challenge. Low prices are needed to attract riders and achieve enough density for efficient pooling (as with Uber Express Pool). At the same time, low prices mean the platform may miss out on revenue from high-valuation riders, which can create losses for the platform if riders cannot be pooled efficiently [34]. In trying to strike a balance, platforms may set prices that are both too high to attract riders *and* too low to comfortably sustain.

To investigate this tension, we consider a dynamic stochastic pricing and matching model, where riders randomly arrive and request a ride based on prices quoted by the platform. Riders who make requests spend limited time on the platform waiting for a match with other riders, and these matches are made dynamically by the platform to maximize the long-run average profit. By developing and analyzing solutions for this model, we make the following contributions:

1. **Why have platforms struggled?** Historically, platforms have offered a single upfront price for shared rides [25, 74]. We show that even on a single origin-destination pair with uniform rider willingness-to-pay distributions, computing the optimal price can be challenging (Proposition 1). Moreover, the optimal price exhibits a discontinuity as a function of the demand density (Remark 1) — the price suddenly jumps as the demand becomes sparser. These characteristics arise from the unique feature

where the cost of a shared ride is influenced by its price. We then analyze a heuristic that is widely used across platforms — an iterative approach to pricing. In this pricing approach, the platform starts with some initial prices and observes the resulting matching outcomes and costs for a period of time. The platform then updates the prices based on the observed costs. In Theorems 1 and 2, we find that this iterative approach always converges to a price that is *higher* than optimal, and that in some settings, the platform can be gradually priced out of the market even when it would be profitable to offer shared rides. These observations point out a critical drawback of offering a single price for a shared ride — there is a risk of *not* being able to match riders after quoting the price. A single price also limits the platform's flexibility in reducing prices to improve cost efficiency.

2. **A way forward.** We discuss an alternative pricing policy, *match-based pricing*, which charges a higher price for riders who get dispatched solo, and offers a discount to riders who are dispatched with a matched rider. By doing so, it incentivizes riders to request when they can be matched well. This policy shares some similarity with the pricing structure that Uber introduced in 2022, which offers "additional savings if [a rider is] matched with a co-rider" [110, 46] (see Section 2.3.1 for detailed comparison). We characterize the optimal solution for a single origin-destination pair and find that although multiple price values are needed in match-based pricing, they are more tractable to compute than the optimal static price (Theorem 3). In addition to improving tractability and profitability, we show in Proposition 2 that the prices quoted to riders under our policy can be lower than the optimal static price when costs are high and demand density is low – a regime where platforms struggle. Moreover, riders *consistently* pay less due to higher cost efficiency (Proposition 3).

3. **Performance on real-world data.** We extend our models and analyses to multiple origin-destination pairs and develop simple and scalable computational approaches. We demonstrate our proposed policy in a large-scale simulation based on real ridesharing data in Chicago. The simulations corroborate our theoretical results and show

substantial gains in efficiency. In particular, our policy enables healthy operations where they have historically been most challenging.

The remainder of this chapter is outlined as follows. In Section 2.2, we review related literature. We describe our models, present our analyses and develop scalable computational approaches in Section 2.3, starting with a single origin-destination pair before generalizing to multiple origin-destination pairs. We report computational results in Section 2.4 and offer concluding remarks in Section 2.5. Code and data to reproduce all computational experiments in the chapter can be accessed at `https://github.com/ORJournal/2023.0513`.

2.2 Literature Review

Ride-hailing, in which riders are served by solo rides, and *ridesharing*, in which riders are pooled together in shared vehicles, are both well-studied in the literature. In ride-hailing, a major focus of the literature has been on the problem of balancing the supply of drivers with the demand for rides. To this end, [69, 17], and [16] study spatial pricing as a means to incentivize drivers to relocate to areas of high demand or to join the platform during periods of high demand. [47] and [75] further study how spatio-temporal pricing structure can incentivize drivers to accept requests for trips. A hallmark of dynamic pricing in ride-hailing is to limit poor rider-driver matches, mitigating the so-called *wild-goose chase* in which drivers suffer poor utilization from the long pickup times associated with poor matches [22, 119]. [21] additionally argue that dynamic pricing offers riders lower prices during periods of low demand and improves riders' access to service during high demand. Taking a different perspective, [12] find that dynamic pricing enables performance that is robust to uncertainty in estimated system parameters.

Apart from pricing, another critical component of ride-hailing is *matching* riders to drivers, where a key trade-off for all participants arises between committing to a match immediately or waiting for a (potentially) better match. [87] discuss the downside of a simple greedy policy that simply dispatches the closest driver to a rider, and propose asymptotically-optimal matching policies. [54] discuss the structure of the optimal matching policy when there are heterogeneous types. [115] consider a setting when drivers and

riders can reject undesirable matches, and develop formulations that ensure stable matching outcomes. [40] study pricing and matching in coordination, and develop a policy that considers potential riders who have not yet converted to requests.

As compared to individual rides, shared rides have the potential to reduce vehicle miles traveled, congestion, and emissions. However, attaining these benefits poses significant operational challenges. One such challenge is in deciding which riders should share service; to this end, [92] propose a two-rider matching problem, and [4] consider pooling riders in high-occupancy vehicles.

However, an upstream difficulty is that of having enough riders to enable efficient sharing in the first place. Common approaches to coalescing riders include solving the matching or pooling problem after *batching* requests over short fixed time windows [10]; asking riders to walk to shared meeting points [99, 119, 41, 123]; or allowing longer detours to enable more pickups or dropoffs [99]. However, these approaches also have their drawbacks. For batching, [88] find that a fixed batch window can split up compatible requests, while [9] argue that a batch window incurs the risk of riders abandoning the platform before dispatch. As for allowing detours, [27, 68], and [101] argue that long detours can deter time-sensitive riders from taking shared rides. Our work takes the alternate approach of using pricing to shape the demand for shared rides.

Whereas pricing for individual rides mainly focuses on shaping demand and supply, new challenges arise in shared rides. One such issue, *cost-sharing*, addresses how the cost of a ride should be split between multiple passengers arriving online with varying attributes (chiefly, origin and destination). [43] and [85] analyze a mechanism for a ridesharing platform that splits costs according to riders' scaled marginal costs, and show that the mechanism has the desirable properties of online fairness, immediate response, individual rationality, budget balance, and incentive compatibility. [55] then consider these properties for a peer-to-peer carpooling setting where drivers seek to partially recoup driving cost without unduly burdening riders. [42] also consider a carpooling setting, and split costs so that riders are incentivized to match at lowest total cost. In addition, [86] and [6] allocate costs while setting tolls to incentivize efficient carpooling.

Another line of work focuses on interactions between several transportation options.

[53] propose a randomized monetary scheme that nudges riders toward selecting shared over individual rides. [63, 122] and [116] model platforms that set prices for individual and shared rides, analyze market equilibria and discuss when it is preferable to offer one or both. [15] studies cost structures that promote platform-based ridesharing versus peer-to-peer carpooling.

Our work differs from these by analyzing status-quo pricing policies used by platforms and comparing them with a pricing policy where prices depend on matching outcomes to encourage matches. A similar feature also appears in [43] and [55]. However, the focus is different — they study incentive properties of various cost-sharing mechanisms where cost allocation continues to decrease as more riders join the ride. Another study close to ours is [62], where they compare *ex-ante* and *ex-post* pricing policies (we call these *static* and *match-based* policies, respectively). They discuss the incentive properties of these mechanisms under certain pricing values with complete information. They conclude that the ex-post mechanism makes it individually rational for riders to always opt for shared rides, leading to desirable outcomes. In contrast, we consider a stochastic setting where the uncertainty of matching outcomes is explicitly modeled. Moreover, we explicitly analyze and discuss the properties of optimal prices, and evaluate the benefits to riders and the platform.

2.3 Models and Preliminaries

In this section, we formally introduce our model preliminaries, analyses, and computational approaches. We start with Section 2.3.1 where all riders are of the same origin-destination pair. Though the setting is stylized, our analysis leads to complete characterizations of various pricing policies and their comparisons, which motivates our proposed policy. In Section 2.3.2, we then extend the model and develop scalable computational approaches for a general shared rides network with multiple origin-destination pairs.

2.3.1 A Single Origin-Destination Pair

We first consider a case where riders arrive at the platform according to a homogeneous Poisson process and consider requesting trips from the same origin to the same destination.

Without loss of generality, we assume that the rider *arrival rate* is one. Each rider's willingness to pay is drawn IID from a distribution function $F(\cdot)$ and the platform can control the rate of ride requests by quoting price p. Riders make *requests* if their willingness to pay is at least as high as the price p. As such, the *request conversion rate* is given by $\lambda = \bar{F}(p)$. We consider a specific case in this subsection where the riders' willingness to pay is uniformly distributed in $[0, 1]$ so that $\lambda = 1 - p$. Results under other willingness-to-pay distributions share similar insights and can be found in Appendix A.4.1. As a convention, we express all that follows in terms of the request conversion rate λ rather than the price p, which can be thought of as the platform setting a target request conversion rate through the corresponding price.

It costs the platform c to pay a driver to drive from the origin to the destination. We assume a capacity of two riders for each vehicle, so it is reasonable to restrict $c \in (0, 2)$ since otherwise, the platform cannot be profitable. When a rider makes a request, if there is no other request already waiting, then the rider waits on the platform for a sojourn time that follows an exponential distribution with rate $\theta > 0$. The limited sojourn time can be interpreted as rider patience, *or* as the platform's patience to wait for another request to arrive, as it would enable the platform to collect greater revenue for the same cost.[1] At the expiration of the sojourn time, the request is dispatched with a driver at a cost of c. But if there is another request waiting, then the two requests are immediately dispatched together with an available driver, now sharing the cost c. When two requests are dispatched together, we call them *matched*.[2] The number of waiting requests in the platform can be modeled as a continuous-time Markov chain with two states shown in Figure 2.1. The system goes from state 0 to 1 if there is a new request, which happens at rate λ. The system goes from state 1 to 0 if the current request's sojourn time elapses (at rate θ), or if a new request is

[1] One example of such practice is the Uber Express Pool launched in 2018 [108] where riders were asked to wait a few minutes (ranging from 0 to 5 minutes) before their trips began to help Uber make better matches with fewer detours, delivering a more affordable shared rides product. The precise waiting time balances the quality of current matches against potential future opportunities. In Appendix A.4.3, we also discuss an alternative where matching is executed according to a fixed batch window. Whether a rider is dispatched solo or matched depends on how many riders arrive in each batch window.

[2] The word *matched* is commonly used to describe the matching of drivers to requests. Throughout this chapter, we will reserve its usage for the matching of requests to other requests.

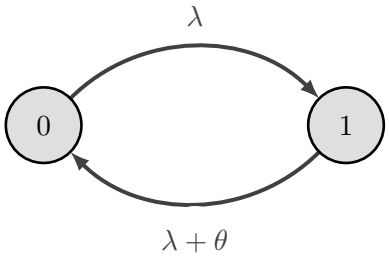


Figure 2.1: A Markov chain for a shared rides system with a single origin-destination pair.

matched and dispatched with the waiting request (at rate λ).

The system can be analyzed as follows. First, the steady-state probabilities are:

$$\pi_0 = \frac{\lambda + \theta}{2\lambda + \theta}, \quad \pi_1 = \frac{\lambda}{2\lambda + \theta}.$$

The *match rate* is the probability that a given request is matched. This is the probability that when a given request arrives, there is one request already waiting; or that there are no requests already waiting, but another will arrive before the given request's sojourn time elapses. Concretely, the match rate, denoted by $\rho(\lambda)$, is:

$$\rho(\lambda) = \pi_1 + \pi_0 \frac{\lambda}{\lambda + \theta} = \frac{2\lambda}{2\lambda + \theta}.$$

The *expected cost* per request is denoted by $\bar{c}(\lambda)$, and is given by:

$$\bar{c}(\lambda) = \frac{c}{2}\rho(\lambda) + c(1 - \rho(\lambda)) = \frac{c}{2}\left(1 + \frac{\theta}{2\lambda + \theta}\right). \tag{2.1}$$

The *cost efficiency* is then computed as $1 - \bar{c}(\lambda)/c$, and one can effectively think of this as the fraction of vehicle miles saved by operating shared rides instead of solo rides.[3]

Denoting the *expected profit* per unit time by $\Pi(\lambda)$, the platform seeks to solve the

[3]For multiple origin-destination pairs, the cost efficiency is computed as one minus total dispatching cost divided by a hypothetical cost if all requests were dispatched solo.

following price optimization problem:

$$\max_{\lambda \in [0,1]} \quad \Pi(\lambda) = (1 - \lambda - \bar{c}(\lambda))\lambda. \tag{2.2}$$

We will use λ^* to denote an optimal request conversion rate for (2.2). The first-order condition of (2.2) requires solving a cubic equation below.

$$\Pi'(\lambda) = -2\lambda - \frac{c\theta^2}{2(2\lambda + \theta)^2} + \left(1 - \frac{c}{2}\right) = 0.$$

We denote its largest real root by $\tilde{\lambda} := \max\{\lambda \in \mathbb{R} : \Pi'(\lambda) = 0\}$ which always exists. We have the following complete characterization of λ^*.

Proposition 1 (Optimal request conversion rate λ^*)**.** *When $c \in (0, 1]$, the unique optimal request conversion rate $\lambda^* = \tilde{\lambda}$. When $c \in (1, 2)$, any optimal request conversion rate satisfies:*

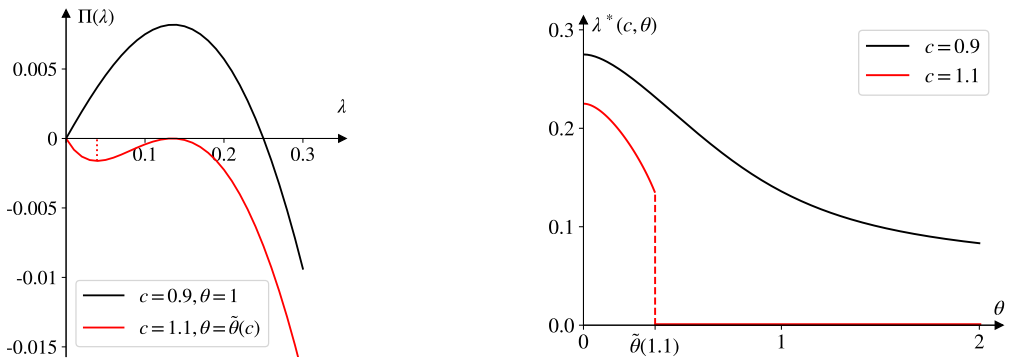
$$\lambda^* \in \begin{cases} \{\tilde{\lambda}\}, & \theta < \tilde{\theta}(c), \\ \{\tilde{\lambda}, 0\}, & \theta = \tilde{\theta}(c), \\ \{0\}, & \theta > \tilde{\theta}(c), \end{cases} \tag{2.3}$$

where $\tilde{\theta}(c) := 3c - 2 - 2\sqrt{2c(c-1)}$.

Consider the price optimization problem (2.2) parameterized by (c, θ). Let $\lambda^*(c, \theta)$ be a function that maps each (c, θ) to an optimal request conversion rate that satisfies the optimality condition (2.3). The following remark notes a surprising observation that optimal conversion rate *cannot* vary continuously with c and θ.

Remark 1 (Discontinuity)**.** *Any optimal request conversion rate $\lambda^*(c, \theta)$ is* discontinuous *at $(c, \tilde{\theta}(c))$ for all $c \in (1, 2)$.*

Figure 2.2a illustrates the shape of the profit function $\Pi(\lambda)$. When $c < 1$, the profit function is well-behaved. On the other hand, when $c > 1$, $\Pi(\lambda)$ can have a notable inflection point — as the request conversion rate increases, profit initially turns negative since the



(a) Profit function $\Pi(\lambda)$ (b) Optimal request conversion rate $\lambda^*(c, \theta)$

Figure 2.2: Pricing in the single origin-destination pair case.

price cannot justify the high dispatching cost; however, as the request conversion rate keeps increasing, it encounters a *phase transition* (after the first stationary point of the red curve marked by the red dotted line) where the increased request conversion rate generates enough cost savings to possibly make the service profitable. The discontinuity of the optimal request conversion rate (Remark 1) is a direct consequence of this phase transition. As Figure 2.2b illustrates, the discontinuity occurs at the place where the highest profit in the second phase equals zero. As θ further increases, the optimal request conversion rate has to drop to zero *abruptly*,[4] since a higher conversion rate would not reduce dispatching costs enough to make the service profitable.

We close this subsection by commenting that although the optimal price characterized in Proposition 1 maximizes the profit, it can still be limiting in the sense that the platform has a risk of not being able to match riders after quoting the price. Many of our discussions below revolve around addressing this issue.

An iterative pricing approach

[4]Under more general willingness-to-pay distributions, the optimal request conversion rate can also drop abruptly to a strictly positive value instead of zero. In Figure A.6 of Appendix A.4.1, we show a case under $c < 1$ where the highest profit in the first phase is strictly positive when the rider's willingness to pay is distributed according to a $\mathsf{Beta}(6, 2)$ distribution.

For large networks with multiple origin-destination pairs, there will be no closed form for $\bar{c}(\lambda)$ and (2.2) is computationally intensive. As a consequence, ridesharing platforms set prices by heuristic, iterative methods in practice. For example, Uber sets the price of shared rides for each origin-destination pair and time by predicting its cost [109]. These prediction or machine learning models are periodically (re)trained using historical requests and matching outcomes under past pricing policies, and then used to set new prices [107]. Similarly, Lyft uses simulation [50] to evaluate cost efficiency and update prices. Such iterative approaches can also be appealing to adapt to changes in environment that are often observed in ridesharing platforms [117, 121, 11], although this is not the focus of our analysis. Beyond ridesharing, similar methods have been applied to fleet assignment and scheduling. In these applications, demand adjusts in response to schedule and capacity modifications, with the process repeating until it converges [35, 65, 18].

We formalize this approach under our single origin-destination setting in Algorithm 1. First, the platform sets a certain initial price which induces a certain request conversion rate; then, it evaluates the expected cost per request and sets a new request conversion rate to optimize the profit given the expected cost, iterating until convergence.

Algorithm 1 An iterative pricing approach for shared rides

Require: An initial request conversion rate $\lambda^{(1)} \in (0, 1]$
1: $k \leftarrow 1$
2: **repeat**
3: $\quad \bar{c}^{(k)} \leftarrow \bar{c}(\lambda^{(k)})$ ▷ See (2.1)
4: $\quad \lambda^{(k+1)} \leftarrow \operatorname{argmax}_{\lambda \in [0,1]} \left(1 - \lambda - \bar{c}^{(k)}\right) \lambda = \frac{1}{2}\left[1 - \bar{c}^{(k)}\right]^+$ ▷ See (2.2)
5: $\quad k \leftarrow k + 1$
6: **until** convergence
7: **return** $\lambda^{(k)}$

Although this iterative pricing approach has been practiced widely, little analysis has been done to understand its performance. We first analyze the convergence of Algorithm 1. For this purpose, it will be convenient to define the following notation:

$$\lambda^+,\, \lambda^- := \frac{1}{4}\left(1 - \theta - \frac{c}{2} \pm \sqrt{\left(1 + \theta - \frac{3c}{2}\right)^2 - 2c(c-1)}\right).$$

Theorem 1 (Convergence of Iterative Pricing)**.** *Algorithm 1 converges to* λ^+*,* λ^- *or* 0 *based on the values of* c*,* θ *and initial conversion rate* $\lambda^{(1)}$ *summarized below.*

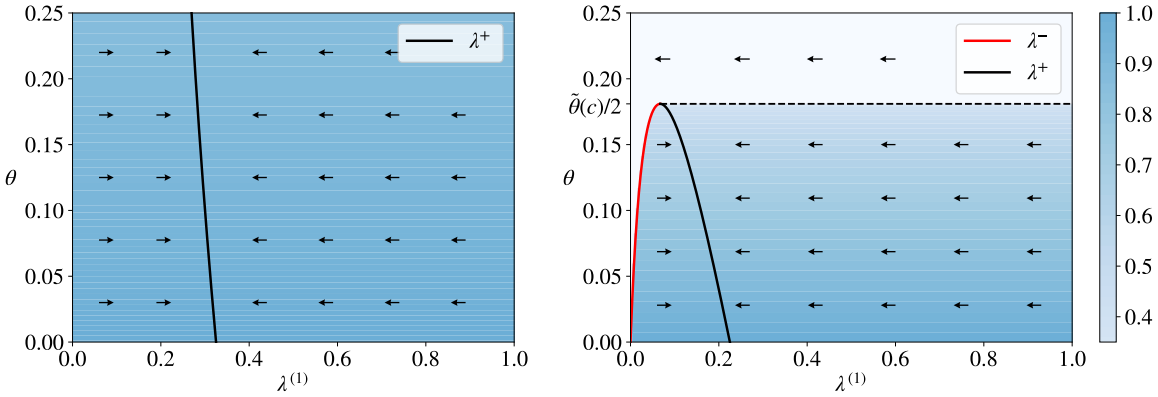
c	θ	$\lambda^{(1)}$	$\lim_{k\to+\infty}\lambda^{(k)}$
$(0,1)$	$(0,\infty)$	$(0,1]$	λ^+
$[1,2)$	$(0,\tilde{\theta}(c)/2]$	$(\lambda^-,1]$	λ^+
		λ^-	λ^-
		$(0,\lambda^-)$	0
	$(\tilde{\theta}(c)/2,\infty)$	$(0,1]$	0

Table 2.1: Convergence results of Algorithm 1.

Figure 2.3 illustrates Theorem 1. When $c \in (0,1)$ and $\theta > 0$, we have global convergence — Algorithm 1 always converges to λ^+ no matter what the initial price is. When $c \in [1,2)$ and $\theta \in (0, \tilde{\theta}(c)/2]$, where Algorithm 1 converges to depends on the initial price. Most interestingly, if the initial price is set too high, i.e., $\lambda^{(1)} \in (0, \lambda^-)$, the iterative pricing approach can lead to a market failure — it induces dynamics that gradually price the platform *out* of the market. To prevent this, platforms should initialize the iterative pricing approach with a low price. Finally, when $c \in [1,2)$ and $\theta > \tilde{\theta}(c)/2$, this approach never leads to a profitable service. Note that this is a larger unprofitable region than that induced by the optimal price characterized in Proposition 1, which is $c \in [1,2)$ and $\theta > \tilde{\theta}(c)$. This turns out not to be a coincidence, as we have the following general observation that the iterative pricing approach *always* leads to a *higher* price compared to the optimal price. Intuitively, this happens because the expected cost per request is kept constant during the price update. Thus, the platform is too optimistic (pessimistic) about raising (lowering) prices and fails to account for the fact that this will hurt (increase) conversion and increase (decrease) the *actual* cost per request.

Theorem 2 (Iterative Pricing Overprices)**.** *Using Algorithm 1, the platform always overprices its rides; namely, for any optimal request conversion rate* λ^* *for problem* (2.2)*,*

$$\lim_{k\to+\infty} \lambda^{(k)} \le \lambda^*, \quad \forall \lambda^{(1)} \in (0,1],\ \theta > 0,\ c \in (0,2).$$



(a) $c = 0.7$ (b) $c = 1.1$

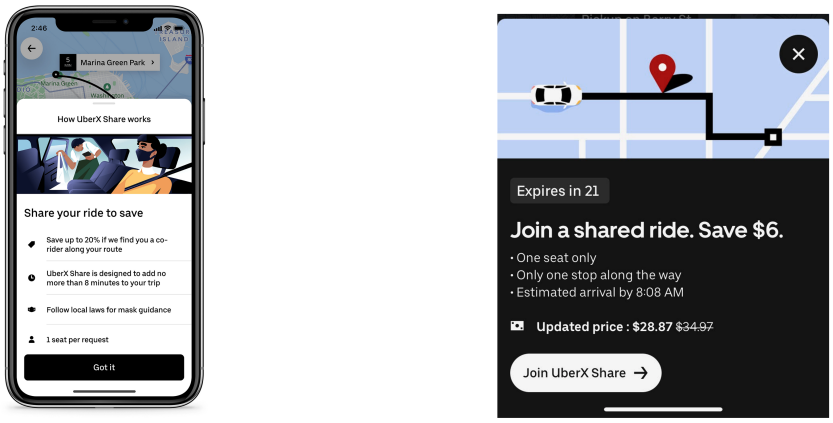
Figure 2.3: Illustration of Theorem 1 and Theorem 2. We visualize two values of c (0.7 and 1.1) as representative examples for $c \in (0, 1)$ and $c \in [1, 2)$. The color hue represents the ratio of iterative pricing's converged request conversion rate ($\lim_{k\to+\infty} \lambda^{(k)}$) to the optimal request conversion rate (λ^*), corroborating Theorem 2's result that iterative pricing overprices. The vectors represent the converging directions for the conversion rates within each area. If the initial conversion rate $\lambda^{(1)}$ lies within the blue area, it monotonically converges to λ^+ indicated by the black solid curve. If it starts inside the white area, it monotonically converges to 0. If it starts at λ^- (the red solid curve), it stays there.

Match-Based Pricing

Having discussed two status-quo approaches for pricing shared rides and their potential drawbacks, we now propose an alternative, which we call *match-based pricing*. In match-based pricing, instead of the platform bearing the risk of quoting an "upfront" price and then not being able to match riders, we allow the platform to set prices depending on the matching outcome. Then, instead of a static request conversion rate λ, there will be *state-dependent* conversion rates $\lambda(0)$ and $\lambda(1)$, where the arguments 0 and 1 refer to the states of having no requests and one request waiting on the platform (same state definition used in Figure 2.1).

A rider who arrives when there is no other request currently on the platform is quoted an *initial price*, denoted $1 - \lambda'(0)$, and the possibility of a *discount* to a price $1 - \lambda(1)$ if she ends up getting matched to a later incoming request. A rider arriving with a request waiting is directly quoted $1 - \lambda(1)$. This policy has some parallels to the current pricing

scheme of UberX Share [110], the renewed shared rides product from Uber after COVID-19, which is illustrated in Figure 2.4a. The key difference is that Uber quotes the *same* initial price to *all* riders, regardless of the state; then, the riders who are matched with another rider are given a discount ex post. This leads to different conversion behavior from our proposed policy, as we discuss in Appendix A.1.1. Another Uber product that resembles our pricing policy for a rider arriving with a request waiting is illustrated in Figure 2.4b; a pop-up offers a discount to UberX riders who can be matched with an existing, nearby rider.



(a) Riders are offered further savings if they are matched.

(b) Riders are invited to join an existing shared ride for further savings.

Figure 2.4: Two Uber products that resemble match-based pricing.

We assume that riders are risk-neutral, non-strategic, and have enough past experience with the platform to understand their probability of being matched if currently waiting to be dispatched, which is $\lambda(1)/(\lambda(1)+\theta)$. This is consistent with prior work that assumes that riders convert based on the expected value of system parameters [53, 63, 58]. Then, the rider who arrives when there is no other request currently on the platform perceives an *expected price*

$$1-\lambda(0) = (1-\lambda(1))\frac{\lambda(1)}{\lambda(1)+\theta} + (1-\lambda'(0))\frac{\theta}{\lambda(1)+\theta}, \tag{2.4}$$

and converts at rate $\lambda(0)$ (note the distinction between the expected price $1-\lambda(0)$ and the initial price $1-\lambda'(0)$)[5].

This formulation is equivalent to the following state-dependent upfront match-based pricing model: riders who arrive with no requests waiting in the system are quoted a fixed price $1-\lambda(0)$, and riders who arrive with one request waiting in the system are quoted a fixed price $1-\lambda(1)$. Although this upfront match-based pricing model suffers from fairness issues as two riders in the same vehicle are charged differently, we will analyze this system in terms of $\lambda(0)$ and $\lambda(1)$ for ease of notation; one can always convert to the previous model via (2.4). We will simply refer to these upfront prices as match-based prices from here onwards. The steady-state probabilities for this Markov chain are:

$$\pi_0 = \frac{\lambda(1)+\theta}{\lambda(0)+\lambda(1)+\theta}, \quad \pi_1 = \frac{\lambda(0)}{\lambda(0)+\lambda(1)+\theta}. \tag{2.5}$$

Consequently, the platform maximizes the expected profit per unit time $\Pi(\lambda(0),\lambda(1))$:

$$\max_{\lambda(0),\lambda(1)\in[0,1]} \quad \Pi(\lambda(0),\lambda(1)) = \frac{\lambda(0)}{\lambda(0)+\lambda(1)+\theta} \cdot \left(\lambda(1) \underbrace{(2-\lambda(0)-\lambda(1)-c)}_{\substack{\text{total profit of} \\ \text{two matched riders}}} + \theta \underbrace{(1-\lambda(0)-c)}_{\substack{\text{profit of one} \\ \text{unmatched rider}}} \right). \tag{2.6}$$

Unlike the optimal price in (2.2), which requires solving a cubic equation, surprisingly, an optimal solution for (2.6) admits a simple closed form, despite the fact that there are two prices.

Theorem 3. *For any $c \in (0,2)$ and $\theta > 0$, there exists a pair of profit-maximizing request conversion rates $(\lambda^*(0),\lambda^*(1))$ for (2.6) that satisfy $\lambda^*(0)+\lambda^*(1) = 1-c/2$. In particular,*

[5]Because riders do not convert based on the initial price, $\lambda'(0)$ is not a meaningful conversion rate, so the initial price $1-\lambda'(0)$ is allowed to be higher than 1.

the following pair is an optimal solution:

$$(\lambda^*(0), \lambda^*(1)) = \begin{cases} \left(\dfrac{(1-c/2)^2 + \theta(1-c)}{2-c+2\theta}, \dfrac{(1-c/2)^2 + \theta}{2-c+2\theta} \right), & \text{if } (1-c/2)^2 + \theta(1-c) > 0, \\ (0, \text{ any } \lambda(1) \in [0,1]), & \text{otherwise.} \end{cases} \tag{2.7}$$

Remark 2 (Continuity)**.** *Unlike the optimal λ^* for* (2.2)*, $(\lambda^*(0), \lambda^*(1))$ can be continuous in (c, θ).*

- *As θ approaches 0 (riders become patient), match-based prices approach a single price, i.e., $\lim_{\theta \to 0} \lambda^*(0) = \lim_{\theta \to 0} \lambda^*(1) = \lim_{\theta \to 0} \lambda^* = (1/2)(1-c/2)$. This single price is the optimal static price at $\theta = 0$, which can be found by replacing $\bar{c}(\lambda)$ in* (2.2) *with $\lim_{\theta \to 0} \bar{c}(\lambda) = c/2$.*

- *As θ increases, then if $c < 1$, match-based pricing approaches a policy that effectively operates solo rides and places the cost burden on the first rider, i.e., $\lim_{\theta \to +\infty} \lambda^*(0) = (1/2)(1-c)$, $\lim_{\theta \to +\infty} \lambda^*(1) = 1/2$, $\lim_{\theta \to +\infty} \rho(\lambda^*(0), \lambda^*(1)) = 0$. Meanwhile, if $c \geq 1$, then the platform eventually shuts down service, i.e., $\lim_{\theta \to +\infty} \lambda^*(0) = 0$.*

It is straightforward that the optimal $(\lambda^*(0), \lambda^*(1))$ for (2.6) leads to a higher profit than the optimal profit for (2.2), since the latter is optimal under the constraint $\lambda(0) = \lambda(1)$. Because the optimal price for (2.2) does not vary based on matching outcomes, in what follows, we refer to it as the optimal *static* price. Let $\bar{\lambda}^* = \pi_0 \lambda^*(0) + \pi_1 \lambda^*(1)$ be the average request conversion rate under the optimal match-based prices and recall that λ^* is an optimal request conversion rate for (2.2). As we formalize in Proposition 2 below, match-based prices can in fact be lower than the optimal price for (2.2) in many cases, especially when the cost is relatively high and demand density is low – a regime where many ridesharing services struggle.

Proposition 2 (Price Comparisons)**.** *For any $c \in (0,2)$, $\lambda^*(1) \geq \lambda^*$ holds for all $\theta > 0$. There exists a threshold of θ before which $\bar{\lambda}^* \leq \lambda^*$, and after which $\bar{\lambda}^* \geq \lambda^*$. Moreover,*

for any $c \in (2/3, 2)$*, there exists a threshold of* θ *before which* $\lambda^*(0) \leq \lambda^*$ *and after which* $\lambda^*(0) \geq \lambda^*$.

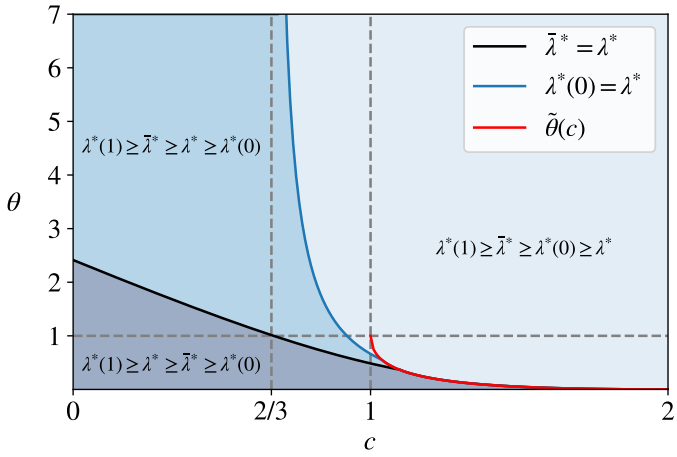


Figure 2.5: Comparisons between optimal match-based prices and static prices.

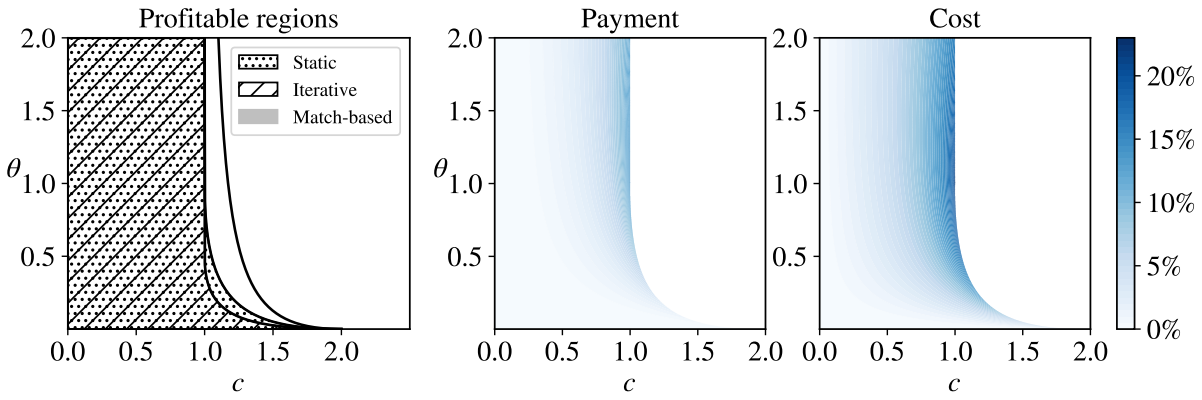
Figure 2.5 illustrates the price comparison discussed in Proposition 2. For reference, the red solid curve depicts $\tilde{\theta}(c)$, $c \geq 1$, above which the optimal static price is the highest possible, so that no riders convert to requests. Intuitively, the match-based price when there is one rider waiting is always the lowest, as the platform certainly achieves the lowest cost. The region of (c, θ) is split by a solid black curve, above which the average match-based price is lower than the optimal static price. This region of lower prices is quite generous. Recalling that the total rider arrival rate is normalized to be 1 in this analysis, Figure 2.5 shows that roughly when $c \geq 0.67$, any expected sojourn time shorter than the expected inter-arrival time leads to lower average match-based prices. What is even more interesting is that there is a region above the blue solid curve where *both* match-based prices are lower than the optimal static price.[6] This occurs *precisely* when $c > 2/3$, the cost is moderately high. With match-based prices, the platform is so willing to lower the price when there is one rider waiting that even a rider who arrives when there is no one waiting will pay

[6]On the other hand, it can be shown that $\lambda'(0) \leq \lambda^*$ always holds using eq. (2.4), i.e., under the match-based implementation, riders are quoted a higher initial price before knowing the matching outcome.

less on average, see eq. (2.4). This reduced price then leads to higher cost efficiency and profitability.

The following result gives further evidence that the increased profit from match-based pricing is achieved in an *altruistic* way — on average, each converted rider pays the *least* under match-based pricing as a result of the *highest* match rate and cost efficiency.

Proposition 3. *For any $c \in (0, 2)$ and $\theta > 0$, the payment and cost per request are lowest under match-based pricing and highest under iterative pricing; the match rate is highest under match-based pricing and lowest under iterative pricing.*



(a) Profitable regions of policies

(b) Percentage decrement of payment and cost per request

Figure 2.6: Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing).

Figure 2.6a depicts the profitable regions of the three policies. All three policies are profitable for any $c \in (0, 1)$ and $\theta > 0$. When $c \geq 1$, static pricing is profitable in $\theta \leq \tilde{\theta}(c)$, iterative pricing is profitable in $\theta \leq \tilde{\theta}(c)/2$, while match-based pricing has the largest profitable area of $\theta \leq (1 - c/2)^2 / (c - 1)$. Figure 2.6b shows the percentage decrement of payment and cost per request under match-based pricing relative to static pricing. Since static pricing induces no throughput when $\theta > \tilde{\theta}(c)$ and $c \geq 1$, we cannot compare these two policies across the boundary. Figure 2.6b suggests that the reduction of rider payment and cost is most pronounced when the cost is relatively high and rider sojourn time is relatively

short.

We conclude this subsection by pointing readers to a few extensions and generalizations provided in the Appendices. In Appendix A.1, we introduce a disutility of sharing caused by discomfort or loss of privacy, as well as an option to request solo rides. Interestingly, we obtain again closed-form solutions for match-based prices for shared rides as well as price for solo rides (see Theorem 4 in Appendix A.1), which generalizes our previous Theorem 3). Moreover, it is reassuring that all results we developed in this subsection remain valid under this generalization, showcasing the robustness of our insights. In Appendix A.1.1, we provide a discussion on the risk aspect of the pricing policies. Finally, in Online Appendix A.4, we provide a few additional robustness tests of our results including different willingness-to-pay distributions beyond uniform (A.4.1), risk aversion of riders and the platform (A.4.2), as well as matching according to a fixed batch window (A.4.3) instead of continuous matching.

2.3.2 Multiple Origin-Destination Pairs

Having discussed the benefits of match-based pricing, we now develop methods to scale the price computation to large-scale ridesharing networks. There are N types of riders with distinct origin-destination pairs. Let $[N] = \{1, \cdots, N\}$. For rider type $i \in [N]$, let Λ_i denote its total exogenous Poisson arrival rate. Put $\mathbf{\Lambda} := \Lambda_{[N]}$. Each type i rider's willingness to pay v is drawn IID from a valuation distribution $F_i(\cdot)$. Assume each rider obtains a utility from taking a ride. For a solo ride, the utility is the willingness to pay minus the payment, $v - p$. For a shared ride, the utility is further reduced by a sharing disutility, β_0, and a detour disutility, $\beta_1 t$. The sharing disutility β_0 represents the discomfort of sharing with other passengers and loss of privacy. Meanwhile, β_1 is the detour disutility coefficient and t is the detour rate — namely, the additional distance a rider travels in a shared ride, relative to the distance if she travels solo. Thus, the utility of a matched rider is $v-p-\beta_0-\beta_1 t$. This model of disutility of taking a shared ride follows similar models in the literature [58, 32, 53].

Since riders might not know in advance whether they will be matched or the amount of detour they might experience, their conversion rates depend on their match rate and expected detour. Moreover, depending on whether a rider gets matched immediately or not

upon arrival, the information received is different. Following the notion used in [56] and [9], a rider is called "passive" if she is matched with a rider arriving earlier; otherwise, she is called "active". Let p_i^1 denote the price quoted to a type i rider if matched immediately upon arrival, and p_i^0 the expected price if the rider has to wait for future dispatches (see equation (2.4) for the single-origin-destination analogue). Let ρ_i^0, ρ_i^1 be the match rates of type i requests who wait or get matched immediately (clearly, $\rho_i^1 = 1$). We let $\mathbb{E}(t_i^0)$ and $\mathbb{E}(t_i^1)$ be the expected detour rates of active and passive riders given that they are matched.[7] The sharing and detour disutilities are then $\bar{u}_i^0 := \rho_i^0(\beta_0 + \beta_1 \mathbb{E}(t_i^0))$ for an active rider, and $\bar{u}_i^1 := \beta_0 + \beta_1 \mathbb{E}(t_i^1)$ for a passive rider; and the conversion rates λ_i are $\Lambda_i \bar{F}(p_i^0 + \bar{u}_i^0)$ and $\Lambda_i \bar{F}(p_i^1 + \bar{u}_i^1)$ respectively. We assume that the inverse $p_i^0(\lambda_i)$ and $p_i^1(\lambda_i)$ are also well defined.

It is important to note that the sharing and detour utilities are a function of the pricing and matching policies deployed. This means that, technically, optimizing the pricing and matching policies needs to satisfy an equilibrium or fixed-point-type constraint where the expected sharing and detour utility is consistent with what the policies induce. This makes the problem bi-level, in addition to the curse of dimensionality of the underlying MDP. To simplify, in this subsection we assume that expected disutility are given exogenously by a related fluid optimization problem (detailed in Section 2.4 and Appendix A.2) that approximates the behavior of the system under the optimal policies.

Let θ again be the rate of the exponentially-distributed rider sojourn time. Let $d = (i, j)$ be a *dispatch* of two matched requests of types i and j. A dispatch can also be a singleton $d = (i)$ for an unmatched request. Denote the minimum driving cost of a dispatch d the platform incurs by c_d. One natural property we have is that $c_{(i)} = c_{(i,i)} \leq c_{(i,j)}$, $\forall i, j \in [N]$.

Let the state of the system at time t be $\mathbf{s}_t = (s_{t,1}, \cdots, s_{t,N}) \in \mathcal{S}$ where $s_{t,i} \in \mathbb{N}$ is the number of type i riders at time t waiting to be dispatched and $\mathcal{S}$ is the state space. The platform deploys the following state-dependent *pricing* and *matching* policies.

- *Pricing Policy.* We describe a state-dependent pricing policy: when a type i request

[7] We assume that even if a rider is matched immediately upon arrival, she is not able to observe the exact amount of detour before requesting and has to take expectation over potential matches. This is consistent with the current implementation at Uber (Figure 2.4b), which does not disclose detour information when offering discounts. Nevertheless, it is straightforward to modify our model and computational approach for this case.

arrives at state $\mathbf{s}_t$, the platform uses pricing policy $\lambda_i : \mathcal{S} \mapsto [0, \Lambda_i]$ to regulate the request rate of type i riders. In Section 2.3.1, we previously showed that for a single origin-destination pair, we can recover *match-based* prices with simple closed-form equations. In Appendix A.5.2, we show that we can do the same for multiple origin-destination pairs.

- *Matching Policy.* The platform deploys a state-dependent matching policy $\phi_i : \mathcal{S} \mapsto [N] \cup \{0\}$. When a type i request arrives at state $\mathbf{s}_t$, this matching policy either (1) immediately matches the arriving request with some available request of type $i' \in [N]$ with $s_{t,i'} > 0$, or (2) lets the arriving request wait in the system for future dispatch, which we label as option 0.

Denote the pricing and matching policies for all types of riders by $\boldsymbol{\lambda} := \lambda_{[N]}$ and $\boldsymbol{\phi} := \phi_{[N]}$. Let $A^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)$ be the counting process of the number of requests up to time t under the pricing and matching policies $\boldsymbol{\lambda}$ and $\boldsymbol{\phi}$. For the n^{th} request arrival, $n \in [A^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)]$, let $\delta_n \in [N]$ be the type of the rider and t_n be its arrival time. Let $D^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)$ be the set of dispatches up to time t. The total profit collected by time t is given by

$$\Pi^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t) = \sum_{\substack{n \in [A^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)]: \\ \phi_{\delta_n}(\mathbf{s}_{t_n})=0}} p^0_{\delta_n}(\lambda_{\delta_n}(\mathbf{s}_{t_n})) + \sum_{\substack{n \in [A^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)]: \\ \phi_{\delta_n}(\mathbf{s}_{t_n})>0}} p^1_{\delta_n}(\lambda_{\delta_n}(\mathbf{s}_{t_n})) - \sum_{d \in D^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)} c_d.$$

We consider stationary policies and hereafter drop the subscript t in the state variable $\mathbf{s}_t$.

The objective of the platform is to maximize expected long-run average profit via

$$g := \sup_{\boldsymbol{\lambda},\boldsymbol{\phi}} \liminf_{t \to +\infty} \frac{1}{t} \mathbb{E}[\Pi^{\boldsymbol{\lambda},\boldsymbol{\phi}}(t)] \tag{2.8}$$

where g is defined as the maximum expected long-run average profit per unit time and the expectation is taken over all the randomness of rider arrivals, requests, and dispatches.

We have the following observation regarding the structure of the optimal matching policy.

Lemma 1. *For any pricing policy $\boldsymbol{\lambda}$ and rider type $i \in [N]$, there exists an optimal matching policy $\boldsymbol{\phi}$ under which any state $\mathbf{s}$ with $s_i \geq 3$ for some $i \in [N]$ is transient. In addition, if*

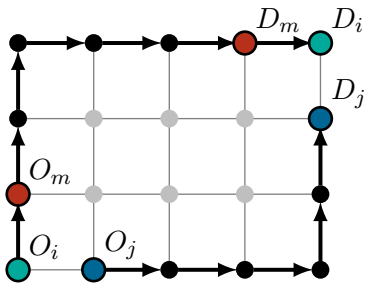


Figure 2.7: An example when two requests of the same type i might be held to enable future matches (i, m) and (i, j).

a rider of type i satisfies $c_{(i,i)} + c_{(m,j)} \leq c_{(i,m)} + c_{(i,j)}$, $\forall m, j \in [N]$, then any state $\mathbf{s}$ with $s_i \geq 2$ is transient.

This result says that regardless of the pricing policy, there always exists an optimal matching policy that never lets *more than two* requests of the same type wait in the system. (Although it might not be immediately obvious why two requests of the same type would ever be left in the system, Figure 2.7 illustrates an example where two requests of type i might be held to enable future matches (i, m) and (i, j); this has lower cost than that of matches (i, i) and (m, j).) Moreover, when the condition $c_{(i,i)} + c_{(m,j)} \leq c_{(i,m)} + c_{(i,j)}$, $\forall j, m \in [N]$ is met for all rider types $i \in [N]$, there exists an optimal matching policy that never lets *more than one* request of the same type wait. This condition is usually referred to as the *weak Monge condition* [20] and emphasizes the value of self-matches. We show that the condition is satisfied for requests in *one* dimension (see Claim 2 and its proof in Appendix A.3). Lemma 1 suggests that, without loss of optimality, one can limit the state space to $\mathcal{S} = \{0, 1, 2\}^N$, or $\mathcal{S} = \{0, 1\}^N$ under the weak Monge condition. This renders the state space finite and countable.

The problem of finding an optimal pricing and matching policy is an average-reward, continuous-time Markov Decision Process (MDP). It can be seen that for any pricing and matching policy, the resulting continuous-time Markov chain is a finite-state *unichain* — there is one recurrent class (containing any states reachable from $\mathbf{s} = \mathbf{0}$, which is reachable from all other states) and the rest are transient states. We can thus rewrite our *continuous-time* MDP as a *discrete-time* MDP using the uniformization method [89, Ch. 11.5.1]. Let

$M = \sum_{i=1}^{N}(\Lambda_i + 2\theta)$ be the maximum rate of transition, so that $1/M$ is the expected length of one time period. According to Theorem 8.4.5 of [89], an optimal policy exists and there exists a *relative* value function $V : \mathcal{S} \mapsto \mathbb{R}$ satisfying the following optimality equation:

$$V(\mathbf{s}) + \frac{g}{M} = \max_{\mathbf{0}\le\boldsymbol{\lambda}(\mathbf{s})\le\boldsymbol{\Lambda}} \frac{1}{M}\Bigg(\sum_{i\in[N]} \max\Bigg\{ \underbrace{\lambda_i(\mathbf{s})}_{\substack{\text{conversion rate of riders}\\ \text{matched upon arrival}}} \Bigg(\underbrace{p_i^1(\lambda_i(\mathbf{s}))}_{\substack{\text{revenue from}\\ \text{rider type } i}} + \max_{j\in[N]: s_j>0} \underbrace{V(\mathbf{s}-\mathbf{e}_j) - c_{(i,j)}}_{\text{match } i \text{ with } j} \Bigg), \\ \underbrace{\lambda_i(\mathbf{s})}_{\substack{\text{conversion rate of}\\ \text{riders let wait}}} \Bigg(\underbrace{p_i^0(\lambda_i(\mathbf{s}))}_{\substack{\text{revenue from}\\ \text{rider type } i}} + \underbrace{V(\mathbf{s}+\mathbf{e}_i)}_{\text{let } i \text{ wait}} \Bigg) \Bigg\} + \sum_{\substack{i\in[N]:\\ s_i>0}} \underbrace{\theta s_i}_{\substack{\text{reneging rate of}\\ \text{waiting requests}}} \Bigg(\underbrace{V(\mathbf{s}-\mathbf{e}_i) - c_{(i)}}_{\text{dispatch } i \text{ solo}} \Bigg) \Bigg) \\ + \underbrace{\frac{M - \sum_{i\in[N]}\lambda_i(\mathbf{s}) - \sum_{i\in[N]:\, s_i>0}\theta s_i}{M} V(\mathbf{s})}_{\text{no event occurs in one time period}}, \quad \forall \mathbf{s}\in\mathcal{S}, \tag{2.9}$$

where g/M is the maximum profit per period and $\mathbf{e}$ is an N-dimensional vector of all ones, and similarly $\mathbf{e}_i$ is an N-dimensional unit vector whose i^{th} element is one and the rest are zeroes. Rearranging the terms yields,

$$g = \max_{\mathbf{0}\le\boldsymbol{\lambda}(\mathbf{s})\le\boldsymbol{\Lambda}} \sum_{i\in[N]} \max\Big\{ \lambda_i(\mathbf{s})\Big(p_i(\lambda_i(\mathbf{s})) + \max_{j\in[N]:\, s_j>0} V(\mathbf{s}-\mathbf{e}_j) - c_{(i,j)} - V(\mathbf{s})\Big), \\ \lambda_i(\mathbf{s})\Big(p_i(\lambda_i(\mathbf{s})) + V(\mathbf{s}+\mathbf{e}_i) - V(\mathbf{s})\Big)\Big\} \\ + \sum_{i\in[N]:\, s_i>0} \theta s_i \left(V(\mathbf{s}-\mathbf{e}_i) - c_{(i)} - V(\mathbf{s})\right), \quad \forall \mathbf{s}\in\mathcal{S}. \tag{2.10}$$

To find the optimal pricing and matching policies, we reformulate (2.10) as the following linear program.

$$\begin{aligned} \min_{g,\, V,\, \Gamma,\, \Psi} \quad & g \\ \text{s.t.} \quad & g \ge \sum_{i\in[N]} \Gamma_{\mathbf{s},i} + \sum_{i\in[N]:\, s_i>0} \theta s_i (V(\mathbf{s}-\mathbf{e}_i) - c_{(i)} - V(\mathbf{s})), \quad \forall \mathbf{s}\in\mathcal{S}, \\ & \Gamma_{\mathbf{s},i} \ge \lambda_i(\mathbf{s})\Big(p_i^1(\lambda_i(\mathbf{s})) - \Psi_{\mathbf{s},i}\Big), \quad \forall i\in[N],\, \forall \mathbf{s}\in\mathcal{S}, \forall \lambda_i(\mathbf{s})\in[0,\Lambda_i], \end{aligned} \tag{2.11a}$$

$$\Gamma_{\mathbf{s},i} \geq \lambda_i(\mathbf{s})\Big(p_i^0(\lambda_i(\mathbf{s})) + V(\mathbf{s}+\mathbf{e}_i) - V(\mathbf{s})\Big), \quad \forall i \in [N],\ \forall \mathbf{s} \in \mathcal{S}, \forall \lambda_i(\mathbf{s}) \in [0, \Lambda_i], \tag{2.11b}$$

$$\Psi_{\mathbf{s},i} \leq c_{(i,j)} + V(\mathbf{s}) - V(\mathbf{s}-\mathbf{e}_j), \quad \forall i,j \in [N],\ \forall \mathbf{s} \in \mathcal{S} :\ s_j > 0. \tag{2.11c}$$

Auxiliary variables $\Gamma_{\mathbf{s},i}$ measure the relative profit of a new incoming request of type i in state $\mathbf{s}$, and $\Psi_{\mathbf{s},i}$ represent the relative dispatching cost for a new type i request when matched immediately upon arrival. Constraints (2.11a), (2.11c), and (2.11b) ensure that the platform makes the best pricing and matching decisions. Specifically, constraints (2.11a) and (2.11b) compare the profit of immediately matching i upon arrival with that of delaying matching i for future requests; and constraints (2.11c) make sure that the most cost-effective matching option is chosen.

There are two computational challenges in solving Problem (2.11). First, there are an infinite number of constraints (2.11a) and (2.11b), which are enumerated over all possible pricing decisions. Second, the size of the state space $\mathcal{S}$ is 3^N. To address these issues, we use cutting planes, value function approximation, and constraint sampling, which we detail below.

Cutting planes. The infinite number of constraints (2.11a) and (2.11b) can be circumvented via a cutting-plane approach. We begin with an arbitrary pricing policy $\bar{\boldsymbol{\lambda}}(\mathbf{s})$, and initialize the linear program only with constraints (2.11a) and (2.11b) for that pricing policy (along with all of constraints (2.11c)). Solving this initial linear program, we obtain the optimal g^*, V^*, $\Gamma^*_{\mathbf{s},i}$, and $\Psi^*_{\mathbf{s},i}$. For each state $\mathbf{s} \in \mathcal{S}$ and each type of request $i \in [N]$, we then solve two simple price optimization problems for constraints (2.11a) and (2.11b):

$$z^{1*}_{\mathbf{s},i} = \max_{\lambda_i(\mathbf{s}) \in [0,\Lambda_i]} \lambda_i(\mathbf{s})(p_i^1(\lambda_i(\mathbf{s})) - \Psi^*_{\mathbf{s},i}), \tag{2.12a}$$

$$z^{0*}_{\mathbf{s},i} = \max_{\lambda_i(\mathbf{s}) \in [0,\Lambda_i]} \lambda_i(\mathbf{s})(p_i^0(\lambda_i(\mathbf{s})) + V^*(\mathbf{s}+\mathbf{e}_i) - V^*(\mathbf{s})), \tag{2.12b}$$

and let their optimal solutions be $\lambda_i^{0*}(\mathbf{s})$ and $\lambda_i^{1*}(\mathbf{s})$, respectively. Then, we check whether (2.11a) and (2.11b) is satisfied for all $\mathbf{s} \in \mathcal{S}$. If (2.11a) is violated for some $\bar{\mathbf{s}} \in \mathcal{S}$, i.e., $\Gamma^*_{\mathbf{s},i} < z^{1*}_{\mathbf{s},i}$, then a set of new constraints for pricing decisions $[\lambda_i^{1*}(\bar{\mathbf{s}})]_{i\in[N]}$ for all such

violated state $\bar{\mathbf{s}}$ is added to (2.11a). Similarly, if (2.11b) is violated for some $\bar{\mathbf{s}} \in \mathcal{S}$, i.e., $\Gamma^*_{\mathbf{s},i} < z^{0*}_{\mathbf{s},i}$, then a set of new constraints for pricing decisions $[\lambda_i^{0*}(\bar{\mathbf{s}})]_{i\in[N]}$ for all such violated state $\bar{\mathbf{s}}$ is added to (2.11b). The linear program is then re-solved. This procedure is repeated until no more constraints can be added, and converges to an optimal solution (see a formal proof in [81]).

Value function approximation. Problem (2.11) is a linear program, but as the size of the state space is 3^N, it still suffers from the curse of dimensionality. We reduce the number of variables V using value function approximation. In particular, we consider an affine approximation with $V(\mathbf{s}) \approx \alpha_0 - \sum_{i\in[N]} \alpha_i s_i$, where α_0 is a constant offset and α_i for $i \in [N]$ weighs the marginal cost of a rider of type i that is waiting in state $\mathbf{s} \in \mathcal{S}$. This approximation gives the following approximate linear program [94, 30].

$$
\begin{aligned}
(\mathsf{ALP}) \quad \min_{g,\,\alpha,\,\Gamma,\,\Psi} \quad & g \\
\text{s.t.} \quad & g \geq \sum_{i\in[N]} \Gamma_{\mathbf{s},i} + \sum_{i\in[N]:\, s_i>0} \theta s_i(\alpha_i - c_{(i)}), \quad \forall \mathbf{s} \in \mathcal{S}, \\
& \Gamma_{\mathbf{s},i} \geq \lambda_i(\mathbf{s})\Big(p_i^1(\lambda_i(\mathbf{s})) - \Psi_{\mathbf{s},i}\Big), \quad \forall i \in [N],\ \forall \mathbf{s} \in \mathcal{S}, \forall \lambda_i(\mathbf{s}) \in [0, \Lambda_i], \\
& \Gamma_{\mathbf{s},i} \geq \lambda_i(\mathbf{s})\Big(p_i^0(\lambda_i(\mathbf{s})) - \alpha_i\Big), \quad \forall i \in [N],\ \forall \mathbf{s} \in \mathcal{S},\ \forall \lambda_i(\mathbf{s}) \in [0, \Lambda_i], \\
& \Psi_{\mathbf{s},i} \leq c_{(i,j)} - \alpha_j, \quad \forall i, j \in [N],\ \forall \mathbf{s} \in \mathcal{S} :\ s_j > 0.
\end{aligned}
$$

Constraint sampling. A final challenge in solving ALP is that there remains a large number of constraints, since each state $\mathbf{s} \in \mathcal{S}$ has its own constraints. A practical technique to address this is through *constraint sampling* on the state space $\mathcal{S}$. In particular, instead of including constraints for all states $\mathbf{s} \in \mathcal{S}$, one can consider constraints only for states $\mathbf{s} \in \hat{\mathcal{S}}$, where $\hat{\mathcal{S}} \subseteq \mathcal{S}$ is a set of states sampled from a simulation of an initial pricing policy $\hat{\boldsymbol{\lambda}}$ and matching policy $\hat{\boldsymbol{\phi}}$. Resolving ALP with the sampled states, $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\phi}}$ can be updated accordingly, and the procedure can then iterate. [31] show that when states are sampled from the state distribution induced by the optimal policy (or a "similar" distribution), only a polynomial number (in N) of sampled states are needed to get a near-optimal solution with high probability. Hence, constraint sampling has been a common tool in practical

implementations of large-scale ALPs [38, 66, 83].

Once the optimal $\alpha^*_{[N]}$ are obtained from solving ALP, we compute $\Psi^*_{\mathbf{s},i} = \min_{j\in[N]:\, s_j>0} \{c_{(i,j)} - \alpha^*_j\}$. Substituting terms in (2.12), we have $z^{1*}_{\mathbf{s},i} = \max_{\lambda_i(\mathbf{s})\in[0,\Lambda_i]} \lambda_i(\mathbf{s})(p^1_i(\lambda_i(\mathbf{s})) - \Psi^*_{\mathbf{s},i})$ and $z^{0*}_{\mathbf{s},i} = \max_{\lambda_i(\mathbf{s})\in[0,\Lambda_i]} \lambda_i(\mathbf{s})(p^0_i(\lambda_i(\mathbf{s})) - \alpha^*_i)$. Again, let the corresponding optimal conversions be $\lambda^{1*}_i(\mathbf{s})$ and $\lambda^{0*}_i(\mathbf{s})$. This yields the pricing policy:

$$\lambda^*_i(\mathbf{s}) \in \begin{cases} \lambda^{1*}_i(\mathbf{s}), & \text{if } z^{1*}_{\mathbf{s},i} \geq z^{0*}_{\mathbf{s},i}, \\ \lambda^{0*}_i(\mathbf{s}), & \text{otherwise.} \end{cases} \tag{2.14}$$

And the matching policy at state $\mathbf{s}$ for an incoming type i request is given by

$$\phi^*_i(\mathbf{s}) \in \begin{cases} \arg\min_{j\in[N]:\, s_j>0} \{c_{(i,j)} - \alpha^*_j\}, & \text{if } z^{1*}_{\mathbf{s},i} \geq z^{0*}_{\mathbf{s},i}, \\ \{0\}, & \text{otherwise.} \end{cases} \tag{2.15}$$

Intuitively, depending on which profit is higher, request i is either kept waiting in the system, which collects a profit of $p^0_i(\lambda^*_i(\mathbf{s})) - \alpha^*_i$ (where α^*_i is the marginal cost of having request i wait in the system); or is matched to another request of type j currently in the system with the lowest cost $\min_{j\in[N]:\, s_j>0} c_{(i,j)} - \alpha^*_j$ (the actual dispatching cost $c_{(i,j)}$ minus the marginal cost α^*_j of letting a request j waiting in the system), collecting a profit of $p^1_i(\lambda^*_i(\mathbf{s})) - (c_{(i,j)} - \alpha^*_j)$.

Recovering equivalent match-based prices. In Appendix A.5.2, we show an equivalent implementation discussed at the beginning of Section 2.3.1. In particular, the platform offers initial prices p'_i to each request of type i if they are kept waiting in the system for future matches upon arrival. If a request of type i is eventually matched with a request of type j, the type i request pays a discounted fare $p_{i,j}$ and the type j request pays $p_{j,i}$ ($p_{i,j} \neq p_{j,i}$ in general). We show that these equivalent match-based prices can be recovered in closed-form based on $\boldsymbol{\lambda}^*, \boldsymbol{\phi}^*$ defined in (2.14) and (2.15) above.

2.4 Computational Experiments

In this section, we present numerical experiments based on ridesharing data in Chicago. Code and data to reproduce all computational experiments in the chapter can be accessed at `https://github.com/ORJournal/2023.0513`. The data provided by Chicago Data Portal[8] describes the time and location of pickup and dropoff of each trip, as well as whether the rider is willing to share the ride ("authorized"). We focus on authorized trips (comprising about 15% of all trips) recorded throughout eight weeks in October and November of 2019 before COVID-19. The first seven weeks are used for training and the last week as a test set. All trips started and ended within the 76 community areas of the City of Chicago[9] (excluding the O'Hare International Airport). Since the data set does not provide information about those riders who arrive without requesting a ride, we use trip data as a proxy of arrival data as commonly assumed in related studies.

We focus on the one-hour time window of 7:30 – 8:30 a.m. on Monday for each week. The total arrival rate in this time window is 28.25 riders per minute in the training set, and 29.43 riders per minute in the test set. To identify the request types, we follow the two-step clustering method of [9]: first, we cluster the 76 community areas into 42 zones; then, we cluster the riders originating from each pickup zone by their dropoff locations using k-means clustering. We obtain in total 244 rider types. Riders in the test sets are mapped to the closest of the 244 types from the training set. We consider mean rider sojourn times in $\{0.5, 1, 3, 5\}$ minutes. Riders' per-mile willingness-to-pay is uniformly distributed between 0 and 1 in dollars. We consider three per-mile costs $c \in \{0.7, 0.9, 1.1\}$ in dollars.[10] We consider the per-mile sharing discomfort $\beta_0 = 0.05$ (that is, 5% of the rider's maximum possible valuation), and the per-mile detour disutility $\beta_1 = 0.675$,[11] both in dollar values. We also impose a maximum detour rate of 0.25, modifying (2.9) accordingly so that matching

[8]Source: `https://tinyurl.com/4dcn9d5d`, accessed 2023-06-30.

[9]Source: `https://tinyurl.com/4yxcvfaj`, accessed 2023-06-30.

[10]This is representative of Uber driver per-distance rate in regions in the US, see `https://rideshareguru.com/uber-driver-pay-rates-by-city-per-mile-and-per-minute/`, accessed 2023-06-30.

[11]β_1 is set using Chicago's minimum wage of \$16.2 per hour, divided by a uniform driving speed of 24 mph [105]. The wage data is sourced from `https://www.chicago.gov/city/en/depts/mayor/press_room/press_releases/2024/july/paid-time-off-day.html`, accessed 2024-08-10.

i with j is only considered if they satisfy the maximum detour constraint. The following three pricing policies are then compared.

- *Static pricing.* Consider a vector of static prices quoted for each rider type with conversion rates $[\lambda_i]_{i\in[N]}$. Let $\Pi(\lambda_1, \cdots, \lambda_N)$ be the maximum expected profit under these request conversion rates when matching decisions are made optimally. This pricing policy finds optimal static prices that maximize $\Pi(\lambda_1, \cdots, \lambda_N)$. This is a hard problem even without matching, as we analyze in Section 2.3.1. It is also not easy to model as a discrete-time MDP, as pricing decisions are coupled across states. We instead use a fluid mathematical program that provably upper bounds $\Pi(\lambda_1, \cdots, \lambda_N)$, and find static prices by optimizing the pricing decisions in this fluid formulation (see Appendix A.2). Then, pricing decisions are fixed at these static prices, and matching decisions are optimized by solving ALP.

- *Iterative pricing.* This is a generalization of Algorithm 1 when riders have multiple origin-destinations. We give each rider type an initial price and sharing and detour disutility, then solve for the optimal matching using ALP. We then simulate the resulting policy and estimate the average cost $\bar{c}_i$ for each rider type i. The cost of a shared ride to each rider involved is decomposed into the sum of road segment costs, where the cost of each road segment is allocated equally between the riders that travel in the vehicle along that road segment (see Section A.5.1 in Appendix A.5 for a detailed illustration). We also estimate sharing and detour disutility through simulation, which forms a new price-demand function $p_i(\lambda_i)$. We then update the price of each rider type by solving the price optimization problem: $\max_{\lambda_i\in[0,\Lambda_i]:p_i(\lambda_i)\geq 0} \lambda_i(p_i(\lambda_i)-\bar{c}_i)$. We repeat this procedure until convergence.

- *Match-based pricing.* This is the proposed policy where both state-dependent pricing and matching decisions are obtained by solving ALP using methods discussed in Section 2.3.2.

As mentioned at the end of Section 2.3.2, all ALP-type formulations are implemented

with constraint sampling. The sharing and detour disutilities in the ALP are initially fixed based on the solution of the fluid model (see derivation in Appendix A.2). States are first sampled from a baseline matching and pricing policy (e.g., a static pricing policy with greedy matching, which immediately matches two requests as long as there is positive cost saving). The sampling is done by adding an arrival process of "observers" into the arrival process of riders, and the current state of the system is sampled as soon as an observer arrives. States can then be resampled under the updated policy, sharing and detour disutility can be updated, and this procedure can be repeated multiple times to pick the policy with the best performance (see, e.g., [38]). It is possible to use real arrivals from the Chicago data to train the model, but this limits the size of the training set. Instead, we generate a synthetic training set with the arrival rates estimated from the real training data under a long simulation time, which ensures that sufficiently many constraints are sampled to generate high-quality solutions. As we mentioned in Section 2.3.2, sharing and detour disutility is a function of the pricing and matching policy. We thus evaluate the performance of the policies by iteratively updating the disutility and corresponding conversion behaviors through repeated simulations until convergence.

The performance of different pricing policies in the test set is presented in Table 2.2. The corresponding result for the training set can be found in Table A.2 of Appendix A.5.3. Performance metrics include, for different values of c and θ: the profit rate per minute, the average quoted price (per rider, per unit distance), the average quoted disutility (per rider, per unit distance, and defined as the average quoted price plus the average *expected* disutility from matching and detour), the average payment (per converted request, per unit distance), average realized disutility (per converted request, per unit distance, and defined as the average payment plus the average disutility *experienced* from matching and detour among the converted requests), the throughput (requests per minute), the match rate (number of matched requests divided by total requests), the cost efficiency (one minus total dispatching cost divided by a hypothetical cost if all requests were dispatched solo), and the average detour rate (over matched requests, and defined as the total detour length divided by a hypothetical total distance of the matched requests assuming all of them are dispatched solo). "Iterative", "Static" and "Match" refer to the metrics under the aforementioned three

policies; "Match vs. X" denotes the percentage increment under the match-based policy over policy X.

The results show that the match-based policy delivers substantial profit improvement over the other two policies, especially when c is relatively high. Notably, when $c = 1.1$, there exist sojourn times where the service is unprofitable under all pricing policies. However, match-based pricing is still able to maintain a profitable shared rides service in higher-patience instances.

Moreover, corroborating and strengthening Proposition 2, the results demonstrate that the match-based policy induces lower quoted prices than those under the iterative policy, and also lower than those under the static policy. The match-based policy also consistently delivers lower payments per converted request and higher match rate and cost efficiency, consistent with Proposition 3. Meanwhile, the throughput of the match-based policy increases significantly in all instances, especially compared with the iterative policy, demonstrating improved accessibility to the service.

Match-based pricing also benefits riders by achieving the minimal average detour rate among matched requests. This is driven by high-quality matches, as demonstrated by the high cost efficiency. This lower detour rate, combined with reduced quoted prices and payments, leads to lower total disutilities (both quoted and realized), thereby enhancing rider surplus.

We further visualize the geographical distribution of seven test-set performance metrics (average quoted price, average quoted disutility, average payment, average realized disutility, cost efficiency, match rate, and average detour rate) under three policies in Figures 2.8, 2.9, and 2.10, focusing on the parameter settings of $c = 0.9$ and average sojourn time of 3 minutes. For all policies, the downtown zones (center-right) have lower quoted prices, lower payments, lower quoted and realized disutilities, higher cost efficiency, higher match rate, and lower detour rate than the rest of the regions thanks to higher demand density.

To further illustrate the relationship between demand density and policy differences, we plot the distribution of arrival rates in Figure 2.11a, and the relative differences between the match-based and static policies in Figure 2.11b (quoted price), Figure 2.11c (quoted disutility), Figure 2.11d (payment), Figure 2.11e (realized disutility) and Figure 2.11f (de-

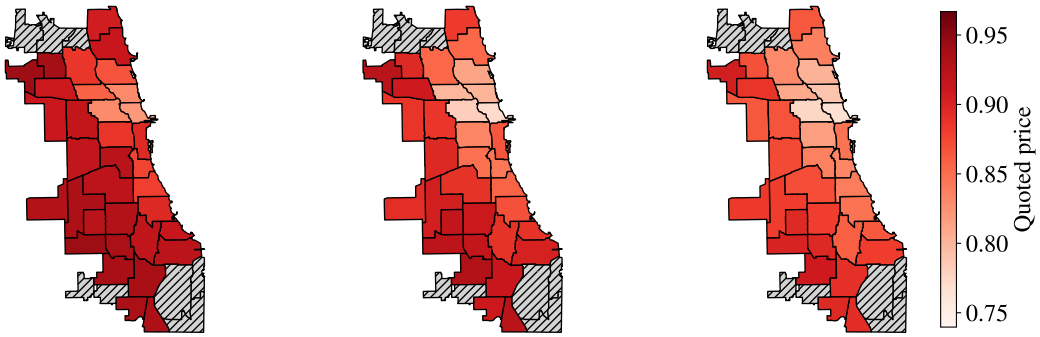
Cost ($/mile)		$c = 0.7$				$c = 0.9$				$c = 1.1$			
Sojourn Time $1/\theta$ (min)		0.5	1	3	5	0.5	1	3	5	0.5	1	3	5
Profit ($/min)	Iterative	3.210	3.516	4.379	4.994	0.415	0.501	0.899	1.262	0.000	0.000	0.000	0.000
	Static	3.251	3.632	4.537	5.077	0.433	0.571	1.184	1.569	0.000	0.000	0.000	0.000
	Match	3.431	3.889	4.844	5.462	0.596	0.850	1.538	2.039	0.000	0.000	0.137	0.280
	Match vs. Iterative	6.9%	10.6%	10.6%	9.4%	43.8%	69.6%	71.0%	61.5%	-	-	∞	∞
	Match vs. Static	5.5%	7.1%	7.2%	7.6%	37.6%	48.9%	29.9%	30.0%	-	-	∞	∞
Quoted Price ($/mile · rider)	Iterative	0.817	0.802	0.770	0.756	0.932	0.919	0.887	0.864	1.000	1.000	1.000	1.000
	Static	0.811	0.791	0.752	0.734	0.922	0.898	0.848	0.823	1.000	1.000	1.000	1.000
	Match	0.803	0.783	0.744	0.726	0.906	0.883	0.836	0.817	1.000	1.000	0.937	0.902
	Match vs. Iterative	-1.7%	-2.3%	-3.3%	-4.0%	-2.8%	-4.0%	-5.8%	-5.5%	0.0%	0.0%	-6.3%	-9.8%
	Match vs. Static	-1.0%	-1.0%	-1.0%	-1.1%	-1.7%	-1.7%	-1.4%	-0.8%	0.0%	0.0%	-6.3%	-9.8%
Quoted Disutility ($/mile · rider)	Iterative	0.837	0.829	0.808	0.796	0.941	0.936	0.916	0.899	1.000	1.000	1.000	1.000
	Static	0.831	0.817	0.788	0.774	0.932	0.917	0.878	0.859	1.000	1.000	1.000	1.000
	Match	0.823	0.808	0.779	0.763	0.921	0.903	0.868	0.850	1.000	1.000	0.959	0.929
	Match vs. Iterative	-1.7%	-2.6%	-3.5%	-4.1%	-2.2%	-3.5%	-5.3%	-5.5%	0.0%	0.0%	-4.1%	-7.1%
	Match vs. Static	-0.9%	-1.1%	-1.1%	-1.5%	-1.2%	-1.5%	-1.2%	-1.1%	0.0%	0.0%	-4.1%	-7.1%
Payment ($/mile · request)	Iterative	0.816	0.798	0.764	0.749	0.929	0.913	0.866	0.836	-	-	-	-
	Static	0.808	0.785	0.743	0.725	0.915	0.883	0.820	0.794	-	-	-	-
	Match	0.787	0.763	0.722	0.705	0.869	0.836	0.788	0.771	-	-	0.835	0.814
	Match vs. Iterative	-3.5%	-4.5%	-5.5%	-5.9%	-6.5%	-8.4%	-9.0%	-7.8%	-	-	-	-
	Match vs. Static	-2.6%	-2.9%	-2.8%	-2.8%	-5.0%	-5.3%	-3.9%	-3.0%	-	-	-	-
Realized Disutility ($/mile · request)	Iterative	0.835	0.825	0.800	0.787	0.939	0.929	0.894	0.870	-	-	-	-
	Static	0.828	0.812	0.780	0.766	0.927	0.906	0.854	0.833	-	-	-	-
	Match	0.810	0.792	0.760	0.745	0.891	0.865	0.826	0.809	-	-	0.872	0.851
	Match vs. Iterative	-3.0%	-4.1%	-5.0%	-5.4%	-5.1%	-6.9%	-7.6%	-6.9%	-	-	-	-
	Match vs. Static	-2.2%	-2.5%	-2.5%	-2.7%	-3.9%	-4.5%	-3.3%	-2.9%	-	-	-	-
Throughput (requests/min)	Iterative	4.725	4.930	5.527	5.831	1.715	1.837	2.397	2.851	0.000	0.000	0.000	0.000
	Static	4.930	5.309	6.083	6.445	1.977	2.382	3.491	4.018	0.000	0.000	0.000	0.000
	Match	5.157	5.569	6.370	6.794	2.306	2.784	3.811	4.297	0.000	0.000	1.220	2.031
	Match vs. Iterative	9.2%	13.0%	15.2%	16.5%	34.4%	51.6%	59.0%	50.7%	-	-	∞	∞
	Match vs. Static	4.6%	4.9%	4.7%	5.4%	16.6%	16.9%	9.2%	6.9%	-	-	∞	∞
Match Rate	Iterative	0.227	0.331	0.499	0.573	0.113	0.204	0.411	0.519	-	-	-	-
	Static	0.237	0.344	0.519	0.609	0.147	0.280	0.490	0.583	-	-	-	-
	Match	0.296	0.401	0.572	0.640	0.300	0.410	0.581	0.639	-	-	0.626	0.651
Cost Efficiency	Iterative	0.060	0.096	0.176	0.218	0.029	0.057	0.140	0.194	-	-	-	-
	Static	0.064	0.105	0.187	0.227	0.038	0.079	0.176	0.217	-	-	-	-
	Match	0.093	0.138	0.220	0.258	0.095	0.145	0.226	0.263	-	-	0.257	0.284
Detour Rate (over matched requests)	Iterative	0.067	0.062	0.048	0.039	0.062	0.056	0.043	0.036	-	-	-	-
	Static	0.066	0.057	0.046	0.041	0.064	0.061	0.043	0.039	-	-	-	-
	Match	0.055	0.047	0.038	0.032	0.049	0.042	0.036	0.027	-	-	0.023	0.018

Table 2.2: Test results on Chicago data ($\beta_0 = 0.05, \beta_1 = 0.675$), averaged over 100 simulation runs using test data.

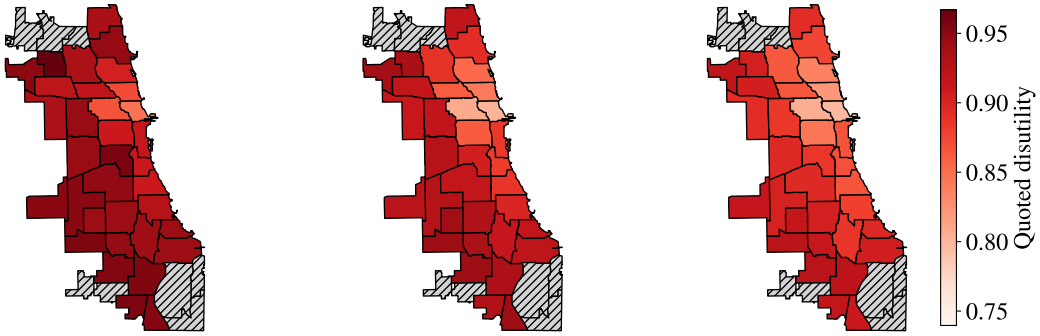
tour rate). In Figure 2.11b, we see that one zone near downtown with dense demand quotes slightly higher prices and induces slightly higher payments under the match-based policy than under the static policy. Meanwhile, in the outer zones with lower demand density, match-based pricing has lower quoted prices and payments. But overall, as shown in Figure 2.11e, the realized disutility per converted request tends to be lower under the match-based policy across almost the entire region, thanks to the higher-quality matching with reduced detours. These observations are surprisingly consistent with the insights derived from the case of a single origin-destination pair, demonstrated in Proposition 2 and Proposition 3. Notably, these lower quoted disutilities in the outer zones mean that match-based pricing can expand access to shared rides in low-demand zones where operations have been most challenging. Combining this expanded access with improved profitability and cost efficiency, match-based pricing can be a win-win for riders and platform alike.

2.5 Conclusion and Discussion

This work highlights the challenges that platforms face in operating shared rides as well as a path forward. We show that by taking a heuristic, iterative approach to setting a static price, ridesharing platforms overprice and suppress demand for shared rides, thereby limiting the areas where they can operate to dense downtown areas. But even the optimal static price is limiting, and incurs the risk that the platform cannot realize cost savings from matching riders with similar origins and destinations. Our proposed match-based pricing policy sets prices according to matching outcomes: it charges a higher price if the request is dispatched unmatched, and offers a discount if the request is matched. We show theoretical results on a single origin-destination pair and validate these results with large-scale simulations on real data from Chicago by developing scalable and simple computational approaches. As compared to the optimal static pricing policy, our policy achieves higher match rate and cost efficiency, along with lower prices paid by the riders. These gains are especially pronounced when costs are high and demand is low, allowing shared rides to operate in areas where they have historically been most challenging.

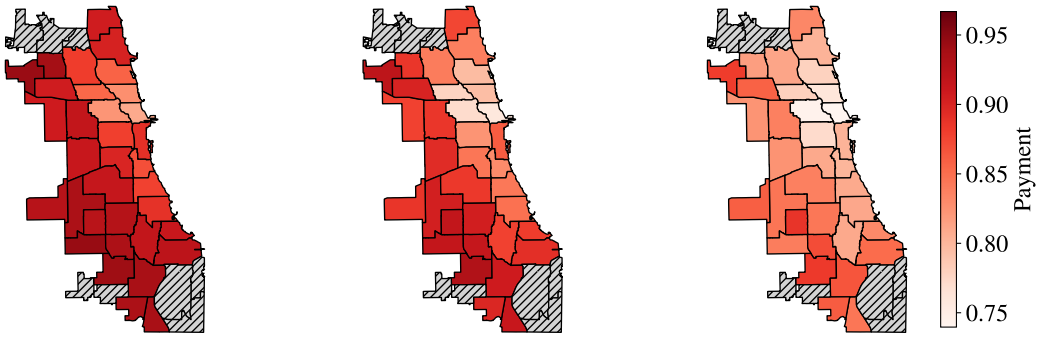


(a) Quoted price (Iterative) (b) Quoted price (Static) (c) Quoted price (Match)

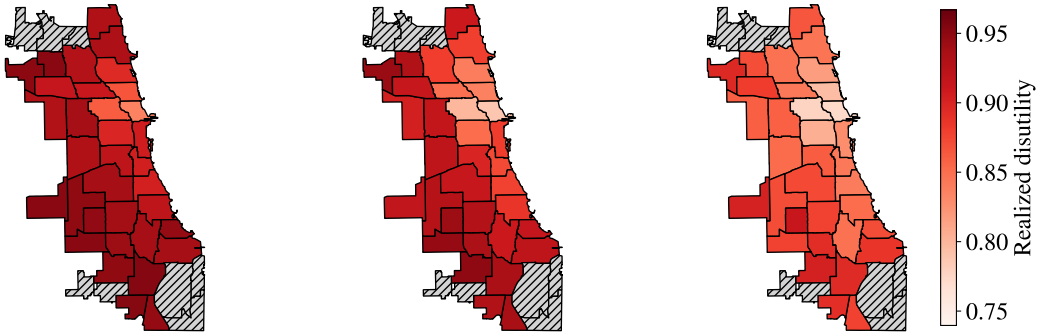


(d) Quoted disutility (Iterative) (e) Quoted disutility (Static) (f) Quoted disutility (Match)

Figure 2.8: Geographical distribution of performance metrics in Chicago ($\beta_0 = 0.05, \beta_1 = 0.675, c = 0.9, \theta = 1/3$): quoted price and quoted disutility. The shadowed zones are those with fewer than 5 rider arrivals originating from them in the test data.

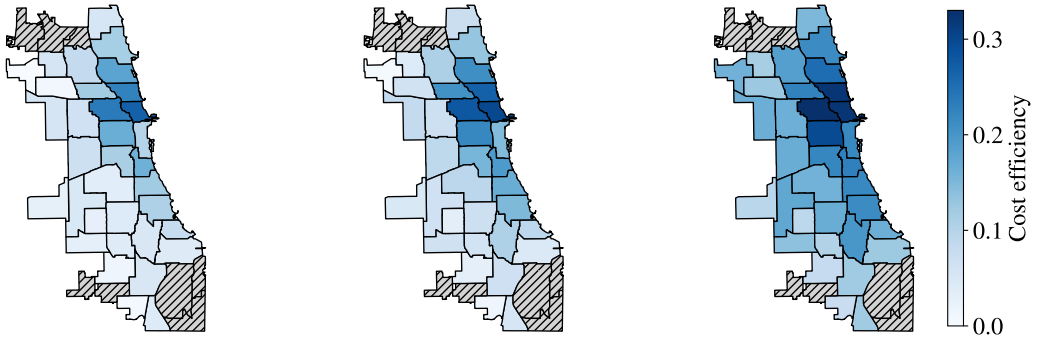


(a) Payment (Iterative) (b) Payment (Static) (c) Payment (Match)

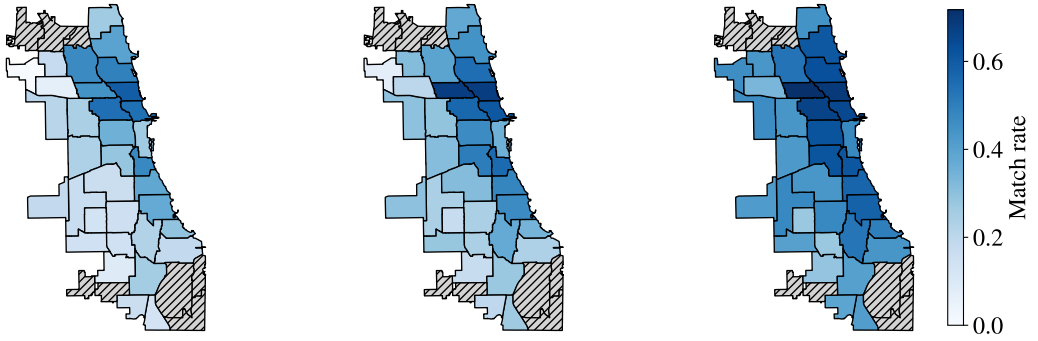


(d) Realized disutility (Iterative) (e) Realized disutility (Static) (f) Realized disutility (Match)

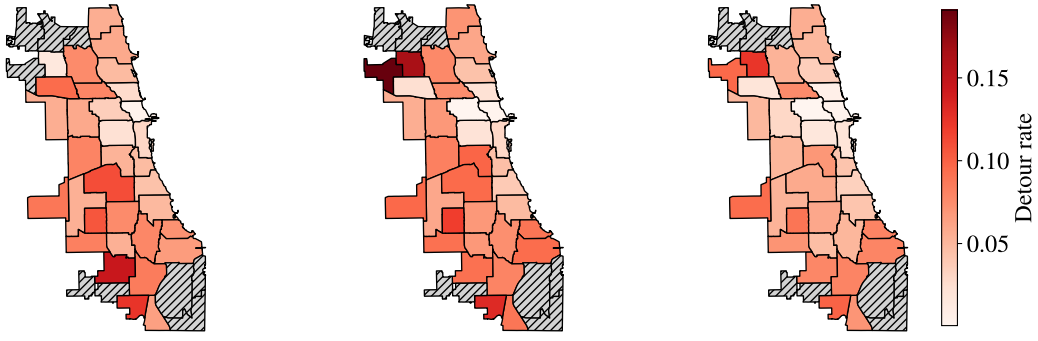
Figure 2.9: Geographical distribution of performance metrics in Chicago ($\beta_0 = 0.05, \beta_1 = 0.675, c = 0.9, \theta = 1/3$): payment and realized disutility. The shadowed zones are those with fewer than 5 rider arrivals originating from them in the test data.



(a) Cost efficiency (Iterative) (b) Cost efficiency (Static) (c) Cost efficiency (Match)

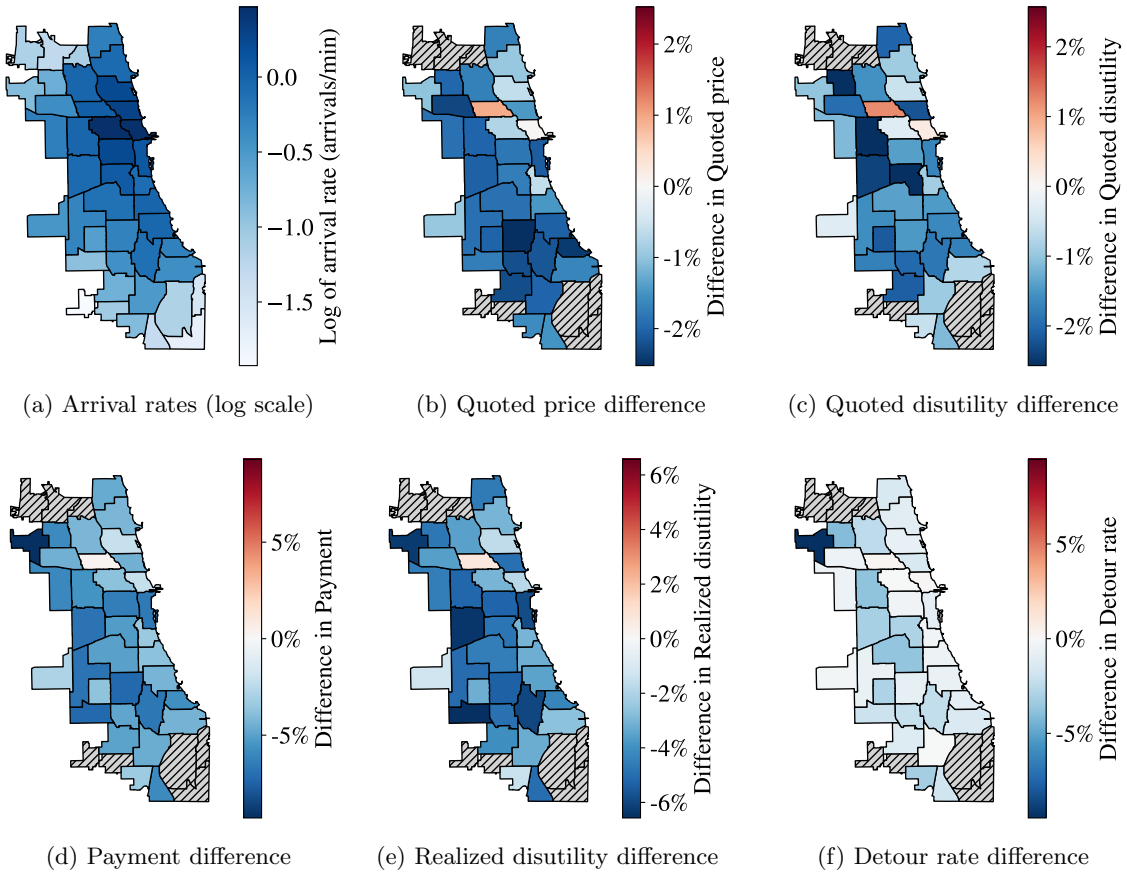


(d) Match rate (Iterative) (e) Match rate (Static) (f) Match rate (Match)



(g) Detour rate (Iterative) (h) Detour rate (Static) (i) Detour rate (Match)

Figure 2.10: Geographical distribution of cost efficiency, match rate, and detour rate in Chicago ($\beta_0 = 0.05, \beta_1 = 0.675, c = 0.9, \theta = 1/3$). The shadowed zones are those with fewer than 5 rider arrivals originating from them in the test data.



(a) Arrival rates (log scale) (b) Quoted price difference (c) Quoted disutility difference

(d) Payment difference (e) Realized disutility difference (f) Detour rate difference

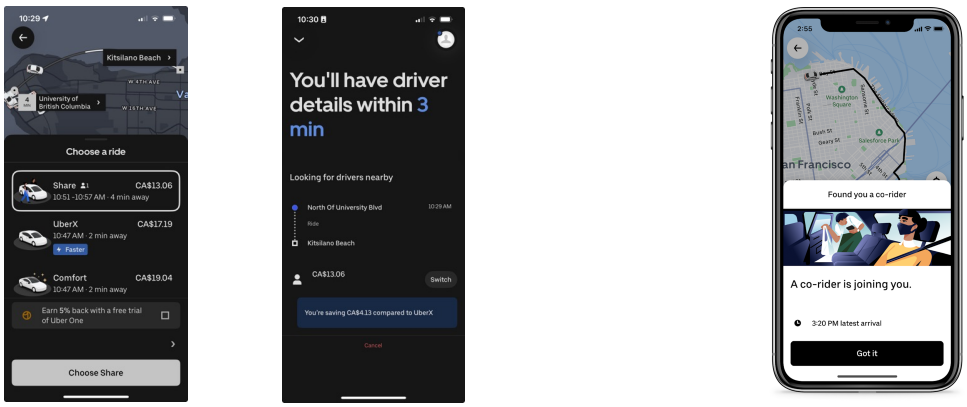
Figure 2.11: Arrival rates and relative differences in quoted price, quoted disutility, payment, realized disutility and detour rate (match-based versus static pricing) in Chicago ($\beta_0 = 0.05, \beta_1 = 0.675, c = 0.9, \theta = 1/3$).

Chapter 3

ON-TRIP MATCHING AND PRICING FOR SHARED RIDES

3.1 Introduction

Another central challenge in operating shared rides is that its success relies heavily on local demand density, which directly affects the availability of quality matches [3]. Platforms acknowledge that "matching with a co-rider is dependent on ... the number of ride requests ... available in a given area" [113], and limit their shared ride offerings accordingly. UberX Share is only available in select major cities with high demand [112], and Lyft, after discontinuing shared rides entirely in 2023, pivoted in 2025 to offering shared rides only at major airports where demand is concentrated [71]. Consequently, the benefits of shared rides have not been realized outside of dense urban centers; notably, lower-density, outlying areas also tend to coincide with transit deserts [60, 5]. An analysis of Toronto ridesharing data finds that downtown Toronto accounts for 66% of total matches with a match rate of 62%, whereas the match rate in the outlying areas drops to only 39% [120].



(a) Pre-trip (b) On-trip

Figure 3.1: Screenshots of Uber app with both pre-trip and on-trip matching.

Although platforms can attempt to increase demand density by asking riders to wait longer for a ride, the resulting inconvenience can deter riders who expect high-quality, on-demand service. Hence, a goal for platforms is to increase matches without excessively inconveniencing riders, especially in low-demand areas. To this end, the platform must capitalize on all matching opportunity—which, in practice, can arise not just before a request is dispatched on a trip (Figure 3.1a), but during the trip as well (Figure 3.1b). However, the dominant matching model in the literature, which is based on non-bipartite matching [92], neglects the latter. In particular, an underlying assumption in this model is that matching is done *pre-trip*, before riders are dispatched; then, unmatched riders are dispatched in solo rides. Matching opportunity that could arise *on-trip*, when a rider is already en-route from origin to destination, is sometimes enabled through reoptimization with updated states for partially-occupied vehicles [4]; however, this phase is usually not modeled in upstream matching decisions, despite their interconnection.

This modeling gap raises the questions of first, how to model and solve on-trip matching problems, and second, when on-trip matching can be valuable for the platform. Under both pre-trip and on-trip matching, riders arrive and depart dynamically over time; hence, a key decision is whether to commit to a match now or wait for a (possibly) better match later. This tension has additional nuance when both types of matching are considered jointly. First, the addition of on-trip matching can make pre-trip matches less likely, as the rider still has opportunity to match after the end of the pre-trip phase. Second, as the rider progresses on-trip from origin to destination, both the length of the trip time available for matching and the compatibility and value of potential matches decrease, adding urgency to the matching decision. An associated challenge in modeling on-trip matching is that evaluating the value of waiting for a match requires tracking riders over the entire duration of their trips.

In addition to investigating the relationship between pre-trip and on-trip matching, we also investigate the interaction of pricing with matching. Pricing and matching decisions are intertwined: low prices can increase demand and encourage matching, and high-quality matching can reduce dispatch costs and allow the platform to lower prices. Hence, we investigate the impact of on-trip matching on pricing, and the associated accessibility and

equity implications for riders.

In our model, riders dynamically arrive, request rides in response to prices quoted by the platform, and wait a limited time for a match to arrive (pre-trip phase). If a rider is dispatched solo, then the platform can continue to evaluate potential matches until she reaches her destination (on-trip phase). The platform seeks to maximize long-run average profit. In this setting, we make the following contributions.

1. **A joint pre-trip and on-trip matching model.** We formulate a fluid relaxation that incorporates both pre-trip and on-trip matching decisions. This model captures a key feature of on-trip matching that if a rider is dispatched solo after the pre-trip phase, she can continue to match, but her position, compatibility, and matching values with other riders evolve along the trip, which is not represented in standard matching models with fixed types. We use the solution to the fluid relaxation to implement a dual-based matching policy. Moreover, we incorporate our matching model into an outer pricing optimization problem.

2. **When to match on-trip?** In our computational results, we demonstrate the contrasting behavior of pre-trip and on-trip matching. Pre-trip matching is primarily about matching value, seeking "ideal" matches with identical origin-destination pairs, hence is favorable when demand is concentrated and riders are highly compatible with each other. In contrast, on-trip matching is favorable when demand is heterogeneous and spatially dispersed, as happens in greater metropolitan areas.

3. **Evaluation on real-world data.** We evaluate our matching and pricing policies with large-scale simulations on Chicago ridesharing data. We show that adding on-trip matching improves profits and efficiency for the platform, and lowers prices for riders. The gains are especially pronounced in the outskirts of the city, where demand is less dense and more spatially dispersed. These areas also tend to have lower median incomes and less access to transit—hence, the lower prices meaningfully impact access to shared rides in historically underserved areas.

The remainder of the chapter is outlined as follows. We review the related literature in Section 3.2. Section 3.3 describes the models, and Section 3.4 reports computational results on both synthetic and real-world instances. We offer concluding remarks in Section 3.5.

3.2 Literature Review

Shared rides face a key operational challenge of deciding which riders should be pooled together in the same vehicle. The dominant modeling approach involves dynamic, stochastic matching on a non-bipartite graph: the nodes represent riders, the edges have weight corresponding to the cost (or reward) of dispatching the associated riders together, and a minimum-cost (or maximum-reward) matching determines the dispatches of riders. Riders arrive and depart[1] dynamically over time, and the key decision involves whether to commit to a match now or wait for a (possibly) better match later.

A common approach to solving such problems involves *greedily* matching riders upon arrival. In a setting where riders depart with some probability after each period, [36] find that a greedy algorithm is asymptotically optimal for reward maximization in a large-market setting, if riders are not too impatient. An alternative approach involves *batching* rider requests over short windows of time (e.g., 3 minutes in Figure 3.1a), and then solving the non-bipartite matching problem on the batch [92]. [36] additionally find that batching, like greedy, is asymptotically optimal if rider impatience is not too high; however, batching requires more patience from riders than greedy. In a setting where sojourn times are uniform and constant, [88] and [10] find that batching (with randomization to prevent the batch window from splitting up compatible matches) attains a constant-factor guarantee. However, in the setting where rider sojourn times follow an exponential distribution, [26] and [9] highlight the importance of making matching decisions at different timescales for riders of different types, even when departure rates are uniform. Hence, [9] find that batching can perform arbitrarily poorly for cost minimization, and develop a greedy matching policy that attains a constant-factor guarantee.

In this literature, the key decision of whether to commit or wait highlights the importance

[1]Departure is also sometimes called *abandonment* in the literature. The time between arrival and departure is called the *sojourn time*, and riders' limited sojourn time is said to reflect their *impatience*.

of market thickness in enabling effective matches. Accordingly, a related upstream problem involves measures to coalesce rider demand and thicken the market. To this end, riders can be asked to wait longer [119], walk to designated meeting points [99, 41, 123], or accept longer detours [99]. However, such asks can also deter riders from taking shared rides because of the associated inconvenience [27, 68]. Alternatively, pricing can encourage usage of shared rides over solo rides [53, 58, 42, 114, 101] or to convert riders who can make high-quality matches [118].

Although the non-bipartite matching formulation has been the focus of the shared rides matching literature, the cost structure has an inherent, underlying restriction that all matching happens *pre-trip*. Namely, if a request is not matched, then it is dispatched on a solo trip, and remains solo thereafter. In practice, however, requests can continue to be matched while *on-trip*. When requests can be matched on-trip, a routing problem also arises, as the shortest route may not be the one that maximizes the probability of finding a subsequent match [126]. The continued matching ability is important for high-capacity settings in which more than two riders can be pooled together [4, 97]—or even low-capacity settings when demand is sparse, as we discuss in Section 3.4. Previous approaches usually consider simplified, static problems where on-trip decisions are made through reoptimization [4], insertion heuristics [97], or greedy assignment [114]. As such, the key decision of whether to commit or wait is not explicitly considered.

We uses a dual-based matching policy that, like earlier works in dynamic stochastic matching, considers the value of a current match versus the value of waiting for a future match. This chapter also closely relates to Chapter 2, which analyzes a pricing policy that offers a discount ex-post to encourage high-quality matches. We similarly consider the interaction of pricing and matching policies, and our dual-based matching policy also bears resemblance to their multiple-origin-destination matching policy. However, whereas Chapter 2 focuses on (1) a single origin-destination setting where matching decisions are trivial, and (2) a multiple-origin-destination setting with pre-trip matching only, we explicitly model the on-trip matching phase with multiple origin-destinations. A challenge with modeling on-trip matching is that dispatched requests and their associated matching compatibilities and values evolve over the course of their trips. Consequently, previous work on pre-trip

matching with fixed edge weights cannot be applied directly. We additionally discuss the impact of on-trip matching on both the pre-trip matching phase and the associated pricing policy, and highlight when each phase of matching is most useful.

3.3 Modeling

In this section, we describe our model. Although the full framework encompasses a *combined matching model* (allowing both pre-trip and on-trip matching), we focus our analysis in the main text on the pure on-trip matching model for ease of exposition. The combined matching model is detailed in Appendix B.1.

We consider a discrete-time, infinite-horizon setting where the time is indexed by $t \in \mathbb{Z}_{\geq 0} := \{0, 1, 2, \cdots\}$. There are N distinct rider types, indexed by $i \in [N] := \{1, \ldots, N\}$, where each type corresponds to a unique origin-destination pair (O_i, D_i). The system is operated with a sufficiently large number of vehicles to serve all rider requests. All vehicles travel at a constant speed, and we define one unit of distance as the distance traveled during one unit of time. The platform incurs an operational cost proportional to the distance traveled by vehicles when serving riders. The unit cost is denoted by c per unit of distance. The platform can match two riders into the same vehicle to share a trip and thereby minimize the total operational cost.

The time horizon is finely discretized such that at most one new rider arrives in the system during any time period $t \in \mathbb{Z}_{\geq 0}$. For each $i \in [N]$, let $\Lambda_i \in [0, 1]$ denote the probability that a new rider of type-i arrives at a given time period. Since arrivals are mutually exclusive across types, arrivals follow a *categorical distribution*, and the probability that no riders arrive is given by $1 - \sum_{i \in [N]} \Lambda_i \geq 0$.[2] Each type-$i$ rider's willingness to pay (WTP) is drawn independently and identically from a cumulative distribution function $F_i(\cdot)$. The rider makes a request (i.e., converts) if their WTP is at least as high as the quoted price p_i. Define $\lambda_i := 1 - F_i(p_i) \in [0, 1]$ as the *conversion probability*. The probability of a type-i rider making a request at a given time period is therefore $\Lambda_i \lambda_i$. We further assume that

[2]We can further assume that the time discretization is sufficiently fine so that $\sum_{i \in [N]} \Lambda_i \ll 1$. Although this condition is not strictly required for the modeling formulation, it enables a convenient approximation of the rider arrival process by a Poisson process when calibrating arrivals in the computational experiments (Section 3.4).

$F_i(\cdot)$ is strictly increasing, such that there is always a one-to-one correspondence between λ_i and p_i. Consequently, optimizing prices is equivalent to optimizing λ_i for all $i \in [N]$. We denote by $p_i(\lambda_i)$ the price that induces conversion probability λ_i for all subsequent analysis.

When a new rider converts and is not immediately matched with another rider, they enter the system as an *on-trip solo rider*: the platform dispatches a vehicle for a solo ride, and this rider begins traveling toward their destination along the shortest path. For the on-trip solo rider, the platform can conduct on-trip matching to match this rider with another new rider, forming a shared ride. We assume that each rider may be matched with at most one other rider throughout the trip.

For the type-i rider, denote the solo trip length from O_i to D_i by ℓ_i units of distance, which equivalently represents a total travel time of ℓ_i time periods. We define the current state of an on-trip solo rider using the tuple (i, u), where $i \in [N]$ is their rider type and the state variable u is an integer satisfying $1 \leq u \leq \ell_i - 1$, denoting the number of time periods elapsed since departure (equivalently, the travel distance already covered along the shortest path). For example, $u = 1$ indicates that the rider converted at the previous period and this is the first period of the solo trip; and $u = \ell_i - 1$ indicates that the rider will reach the destination at the next time period and exit the system. For notational convenience, define $\mathcal{C} := \{(i, u)\}_{i \in [N], u \in [\ell_i - 1]}$ to be the set of all states of on-trip solo riders. If a new rider of type $i \in [N]$ is matched with an on-trip solo rider $(j, u) \in \mathcal{C}$, denote the (remaining) shared trip length by $\ell_{i,j}^u$. The shared trip starts from the current location of the on-trip solo rider (j, u), then visits O_i to pick up the second rider, and then drops off the two riders at D_i and D_j in the order that minimizes the travel distance (either $D_i \to D_j$ or $D_j \to D_i$).

Riders' Matching Compatibility. An on-trip solo rider can only be matched with a new rider if they are compatible. Let $\mathcal{N}_{j,u}^+$ be the set of rider types compatible with a solo rider $(j, u) \in \mathcal{C}$. Generally, the platform can use various metrics to define the compatible set. In this chapter, we set two conditions for a solo rider (j, u) to be compatible with a new rider of type i:

- *Trip Length Condition*:

$$\ell^u_{i,j} < \ell_i + \ell_j - u,$$

that is, the shared trip length must be smaller than the summation of remaining solo trip lengths for i and (j, u);

- *Backtracking Condition*:

(j, u) should not be on any shortest path from O_i (exclusive) to D_j.

Otherwise, picking up the new type-i rider could later involve *backtracking*, i.e., traversing nodes that the solo rider (j, u) had already visited, creating a bad experience for the solo rider (j, u).



(a) Violation of trip length condition

(b) Violation of backtracking condition

Figure 3.2: Illustrations of incompatibility between a solo rider (j, u) and a new rider of type i.

Figure 3.2 illustrates two scenarios where a solo rider (j, u) is incompatible with a new rider of type i. In Figure 3.2a, the trip length condition is violated because the shared trip length (i.e., the distance from (j, u) to D_i) exceeds the sum of the two individual trip lengths. In Figure 3.2b, although the trip length condition is satisfied, (j, u) is now between O_i and D_j, meaning that matching (j, u) with the new type-i rider requires traveling back toward O_i first, violating the backtracking condition.

One can easily verify that matching (j, u) with a new rider of the same type j will always violate the backtracking condition; thus, the platform should never match two riders of the same type in the on-trip matching. This behavior contrasts with pre-trip matching, where same-type matches are ideal.

3.3.1 Markov Decision Process

The optimal pricing and matching problem is formulated as an average-reward, discrete-time Markov Decision Process (MDP). Let the state of the system at time $t \in \mathbb{Z}_{\geq 0}$ be $\mathbf{s}_t = (\mathrm{s}_{t,i,u})_{(i,u)\in\mathcal{C}} \in \mathcal{S}$, where $\mathrm{s}_{t,i,u} \in \mathbb{Z}_{\geq 0}$ is the number of solo riders in state $(i,u) \in \mathcal{C}$ at time t and $\mathcal{S}$ is the state space. The platform makes decisions on the pricing and matching policies:

- *Pricing Policy.* The platform implements a static (i.e., state-independent) pricing policy, by controlling the conversion probability $\lambda_i \in [0,1]$ for rider type $i \in [N]$. Denote $\boldsymbol{\lambda} := \lambda_{[N]}$, the pricing policy for all rider types.

- *Matching Policy.* The platform implements a state-dependent matching policy $\phi_i : \mathcal{S} \mapsto \mathcal{C} \cup \{0\}$. When a new rider of type $i \in [N]$ converts at time t, given the current system state $\mathbf{s}_t$, the platform either (i) immediately matches the new rider with an existing on-trip solo rider $(j,u) \in \mathcal{C}$ such that $i \in \mathcal{N}^+_{j,u}$ and $\mathrm{s}_{t,j,u} > 0$, or (ii) dispatches the new rider a solo ride following the shortest route, where we denote as action 0. Denote $\boldsymbol{\phi} := \phi_{[N]}$, the matching policy for all rider types.

Since the platform instantly dispatches all new requests and at most one new rider makes a request at each time period, we have $\mathrm{s}_{t,i,u} \in \{0,1\}$, that is, there is at most one solo rider for each rider type at each possible position at any time. Therefore, the state space $\mathcal{S}$ is finite, with $|\mathcal{S}| = 2^{|\mathcal{C}|}$. The state transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ depends on the realization of the stochastic rider arrival, conversion, and the platform's matching decision at time t. Specifically, at state $\mathbf{s}_t$, one of three mutually exclusive events happens:

- No rider makes a request (with probability $1 - \sum_{k\in[N]} \Lambda_k \lambda_k$) at time t. In this case, we have

$$\mathrm{s}_{t+1,i,u} = \begin{cases} 1, & \text{if } u \geq 2 \text{ and } \mathrm{s}_{t,i,u-1} = 1, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall (i,u) \in \mathcal{C},$$

  that is, all existing solo riders move forward. Note that riders in state $(i, \ell_i - 1)$ exit the system at time $t+1$ and are implicitly not included in $\mathbf{s}_{t+1}$.

- A type-j rider makes a request (with probability $\Lambda_j\lambda_j$) at time t and the platform's decision is to let them wait. In this case, we have the following state transition

$$\mathrm{s}_{t+1,i,u} = \begin{cases} 1, & \text{if } u = 1 \text{ and } i = j, \\ 1, & \text{if } u \geq 2 \text{ and } \mathrm{s}_{t,i,u-1} = 1, \\ 0, & \text{otherwise,} \end{cases} \quad \forall (i,u) \in \mathcal{C},$$

  that is, the new type-j rider is added into state $(j,1)$ at time $t+1$.

- A type-j rider requests (with probability $\Lambda_j\lambda_j$) at time t and the platform's action is to match them with an existing solo rider $(k,v) \in \mathcal{C}$. In this case, we have

$$\mathrm{s}_{t+1,i,u} = \begin{cases} 1, & \text{if } u \geq 2, (i,u) \neq (k,v) \text{ and } \mathrm{s}_{t,i,u-1} = 1, \\ 0, & \text{otherwise,} \end{cases} \quad \forall (i,u) \in \mathcal{C},$$

  that is, the solo rider (k,v) forms a shared ride and leaves the system.

In the following analysis, we consider stationary policies and thus replace $\mathbf{s}_t = (\mathrm{s}_{t,i,u})_{(i,u)\in\mathcal{C}}$ with $\mathbf{s} = (\mathrm{s}_{i,u})_{(i,u)\in\mathcal{C}}$ by dropping subscript t.

Under stationary pricing and matching policies $\boldsymbol{\lambda}, \boldsymbol{\phi}$, let $\{A_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}\}_{t\geq 0}$ denote the counting process of the number of riders who end up solo (i.e., reach the destination without being matched) up to time t. For $n \in [A_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}]$, let $i_n \in [N]$ denote the type of the n-th rider. Let $\{M_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}\}_{t\geq 0}$ denote the counting process of the number of matches, and for the n-th match, $n \in [M_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}]$, let $i_n \in [N]$ denote the type of newly converted rider and $(j_n, u_n) \in \mathcal{C}$ denote the state of the existing on-trip solo rider. The total profit up to time t is given by

$$\Pi_t^{\boldsymbol{\lambda},\boldsymbol{\phi}} = \sum_{n\in[A_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}]} [p_{i_n}(\lambda_{i_n}) - c\ell_{i_n}] + \sum_{n\in[M_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}]} \left[p_{i_n}(\lambda_{i_n}) + p_{j_n}(\lambda_{j_n}) - cu_n - c\ell_{i_n,j_n}^{u_n}\right],$$

where the first term is the total profit collected from riders that end up solo, and the second term is the total profit collected from riders that are successfully matched. Note that for the matched riders, this total distance is composed of the solo distance already covered by

(j_n, u_n), which is u_n, and the remaining shared trip distance, $\ell_{i_n,j_n}^{u_n}$.

The objective of the platform is to maximize long-run average profit, given by

$$\Pi := \sup_{\boldsymbol{\lambda},\boldsymbol{\phi}} \lim_{t\to+\infty} \frac{1}{t}\mathbb{E}[\Pi_t^{\boldsymbol{\lambda},\boldsymbol{\phi}}],$$

where the expectation is taken over all the randomness, including rider arrivals, conversions, and matching decisions.

Under any pricing and matching policy, the induced Markov chain is finite and a unichain (i.e., every recurrent state is reachable from the empty state $\mathbf{s} = \mathbf{0}$), ensuring the existence of an optimal stationary policy (see, e.g., Theorem 8.4.5 in [89]). However, due to the exponential growth of the state space size $|\mathcal{S}| = 2^{|\mathcal{C}|}$ with respect to the number of rider types N and the trip lengths $(\ell_i)_{i\in[N]}$, solving this MDP exactly is computationally intractable. Therefore, the subsequent analysis focuses on a fluid relaxation of this problem.

3.3.2 Fluid Model

We now formulate a fluid relaxation of the MDP model, which provides an upper bound on the optimal MDP profit achieved under the best static pricing and matching policy (as stated in Proposition 4). Define the flow variables $x_{i,j}^u$ for all $(j,u) \in \mathcal{C}$ and $i \in \mathcal{N}_{j,u}^+$, representing the average rate of matching new riders of type i with existing solo riders (j,u) per unit of time. Define y_i^u as the average rate at which existing solo riders $(i,u) \in \mathcal{C}$ do not get matched in this particular period and continue solo. Specifically, $y_i^{\ell_i-1}$ corresponds to the flow of type-i riders who end up solo. We also define y_i^0 as the average rate of new riders of type i that are not immediately matched and become on-trip solo riders, for all $i \in [N]$.

The profit maximization problem is formulated as a bilevel optimization:

$$g := \max_{\boldsymbol{\lambda}\in[\mathbf{0},\mathbf{1}]} \quad \sum_{i\in[N]} \Lambda_i \lambda_i p_i(\lambda_i) - C(\boldsymbol{\lambda}), \tag{3.1}$$

where the first term represents the expected revenue collected per unit time, and $C(\boldsymbol{\lambda})$ is

the inner-level cost minimization problem, given by

$(\mathsf{CB}(\boldsymbol{\lambda}))$

$$C(\boldsymbol{\lambda}) := \min_{\boldsymbol{x},\boldsymbol{y}} \quad \sum_{i\in[N]} \sum_{u=0}^{\ell_i-1} c y_i^u + \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} c\ell_{i,j}^u x_{i,j}^u \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} x_{i,j}^u + y_i^0 = \Lambda_i\lambda_i, \qquad \forall i \in [N], \tag{3.2b}$$

$$\sum_{i\in\mathcal{N}_{j,u}^+} x_{i,j}^u + y_j^u = y_j^{u-1}, \qquad \forall (j,u) \in \mathcal{C}, \tag{3.2c}$$

$$\Lambda_i\lambda_i y_j^u \geq \left(1 - \sum_{k\in\mathcal{N}_{j,u}^+} \Lambda_k\lambda_k\right) x_{i,j}^u, \qquad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+, \tag{3.2d}$$

$$x_{i,j}^u \geq 0, \qquad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+,$$

$$y_i^u \geq 0, \qquad \forall i \in [N], 0 \leq u \leq \ell_i - 1, u \in \mathbb{Z}.$$

The objective function (3.2a) represents the average total cost per unit time. The first term corresponds to the cost incurred by on-trip solo riders, while the second term captures the cost of serving matched trips. Constraints (3.2b) and (3.2c) impose steady-state flow balance conditions. Specifically, (3.2b) gives a flow balance for new requests of type $i \in [N]$ with the conversion rate of $\Lambda_i\lambda_i$—they are either matched with existing solo riders $(j,u) \in \mathcal{C}$ such that $i \in \mathcal{N}_{j,u}^+$, resulting in a flow of $x_{i,j}^u$, or dispatched solo, resulting in a flow of y_i^0. Constraint (3.2c) gives the flow balance for on-trip solo riders $(j,u) \in \mathcal{C}$—under the steady state, the inflow of on-trip solo riders entering the state (j,u) is y_j^{u-1}, and they are either matched with new requests $i \in \mathcal{N}_{j,u}^+$, resulting in a flow of $x_{i,j}^u$, or continue solo, resulting in a flow of y_j^u. Constraint (3.2d) is a set of ratio constraints that limit the match rate relative to the solo flow, ensuring that the match rate is not overestimated in the relaxation. To get a clearer view of the ratio constraints, we rearrange (3.2d) as follows:

$$\frac{y_j^u}{x_{i,j}^u} \geq \frac{1 - \sum_{k\in\mathcal{N}_{j,u}^+} \Lambda_k\lambda_k}{\Lambda_i\lambda_i}, \quad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{i,u}^+. \tag{3.3}$$

Intuitively, the term $1 - \sum_{k \in \mathcal{N}^+_{j,u}} \Lambda_k \lambda_k$ is the probability that no compatible riders convert in a given time period, which means that the rider (j, u) must continue solo and is thus associated with the solo flow y^u_j. The term $\Lambda_i \lambda_i$ is the probability that a new rider i converts and is related to the match flow $x^u_{i,j}$, since matching can only happen when a new rider i converts. Combining these two terms gives the ratio constraint (3.3).

The following proposition states that the optimal objective value of the fluid relaxation provides an upper bound on the optimal long-run average profit of the MDP model.

Proposition 4. $g \geq \Pi$.

3.3.3 Pricing and Matching

With the help of the fluid relaxation, we can avoid the curse of dimensionality and design scalable algorithms that yield effective pricing and matching solutions. Let $x^u_{i,j}(\boldsymbol{\lambda})$ and $y^u_i(\boldsymbol{\lambda})$ denote the optimal primal solutions of $\mathsf{CB}(\boldsymbol{\lambda})$, and let $\gamma_i(\boldsymbol{\lambda})$, $\xi^u_j(\boldsymbol{\lambda})$ and $\eta^u_{i,j}(\boldsymbol{\lambda})$ denote the optimal dual variables associated with constraints (3.2b), (3.2c) and (3.2d), respectively. According to the Envelope Theorem (see, e.g., Corollary 5 in [77]), when $C(\boldsymbol{\lambda})$ is differentiable at $\boldsymbol{\lambda}$, its gradient, denoted by $\nabla C(\boldsymbol{\lambda})$, can be given as

$$\nabla C(\boldsymbol{\lambda}) = \left[\Lambda_i \gamma_i(\boldsymbol{\lambda}) - \Lambda_i \sum_{\substack{(j,u) \in \mathcal{C}: \\ i \in \mathcal{N}^+_{j,u}}} \left(y^u_j \eta^u_{i,j}(\boldsymbol{\lambda}) + \sum_{k \in \mathcal{N}^+_{j,u}} x^u_{k,j} \eta^u_{k,j}(\boldsymbol{\lambda}) \right) \right]_{i \in [N]}.$$

For the case when $C(\boldsymbol{\lambda})$ is nondifferentiable at $\boldsymbol{\lambda}$, we continue to use the notation of $\nabla C(\boldsymbol{\lambda})$, which can be interpreted as the subgradient or supergradient (whenever such gradients exist) that informs the ascent direction.

The pricing optimization problem formulated in (3.1) is, in general, non-convex. Recent work by [24] demonstrates the effectiveness of the Minorization-Maximization (MM) algorithm for a similar fluid matching model, subject to both flow balance and ratio constraints. Therefore, in this chapter, we adapt the MM algorithm proposed in [24]. Let $\boldsymbol{\lambda}^{(k)}$ denote the solution estimate after k iterations. In the $(k+1)$-th iteration, we construct a surrogate

function $Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)})$ defined as follows:

$$Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)}) = \sum_{i \in \mathcal{N}} \Lambda_i \lambda_i p_i(\lambda_i) - \nabla C(\boldsymbol{\lambda}^{(k)})^\top (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(k)}). \tag{3.4}$$

In (3.4), we construct the linear function $\nabla C(\boldsymbol{\lambda}^{(k)})^\top (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(k)})$ to replace $C(\boldsymbol{\lambda})$. We then perform an iteration by letting $\boldsymbol{\lambda}^{(k+1)} \leftarrow \arg\max_{\boldsymbol{\lambda} \in [\mathbf{0},\mathbf{1}]} Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)})$. Under the standard revenue management assumption that $\lambda_i p_i(\lambda_i)$ is concave, maximizing $Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)})$ becomes a convex optimization problem and can thus be solved efficiently. We repeat this procedure until the MM iteration ceases to improve the objective value. This method is numerically robust; we find empirically that it converges to consistent conversion values starting from different random initial solutions.

After obtaining the pricing result, denoted by $\hat{\boldsymbol{\lambda}}$, we implement a heuristic dual-based matching policy to conduct the on-trip matching. Whenever a new rider of type $i \in [N]$ converts, let

$$\kappa_i(\hat{\boldsymbol{\lambda}}) := \min_{(j,u) \in \mathcal{C}: i \in \mathcal{N}^+_{j,u}, s_{j,u}=1} \{c\ell^u_{i,j} - \xi^u_j(\hat{\boldsymbol{\lambda}})\},$$

then the matching policy is given by

$$\phi_i(\mathbf{s}) \in \begin{cases} \arg\min_{(j,u) \in \mathcal{C}: i \in \mathcal{N}^+_{j,u},\, \mathrm{s}_{j,u}=1} \{c\ell^u_{i,j} - \xi^u_j(\hat{\boldsymbol{\lambda}})\}, & \text{if } \gamma_i(\hat{\boldsymbol{\lambda}}) \geq \kappa_i(\hat{\boldsymbol{\lambda}}), \\ \{0\}, & \text{otherwise.} \end{cases} \tag{3.5}$$

Intuitively speaking, the dual variable $\gamma_i(\hat{\boldsymbol{\lambda}})$, associated with (3.2b), measures the marginal cost of *not* letting a new type-i rider match immediately, and the dual variables $\xi^u_j(\hat{\boldsymbol{\lambda}})$, associated with (3.2c), measures the marginal cost of not matching the on-trip solo rider (j, u) (i.e., keeping them solo). If matching the new type-i rider with the on-trip solo rider (j, u), it results in a (generalized) cost of $c\ell^u_{i,j} - \xi^u_j(\hat{\boldsymbol{\lambda}})$. Therefore, the best matching will be the one that results in a minimum cost of $\kappa_i(\hat{\boldsymbol{\lambda}})$. If $\kappa_i(\hat{\boldsymbol{\lambda}})$ is no larger than $\gamma_i(\hat{\boldsymbol{\lambda}})$, matching is preferred; otherwise, the new rider should be dispatched solo.

Remarks on the Model without Ratio Constraints. In our fluid model, the ratio constraints in (3.2d) are introduced to prevent overestimation of match rates. The following proposition demonstrates that relaxing these ratio constraints may indeed lead to overly restrictive matching policies.

Proposition 5. *When the ratio constraints* (3.2d) *are relaxed, there exists an optimal dual solution such that* $\xi_i^u(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) - cu,\ \forall (i,u) \in \mathcal{C}$. *In this case, the matching policy* (3.5) *reduces to a rule where a new rider of type* $i \in [N]$ *is matched with an on-trip solo rider* $(j,u) \in \mathcal{C}$ *only if* $i \in \mathcal{N}_{j,u}^+$ *and* u *satisfies* $\ell_{i,j}^u + u = \ell_{i,j}^1 + 1$.

Under the matching policy given in Proposition 5, the platform will match riders i and j only if the current location of rider j (i.e., the value of u) is "optimal", in the sense that the realized total trip length $\ell_{i,j}^u + u$ must equal the minimum possible length achievable one time period after j departs, which is $\ell_{i,j}^1 + 1$. This requirement is practically unrealistic. Unless O_j lies exactly on one shortest path from O_i to D_i, the total trip length typically varies with u, making this equality condition nearly impossible to satisfy for any $u > 1$. Therefore, this proposition confirms that the ratio constraints (3.2d) play a critical role in regulating feasible match rates relative to solo rates, thereby avoiding overly optimistic (or restrictive) matching policies.

3.4 Computational Experiments

In this section, we evaluate the performance of a matching policy (and corresponding pricing policy) that combines pre-trip and on-trip matching, which reflects ridesharing dispatch in practice. Under this matching policy, each request first waits on the platform for a suitable match for up to T periods; if it remains unmatched, it is dispatched solo, and can continue to be matched on-trip. First, to understand the scenarios that favor each component of the combined policy, we compare the performance of pre-trip and on-trip matching on small synthetic examples (Section 3.4.1). Then, we present large-scale computational results for the combined policy on real-world ridesharing data in Chicago (Section 3.4.2).

The combined policy extends the pure on-trip matching model described in Section 3.3; we present its fluid formulation, along with the corresponding pricing and matching policies,

in Appendix B.1. The pure pre-trip matching policy, along with its corresponding pricing policy, is implemented using the model for the combined policy, but letting $\mathcal{N}^+_{j,u} = \emptyset$ for all $u > 0$ for all $j \in [N]$.

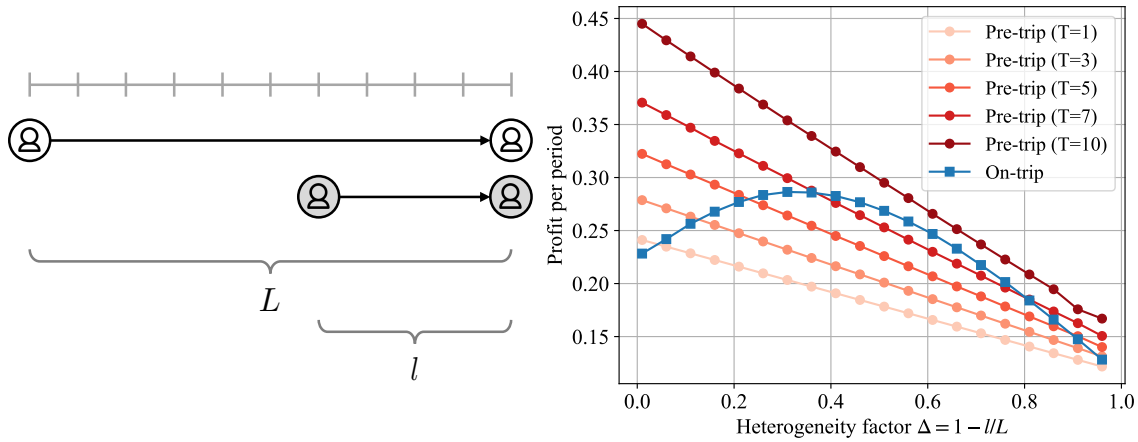
Although we usually refer to our policies as the combined, pre-trip, and on-trip matching policies for brevity, all of our experiments compute matching and pricing policies *jointly*.

3.4.1 Synthetic Instances

We first compare the profits of the pre-trip and on-trip matching policies (with pricing) in two synthetic examples. For both instances, we use cost $c = 0.7$ and total per-period arrival probability $\sum_{i=1}^{N} \Lambda_i = 0.1$, which is equally divided among all rider types. Following [17] and [118], we let riders' willingness to pay be uniformly distributed in [0, 1], and thus $\lambda_i = 1 - p_i$ for each rider type $i \in [N]$. We consider pre-trip waiting periods $T \in \{1, 3, 5, 7, 10\}$. To evaluate each policy, we simulate each policy on synthetic arrivals over 10^7 periods and compute the resulting profit.

Example 1 (Two rider types: one long and one short). The first synthetic instance uses two rider types illustrated in Figure 3.3a. The two rider types, each with the same arrival probability $\Lambda = 0.05$ per period, travel from left to right along a line toward a common destination. One rider type has a longer length of $L = 100$ periods, and the other has a shorter length of $l \in \{1, \cdots, L\}$ periods. To represent the difference between the two rider types, we define a *demand heterogeneity* parameter $\Delta = 1 - l/L \in [0, 1]$: a smaller Δ indicates that the riders are more similar, with $\Delta = 0$ corresponding to two riders with identical origin-destinations, who form an "ideal" match.

Figure 3.3b compares the profits of pre-trip matching (shades of red) and on-trip matching (blue) for varying Δ. Naturally, the pre-trip matching profit increases in T, because a longer waiting period coalesces demand more effectively and enables better matches. More importantly, the pre-trip matching profit decreases monotonically in Δ, whereas the on-trip matching profit first increases, then decreases with Δ. This contrasting behavior highlights the different strengths of the two matching policies. Pre-trip matching is primarily concerned with matching *value* (i.e., the cost saved by matching two riders together), with



(a) Two rider types.

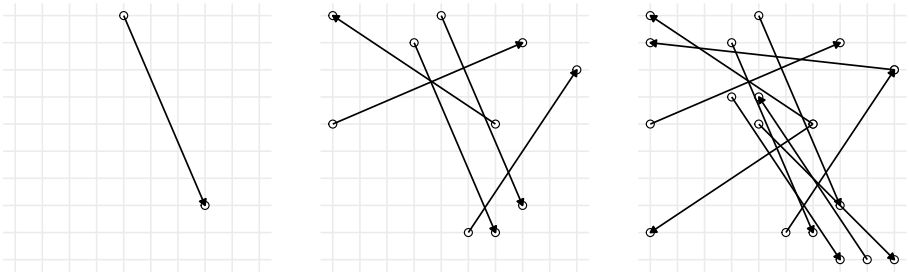
(b) Profits of pre-trip and on-trip matching.

Figure 3.3: Illustration of the two-rider-type example, and profit comparison between pre-trip and on-trip matching under varying demand heterogeneity.

the ideal scenario being $\Delta = 0$, a single origin-destination. Hence, when the demand is homogeneous (Δ is low), the matching value is high, and pre-trip matching performs well. In contrast, on-trip matching exhibits a trade-off between matching value and matching *opportunity* (i.e., the likelihood of a match occurring). When the demand is homogeneous (Δ is low), the matching value is high, but the matching opportunity is low. This is because the distance between the two riders' origins is short, and once the vehicle passes the short rider's origin, it can no longer match the riders because of the backtracking condition. Meanwhile, when the demand is heterogeneous (Δ is high), the matching opportunity is high, but the matching value is low. At a moderate level of demand heterogeneity, both matching opportunity and matching value are present, benefiting on-trip matching. Thus, it is the importance of matching opportunity, as opposed to matching value alone, that drives the differing behavior of on-trip versus pre-trip matching.

Example 2 (N rider types on a grid network). Next, we illustrate the impact of demand heterogeneity on a larger instance. Now, we have a 10×10 grid network, where each edge length is ten periods. Keeping the total per-period arrival probability fixed as

$\sum_{i=1}^{N} \Lambda_i = 0.1$, we generate N distinct rider types of equal length $L = 100$, varying N from 1 to 30, and generating five instances for each N. Figure 3.4 shows three examples of rider types generated for $N = 1, 5$, and 10. As N increases, the demand becomes more spatially dispersed, and we can examine the effect of the dispersion on pre-trip and on-trip matching.



(a) $N = 1$ (b) $N = 5$ (c) $N = 10$

Figure 3.4: Examples of the distributions of the random rider types on the grid network ($N = 1, 5, 10$; $L = 100$).

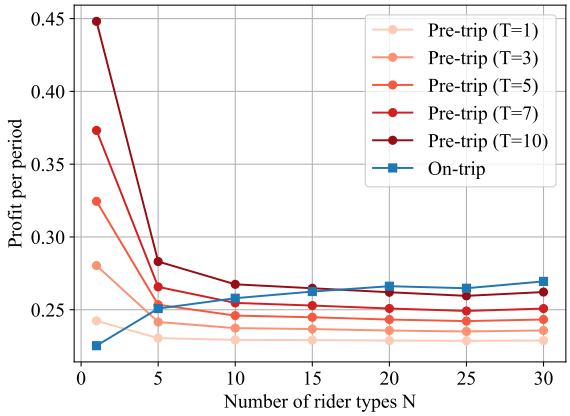


Figure 3.5: Profit comparison between on-trip and pre-trip matching policies under different numbers of rider types on the grid network.

Figure 3.5 presents the pre-trip (shades of red) and on-trip matching (blue) profits for varying N. Pre-trip matching profit decreases markedly with N, reflecting the earlier

observation that the ideal scenario for pre-trip matching is a single origin-destination: self-matches are common when demand is highly concentrated (N is low), but become rarer as demand disperses (N increases). Meanwhile, on-trip matching profit increases with N. As demand spreads out, more on-trip matching opportunities are created, which offsets the effect of lower matching value between riders.

The favorable performance of on-trip matching when riders are spatially dispersed proves to be important in real-world settings, which, despite showing high demand in dense corridors between major points of interest, also shows substantial demand spread across greater metropolitan areas. In the next section, we apply a policy that combines pre-trip and on-trip matching (and pricing) to a real-world setting in Chicago.

3.4.2 Chicago Case Study

Setup

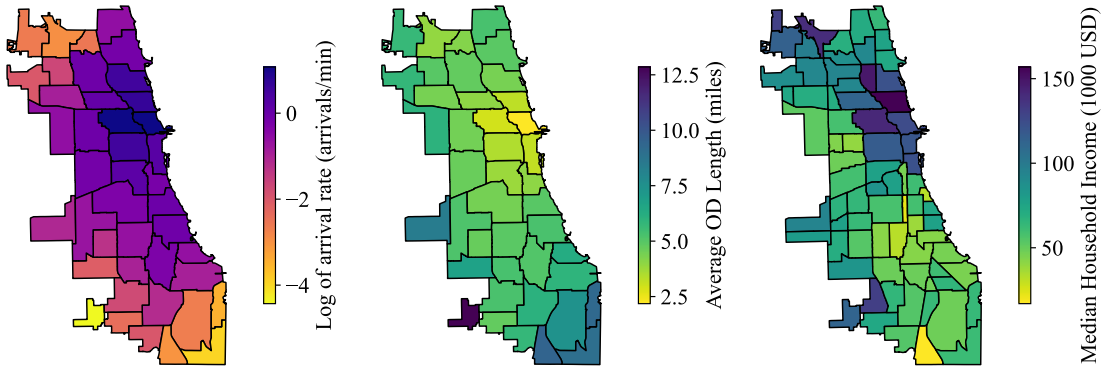
The data set provided by Chicago Data Portal[3] contains the pickup and dropoff time and location of each trip, along with an indicator of whether the rider is willing to share the ride ("authorized"). We focus on authorized trips (about 15% of all trips) recorded over eight weeks in October and November of 2019 before COVID-19. The first seven weeks serve as the training set, and the final week serves as the test set.

Rider Demand. Since the data set does not include riders who arrive without requesting a ride, we use trip data as a proxy of arrival data. We focus on the one-hour time window of 7:30 – 8:30 a.m. on Monday for each week. The total arrival rate in this time window is 28.25 riders per minute in the training set, and 29.43 riders per minute in the test set.

To create a set of request types, we follow the two-step clustering method of [9]. First, we cluster the 76 community areas of the City of Chicago[4] (excluding the O'Hare International Airport) into 42 zones. Then, we cluster the riders originating from each pickup zone by their dropoff locations using k-means clustering. We obtain in total 259 rider types. Riders

[3]Source: `https://tinyurl.com/4dcn9d5d`, accessed 2023-06-30.

[4]Source: `https://tinyurl.com/4yxcvfaj`, accessed 2023-06-30.



(a) Log of Arrival Rates (b) Trip Lengths (c) Household Incomes

Figure 3.6: Geographical distribution of arrival rates, trip lengths, and household incomes in Chicago neighborhoods.

in the test set are mapped to the closest of the 259 types from the training set.

The geographic distributions of arrival rate, trip length, and household income are shown in Figure 3.6.[5] The downtown zones (center-right) have the highest demand, the shortest average trip lengths, and the highest household incomes; the zones in the outskirts have the lowest demand, the longest average trip lengths, and the lowest household incomes. Moreover, the zones in the outskirts also tend to have the lowest levels of transit accessibility [37].

Road Network. Recall that in the discrete-time setting described in Section 3.3, time periods are short enough that there is at most one arrival per period, with the probability of an arrival $\sum_{i\in[N]} \Lambda_i \ll 1$. Accordingly, we set the period length to be $\delta = {}^{0.1}/_{\sum_{i\in[N]} \Lambda_i} = {}^{0.1}/_{28.25 \text{ riders}\cdot\text{min}^{-1}} = {}^{0.1}/_{0.47 \text{ riders}\cdot\text{s}^{-1}} = 0.21$ s. Recall further that vehicles travel one unit of distance per period. Then, at a speed of 24 mph ≈ 10.73 m/s [105], a vehicle travels a distance of (10.73 m/s)(0.21 s) ≈ 2.28 m. We therefore discretize the Chicago road network (obtained using the `OSMnx` package, [19]) into segments of length 2.28 m, resulting in a road

[5]Figures 3.6a and 3.6b use the clustered zones described in the previous paragraph. Figure 3.6c uses the original community areas in the data from `https://metop.io/insights/agqp`.

network with 4,254,632 nodes and 8,606,506 edges.

Payments and Costs. We assume riders' willingness-to-pay is uniformly distributed between 0 and 1 in dollars per mile, so that $\lambda(p) = 1-p$. We consider costs $c \in \{0.7, 0.9, 1.1\}$ in dollars per mile.[6]

Policies. We focus on the combined matching policy, and use pre-trip matching as a benchmark. We consider waiting window lengths of $\{0.0, 0.5, 1.0, 1.5, 2.0\}$ minutes, which corresponds to $T \in \{0, 141, 282, 423, 564\}$ periods. All policies are trained on the training data and evaluated in simulation on the test data over 100 repetitions.

Results

Overall Performance. The performance metrics of the combined and pre-trip matching policies in the test data and training data are presented in Table 3.1 and Table B.1 (Appendix B.3.1), respectively. Performance metrics include the profit rate per minute, the average quoted price (per rider, per unit distance), the average payment (per converted request, per unit distance), the throughput (requests per minute), the match rate (number of matched requests divided by total requests), the on-trip match portion (the fraction of matches made when one of the requests had originally been dispatched solo), the cost efficiency (one minus total dispatching cost divided by a hypothetical cost if all requests were dispatched solo),[7] and the average detour rate (over matched requests, and defined as the total detour distance divided by a hypothetical total distance of the matched requests assuming all were dispatched solo).

The results show that combined matching improves profit substantially across all instances compared with pre-trip matching, especially when the waiting window T is relatively small and the cost c is relatively high. Under these settings, as reflected in the on-trip match

[6]This is representative of Uber driver per-distance rate in regions in the US, see `https://rideshareguru.com/uber-driver-pay-rates-by-city-per-mile-and-per-minute/`, accessed 2023-06-30.

[7]Cost efficiency can be interpreted as the percentage improvement in cost for shared rides relative to solo rides. Under two-party matching, cost efficiency ranges from zero (pure solo rides) to 0.5 (ideal shared rides, where every rider is dispatched in a shared ride with another rider sharing the exact origin and destination).

Cost ($/mile)		$c = 0.7$					$c = 0.9$					$c = 1.1$				
Waiting Window (min)		0	0.5	1	1.5	2	0	0.5	1	1.5	2	0	0.5	1	1.5	2
Profit ($/min)	Pre-trip	2.734	4.079	4.928	5.496	5.872	0.302	0.762	1.297	1.750	2.079	0.000	0.000	0.000	0.000	0.015
	Combined	4.750	5.301	5.755	6.162	6.481	1.343	1.690	2.020	2.330	2.576	0.000	0.042	0.163	0.251	0.404
	Difference	73.8%	30.0%	16.8%	12.1%	10.4%	344.1%	121.8%	55.7%	33.1%	23.9%	-	∞	∞	∞	2555.7%
Quoted Price ($/mile)	Pre-trip	0.850	0.798	0.777	0.765	0.756	0.950	0.902	0.873	0.857	0.846	1.000	1.000	1.000	1.000	0.988
	Combined	0.800	0.781	0.767	0.758	0.750	0.892	0.873	0.856	0.845	0.836	1.000	0.984	0.962	0.949	0.934
	Difference	-5.9%	-2.1%	-1.2%	-0.9%	-0.8%	-6.1%	-3.3%	-1.9%	-1.4%	-1.2%	0.0%	-1.6%	-3.8%	-5.1%	-5.4%
Payment ($/mile)	Pre-trip	0.850	0.791	0.767	0.754	0.744	0.950	0.887	0.850	0.830	0.812	-	-	-	-	0.773
	Combined	0.792	0.772	0.757	0.747	0.739	0.871	0.850	0.830	0.815	0.805	-	0.880	0.876	0.850	0.833
	Difference	-6.9%	-2.3%	-1.2%	-0.9%	-0.7%	-8.3%	-4.2%	-1.8%	-1.1%	-0.9%	-	-	-	-	7.7%
Throughput (requests/min)	Pre-trip	4.387	5.920	6.534	6.893	7.157	1.459	2.848	3.700	4.198	4.522	0.000	0.000	0.000	0.000	0.358
	Combined	5.867	6.423	6.811	7.095	7.329	3.152	3.735	4.213	4.557	4.817	0.000	0.481	1.101	1.500	1.920
	Difference	33.7%	8.5%	4.2%	2.9%	2.4%	116.1%	31.1%	13.9%	8.6%	6.5%	-	∞	∞	∞	436.9%
Match Rate	Pre-trip	0.000	0.392	0.527	0.591	0.624	0.000	0.343	0.506	0.588	0.637	-	-	-	-	0.644
	Combined	0.566	0.647	0.700	0.735	0.760	0.570	0.653	0.710	0.746	0.771	-	0.741	0.778	0.829	0.860
On-trip Match Portion	Combined	1.000	0.660	0.448	0.317	0.237	1.000	0.719	0.483	0.343	0.255	-	0.810	0.598	0.414	0.287
Cost Efficiency	Pre-trip	0.000	0.105	0.160	0.193	0.214	0.000	0.082	0.147	0.187	0.212	-	-	-	-	0.322
	Combined	0.149	0.181	0.206	0.228	0.245	0.148	0.177	0.203	0.225	0.241	-	0.223	0.233	0.253	0.276
Average Detour Rate (matched requests)	Pre-trip	-	0.097	0.084	0.075	0.066	-	0.103	0.085	0.073	0.068	-	-	-	-	0.000
	Combined	0.100	0.096	0.091	0.085	0.080	0.099	0.095	0.091	0.084	0.079	-	0.055	0.072	0.076	0.073

Table 3.1: Test results on Chicago data.

portion, the pre-trip matching phase is increasingly suppressed by low waiting windows and throughput (the latter exacerbating the effect of the former), making the addition of on-trip matching more important.

Note that when $T = 0$, pre-trip matching reduces to pure solo rides (i.e., all requests are dispatched solo immediately upon arrival, and remain solo), and combined matching reduces to pure on-trip matching (i.e., all requests are either matched to pre-existing on-trip riders or dispatched solo immediately upon arrival, and riders dispatched solo can continue to be matched while on-trip). In this setting and under pre-trip matching, all riders are quoted the price $p = (1 + c)/2$ (in dollars per mile), which follows from solving the simple profit maximization problem $\max_{p\in[0,1]} \lambda(p)(p-c)$, with $\lambda(p) = 1-p$ under our willingness-to-pay model.

Therefore, when $c \geq 1$, all service under pre-trip matching is completely shut down, as the cost can never be recouped with only solo rides. Although combined matching (or pure on-trip matching) is also unprofitable when $T = 0$ and $c = 1.1$, there exist longer waiting windows (0.5 to 1.5 minutes) for which combined matching is profitable while pre-trip matching remains unprofitable. In fact, as reflected in the on-trip match portion (< 1) in these settings, adding on-trip matching can also add some pre-trip matching, despite pre-trip matching being unprofitable alone. This is because on-trip matching serves as a

backup option after initial solo dispatch. Therefore, the platform is willing to accept more requests, which brings in opportunities to match requests pre-trip.

The profit gains under combined matching are largely driven by improvements in match rate and cost efficiency, and despite the substantial increase in match rates, the detour rates remain relatively modest. There is only one exception to the improvements in cost efficiency, which arises when $c = 1.1$ and $T = 2$ minutes. In this setting, pure pre-trip matching shows higher cost efficiency, despite a lower match rate. This is because the pricing policy under pre-trip matching essentially shuts down all but a small number of closely-located downtown riders that can effectively self-match.

The improvements in cost efficiency allow the platform to lower the prices quoted to riders and the resulting payments, thereby increasing throughput and improving overall accessibility. The lower prices reflect the importance of making matching and pricing decisions *jointly*, in contrast with previous literature which studies these components separately. Improving matching (via the addition of on-trip matching) then promotes a virtuous cycle: the platform can afford to lower prices, which then increases throughput and makes it even easier for the platform to find higher-quality matches. The combination of higher profit and lower prices make the combined pre-trip and on-trip matching policy a win-win for platform and riders alike.

These wins can also be contextualized in terms of waiting time. To this end, Figure 3.7 compares the profits of combined and pre-trip matching under different waiting periods, focusing on cost $c = 0.7$. Blue means that the profit of combined matching is higher, and red means that the profit of pre-trip matching is higher. Naturally, combined matching improves when its own waiting window increases relative to that for pre-trip matching. But more importantly, the paler cells on the chart show the waiting windows required for pre-trip and combined matching to attain the same profit. Overall, combined matching can decrease the waiting window by roughly 0.5 to 1 minute and still attain a comparable profit as pre-trip matching. For example, from the profit numbers in Table 3.1, pre-trip matching needs a 2-minute waiting window to attain a profit of \$5.872/min, while combined matching needs only a 1-minute window to attain a comparable profit of \$5.755 per minute (a gap of only 2.0%). This reduced waiting window highlights another benefit of combined matching

for riders: rather than absorbing the gains in profit, the platform can redistribute these gains back to riders in the form of substantially shorter pre-trip waiting windows.

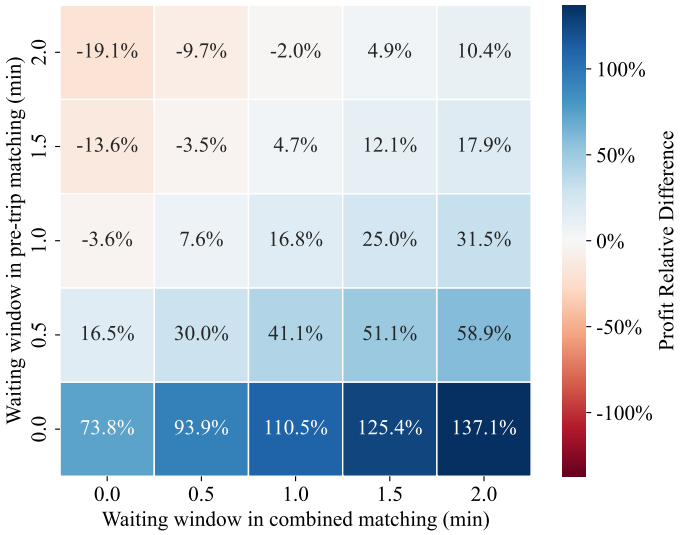
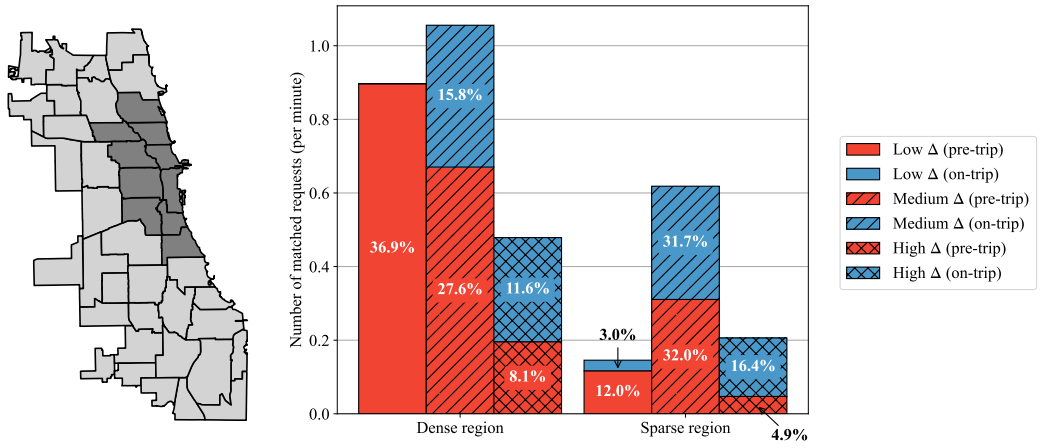


Figure 3.7: Profit comparison between combined and pre-trip matching policies under different waiting periods ($c = 0.7$).

How and Where to Match. To understand how the combined policy achieves its performance, we investigate where pre-trip and on-trip matching are favored within the combined policy. To this end, we split the 42 zones of Chicago into two regions (Figure 3.8a): the dense region (dark gray) consists of 10 zones with the highest demand density (0.23 riders per minute per square mile), while the sparse region (light gray) consists of the remaining 32 zones (0.04 riders per minute per square mile).

Following the heterogeneity factor Δ introduced in Example 1 (Section 3.4.1), we define Δ for a match as one minus the ratio of overlap distance to total shared ride distance. As before, a smaller Δ corresponds to a match that is closer to the ideal match between two riders of identical origin-destinations. Figure 3.8b compares the number of matched requests per minute between dense and sparse regions, and further breaks down the matches by pre-trip (red) versus on-trip (blue), and by low (solid), medium (hatched), and high

(crosshatched) heterogeneity levels.[8] The results show a clear contrast in matching tendency between dense and sparse regions. In the dense region where demand is concentrated, most matches are low-heterogeneity pre-trip matches. In contrast, in the sparse region where demand is spatially dispersed, medium-to-high-heterogeneity on-trip matches become more common. This trend corroborates both synthetic examples (Section 3.4.1), in which on-trip matching becomes favorable as demand spatially disperses with a moderate level of heterogeneity—now with real-world data.



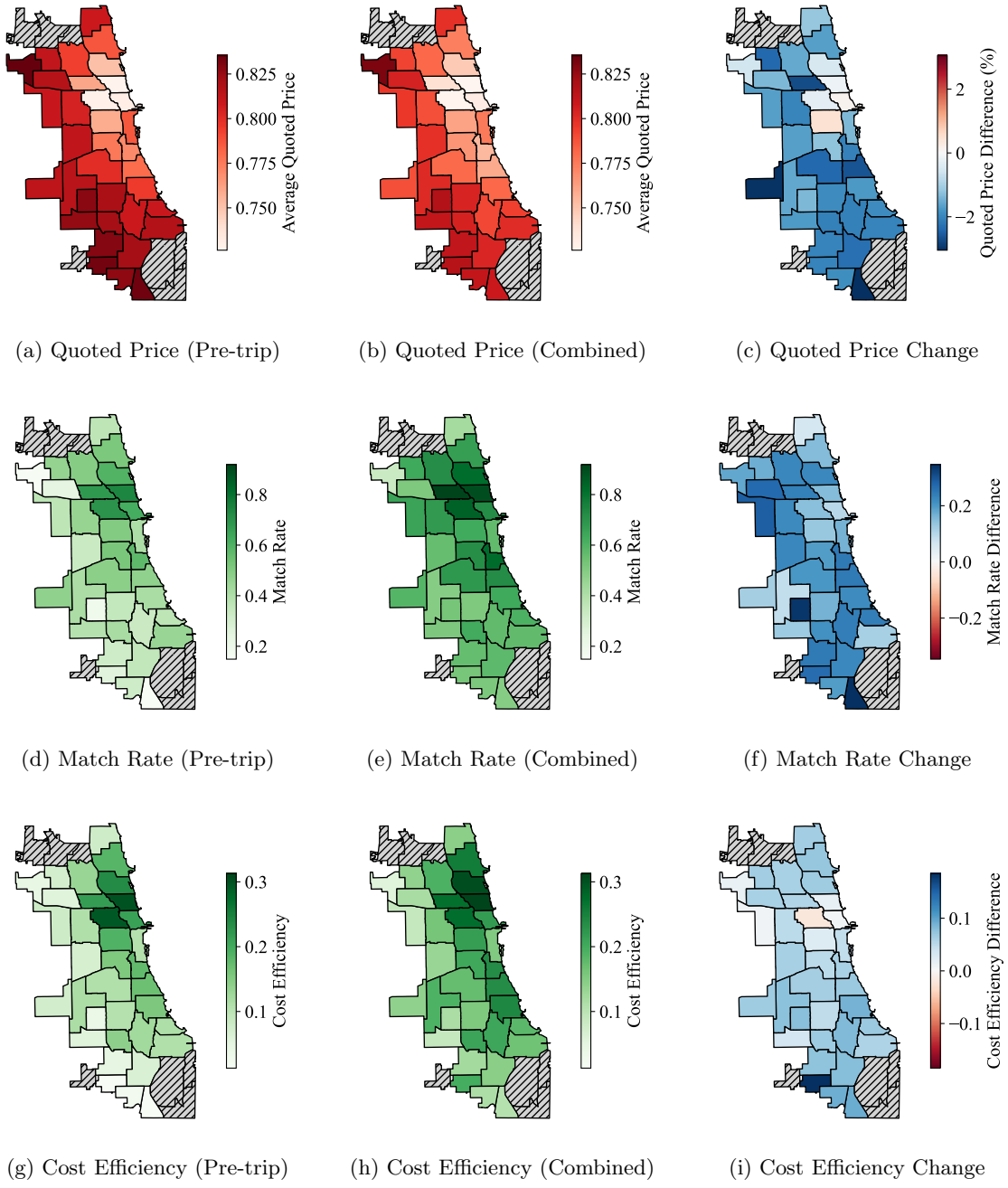
(a) Region split

(b) Matching outcome distribution

Figure 3.8: Region split in Chicago and matching outcome distribution of combined matching when $c = 0.9$, waiting period is 1.5 min.

Equity Implications. To further break down the impact of each matching policy across the different zones of Chicago, we visualize the geographical distribution of average quoted price, cost efficiency, and match rate on the test set under the two matching policies in Figure 3.9, focusing on $c = 0.7$ and a waiting window of 1 minute. Each row in Figure 3.9 contains three subfigures for a particular metric: the distributions under the two matching policies, and the comparison (relative difference for quoted price, and absolute difference for match rate and cost efficiency).

[8] We define a match with $\Delta < 1/3$ as low heterogeneity, $1/3 \leq \Delta < 2/3$ as medium heterogeneity, and $\Delta \geq 2/3$ as high heterogeneity.



(a) Quoted Price (Pre-trip) (b) Quoted Price (Combined) (c) Quoted Price Change

(d) Match Rate (Pre-trip) (e) Match Rate (Combined) (f) Match Rate Change

(g) Cost Efficiency (Pre-trip) (h) Cost Efficiency (Combined) (i) Cost Efficiency Change

Figure 3.9: Geographical distribution of performance metrics in Chicago ($c = 0.7$, waiting window of 1 minute). The shadowed zones are those with fewer than five rider arrivals originating from them in the test data.

Under pre-trip matching, the outskirts have higher quoted prices (Figure 3.9a), lower cost efficiency (Figure 3.9g), and lower match rate (Figure 3.9d) because of the lower demand density: in fact, match rates are low and cost efficiency is close to zero, showing that the platform effectively offers solo rides service in these zones. Moreover, the high prices deter access to shared rides.

Accordingly, the outskirts are where the combined matching policy makes the greatest impact: quoted prices are lower (Figure 3.9c), driven by higher match rate (Figure 3.9f) and cost efficiency (Figure 3.9i). The improvement in match rate and cost efficiency comes from the new matching opportunities created by on-trip matching. Whereas outskirts riders have difficulty getting matched pre-trip because of the sparsity of the demand, they can pick up later arriving riders from denser zones along the route. Because these riders are also associated with longer trips (Figure 3.6b), the matching opportunity can be substantial. More effective matching drives a decrease in quoted prices, expanding shared rides access to zones where such service has historically been the most challenging to operate. The effect of expanded shared rides access in these sparse zones is substantial: despite their low density per square mile, they comprise 42% of the overall demand. Moreover, these zones generally correspond to the lowest median household incomes (Figure 3.6c) and levels of transit accessibility [37].

We note that although quoted prices decrease overall, downtown zones can see slightly higher quoted prices (Figure 3.9c). This effect can arise from competition between downtown-downtown matches and outskirts-downtown matches under intermediate waiting windows. In such settings, the waiting window is long enough for downtown riders, but too short for outskirts riders, to match effectively pre-trip. Under pre-trip matching, the platform can rely on downtown-downtown matches, and therefore prefers to concentrate more downtown riders by lowering the downtown price. In contrast, combined matching enables more outskirts-downtown matches: an outskirts rider who initially is dispatched solo can likely pick up a downtown rider along the way, as the downtown is dense and the outskirts riders have long trips (Figure 3.6). Hence, competition is created between downtown-downtown and outskirts-downtown matches, and the combined policy pushes downtown prices slightly higher, suppressing pre-trip matches and shifting downtown riders toward on-trip matches

with outskirts riders. For more detail, see Appendix B.3.2.

Overall, we find that the addition of on-trip matching enables higher profits and efficiency for the platform, allowing it to either lower overall prices for riders or redistribute the gains in the form of lower waiting times in the pre-trip phase. The greatest benefits are seen in the sparse outskirts zones, expanding access to shared rides to areas that have historically been underserved.

3.5 Conclusion and Discussion

Pre-trip matching decides matches before riders are dispatched; hence, a rider who is dispatched solo remains solo for the duration of her trip. By contrast, on-trip matching allows the platform to make matches even after a rider is initially dispatched solo. In the on-trip setting, a key feature is that the rider must be tracked over the course of her entire trip, as her matching opportunity and value change with her position. In this work, we develop joint pre-trip and on-trip matching policies and investigate the conditions that favor each type of matching. We then embed our matching model into an outer pricing optimization problem to understand the interaction of pricing and matching decisions.

Through computational results on both synthetic and real-world data, we find that pre-trip matching is favorable in dense downtown areas, which approach the ideal single-origin-destination setting. Meanwhile, on-trip matching is favorable in sparse outskirts where riders are spatially dispersed, which enables the platform to manage a trade-off between matching opportunity and matching value. Adding on-trip matching increases profit and efficiency, allowing the platform to lower prices for riders and expand access to shared rides, especially in the sparse outskirts. Thus, successful operation of shared rides in large metropolitan areas, where a substantial portion of the demand can be spatially dispersed, requires both pre-trip *and* on-trip matching, emphasizing the importance of modeling this new matching paradigm.

Chapter 4

CALYBER: A RIDESHARING GAME

4.1 Introduction

A ridesharing platform is a transportation service provider that connects customers (*riders*) with drivers who can take them from their origins to their desired destinations, usually via a mobile app. Their products usually include both *solo* and *shared* rides; the former is similar to a taxi service, whereas the latter pools multiple riders with compatible origin-destinations in the same vehicle. The shared rides products, called UberX Share (formerly UberPool) and Lyft Shared (formerly Lyft Line), were launched around 2014 to improve vehicle utilization, lower costs, and mitigate congestion by matching riders with similar routes. These products initially gained significant popularity in urban areas and at their peak accounted for a substantial share of trips (20%–30%) on major platforms [70, 72].

However, sustaining shared ride products has proven challenging. Platforms must navigate issues such as low match rates outside dense city centers, balancing affordable prices with operational costs, and maintaining customer satisfaction with respect to detours and waiting times. The onset of the COVID-19 pandemic disrupted shared rides further, and even after relaunch efforts [73, 110], adoption has remained inconsistent. Companies experimented with various pricing models and incentives to encourage adoption and improve matching efficiency, but these efforts often fell short of achieving the desired outcomes [33, 29]. These experiences highlight both the opportunities and the persistent challenges associated with shared ride services in practice.

This case is developed against this backdrop, drawing on the direct field experience of authors who worked at leading ridesharing companies during the formative years of shared ride product development, and motivated by the policies developed in [118]. *Calyber* is a fictitious Chicago-based ridesharing company striving to offer affordable and efficient carpool service, yet it faces significant operational hurdles. A central challenge is that low

prices are required to attract enough demand for effective matching, but such prices may not recover dispatching costs. Conversely, higher prices can suppress demand density, leading to inefficient matches and longer detours. To address these issues, CEO Eliot George and Chief Scientist Thea Bruckner propose overhauling the company's current pricing and matching engine (an unprofitable black box that had been developed by a former employee, Nick Bailey), aiming to develop a transparent, data-driven, and real-time engine that sets prices and matches riders based on cost structures, demand patterns, and rider behavior. Students are provided with training data generated from the current pricing and matching engine ("Nick's engine"). Guided by the instructor and following Thea's recommendations in the case, they use this data to develop the improved pricing and matching policies ("Thea's engine"). Finally, they experiment with and develop their own pricing and matching policies.

This case invites students to explore the interplay between pricing, matching, and experimentation in the design of an effective shared rides product. It offers a hands-on and engaging experience in developing and deploying data-driven, real-time analytics solutions for a ridesharing marketplace. At its core, the case centers on stochastic dynamic decision-making, where each real-time decision should account for both its immediate reward and its future implications—a methodological concept that is often challenging to convey through traditional instruction. By situating this concept within a modern and relevant context, the case provides an immersive, data-driven tool for teaching dynamic decision-making in practice.

The operational context is as follows. Riders arrive randomly over time. For each arriving rider, the platform sets a price, which the rider may accept or reject. If accepted, the platform attempts to match the rider with another waiting rider, enabling them to share a ride to their destinations. The platform earns revenue from the prices paid by both riders while incurring driving costs for their pickup and drop-off. If no suitable match is immediately available, the platform can defer matching; however, riders have limited waiting patience, and if this threshold is exceeded, the platform must dispatch the rider solo, resulting in higher costs and lower efficiency.

In this simulation game, students play the role of the platform. Their objective is to maximize profit by setting prices for incoming riders and matching those who accept (referred

to as "converted" riders or requests). Total profit is calculated as the revenue collected from all riders minus the total driving costs. The challenge lies in navigating key trade-offs: Pricing decisions must account for both the cost of serving each rider and the rider's willingness to pay—cost assessment is nontrivial, as it depends on future, uncertain matching outcomes. For matching, students must decide whether to match riders immediately or wait for potentially better matches, all while managing the risk that riders' patience will expire, forcing costly solo rides.

The case activities span several weeks, during which students are provided with a historical ridesharing dataset[1] containing details on over $10,000$ rider arrivals, prices, conversion rates, matching outcomes, and more. They also receive a code template specifying the required input and output formats, enabling them to design and implement their own pricing and matching policies. Student policies are evaluated and ranked through simulation on both validation and test datasets. The validation set is used for optional midterm evaluation and providing feedback to students, whereas the test set determines the final ranking. Performance is assessed primarily on the profitability of each policy, with the secondary objective being matching efficiency.

4.2 Related Work

Ridesharing and the broader transportation sharing economy provide contemporary and relatable applications to engage students. Operations strategy cases ask students to discuss the evolution of the ridesharing business model [79], as well as competition with the taxi industry [102] and rival platforms [23, 125, 51]—issues familiar to students thanks to extensive coverage by the popular media [7, 84, 14].

Another relevant topic involves the design of UberPool and related carpooling products, where the aspiration of "[turning] every car into a shared car... reducing traffic and pollution" meets the challenge of providing a positive rider experience [57]. To motivate the benefits of carpooling (among other applications), [13] use a fifteen-passenger exercise to illustrate how individual passengers reduce costs by collaborating and discusses allocation

[1]Source: `https://tinyurl.com/4dcn9d5d`, accessed 2023-06-30.

rules to distribute these benefits between passengers. But despite these potential gains, encouraging passengers to share rides can be challenging; [67] describes Uber's marketing research process and finds that unpredictable detours are a major deterrent for UberPool. [39] discusses the subsequent launch of ExpressPool, a product that asks riders to walk short distances and wait briefly before being matched to co-riders, allowing the platform to make more efficient matches with shorter detours. The case then asks students to analyze the results of an experiment to decide whether to ask riders to wait longer to be matched, which improves matching but also incurs the risk of riders canceling their rides, a trade-off that also arises in our case. However, instead of directly providing students with the results of a given matching policy, we ask students to design and evaluate their *own* pricing and matching policies on large-scale data. Our setup allows students to creatively explore different approaches and mimics the experience of deploying an experiment in a real-world technology company.

Beyond ridesharing, bikesharing has been used to teach process flow [2], inventory management [48, 76], and demand analysis [82]; carsharing has been used to teach capital budgeting [52]; and car rental has been used to evaluate wage structure and outsourcing decisions [90]. These cases highlight the value of using shared mobility applications to teach classic operations problems in modern contexts.

Our case also relates to classroom simulation games. For example, previous games have placed students in the roles of actors in a supply chain [98, 28, 124] or a healthcare system [93, 64]; students can order inventory, price goods, treat patients, and schedule jobs, among other decisions. The games simulate complex system dynamics that respond to students' decisions, for an engaging and immersive classroom experience. Our case most closely relates to games that involve pricing in a variety of industry contexts: tourism [100, 45], rental cars [49], hotels [59], as well as ridesharing [78]. These games expose students to concepts such as demand uncertainty, valuation-based pricing, dynamic pricing, customer segmentation, capacity management, and competition, in an engaging and interactive setting. Our case touches on many of these concepts (chiefly demand uncertainty, valuation-based pricing, and dynamic pricing) in ridesharing. We also incorporate an additional component of matching riders to illustrate how pricing interacts and integrates with other decisions,

thereby providing a holistic view of dynamic decision-making for a ridesharing platform.

4.3 Learning Objectives

(1) Understand the online and dynamic nature of an on-demand ridesharing platform, and identify key sources of uncertainty and associated trade-offs involved in dynamic pricing and matching.

(2) Apply data analytics techniques to Calyber's historical operational data and provide insights into demand patterns and rider behavior.

(3) Design and implement data-driven policies for dynamic pricing and matching.

(4) Experiment with and evaluate the developed dynamic pricing and matching policies using validation and test data. Interpret key performance indicators, critique and refine the policies, and discuss implications for Calyber's shared rides offering.

4.4 Audience, Prerequisites and Position in Course

The case is best suited for senior undergraduate or graduate students with an analytics-related background, or for research-oriented PhD students seeking hands-on experience in developing data-driven dynamic policies. For junior undergraduate students, the teaching plan can be modified to incorporate more instructor guidance. The case has been successfully implemented with a range of students, from sophomores to PhDs.

Students are expected to have basic proficiency in Python programming and data analytics, as they will work with a dataset of over 10,000 riders and implement their policies using a provided skeleton file. For students with less programming experience, a set of tutorial-style Jupyter notebooks is provided, starting with descriptive analytics on Calyber's historical data and progressing to step-by-step instructions for building Thea's policy.

Although familiarity with (dynamic) optimization and simulation is helpful for designing and evaluating effective policies, it is not required. The case is *self-contained* and can be successfully completed without prior knowledge in these areas.

The case can complement a wide range of courses, including topical offerings in supply chain and logistics (e.g., inventory control and online order fulfillment), marketplace design (e.g., dynamic matching), as well as methodological courses in dynamic optimization and analytics.

4.5 Teaching Plan

This section presents some general guidelines of using the case.

Logistics. This case is intended to be implemented as a 5–6 week group project during the second half of a semester- or quarter-long course. We recommend structuring the case around three class sessions and two rounds of submissions. It begins with an introduction session, followed by several weeks of data exploration and policy development. Midway, teams submit their policies for a validation round evaluation and receive feedback, and the instructor can hold a discussion session. The final weeks are dedicated to improvements based on validation feedback and preparing a final submission for the test round. The last class session is used to announce final rankings based on test data, present strategies and outcomes, and reflect on key insights.

The case can also be run as a 2–3 week intensive module, with the first week for instructor-led tutorials, the second for policy development, and the final week for validation, testing, and presentations. This format has been used successfully in a three-week summer camp for sophomore and junior engineering students, supported by tutorial-style Jupyter notebooks in the supplemental materials.

Implementation. For classroom implementation, we recommend using GitHub Classroom, which enables students to form teams and work in team-specific repositories automatically forked from a master template. This setup facilitates collaborative development, version control, and seamless submission of deliverables through each team's assigned repository.

Case Analysis. If the instructor implements the case as a 5–6 week group project within a semester- or quarter-long course, we recommend walking students through a case analysis during the validation session (the second class session). The case analysis covers the ridesharing platform's dynamics, the development of Thea's policies, and possible improvements. Going over the analysis in a later stage encourages exploration and prevents students from being constrained by a limited set of prescribed methods. It is also more effective at a later stage, as students will already be familiar with the dataset and will have developed and evaluated preliminary pricing and matching policies, enabling them to better engage with and appreciate the analysis. Alternatively, if the instructor delivers the case as a three-week intensive module, this analysis can be conducted during the first week to jump-start the teams' development process.

4.6 Teaching Experience

This case has been taught at UC Berkeley three times: twice in a graduate course on supply chain and logistics, INDENG C253/CIVENG C258 Supply Chain and Logistics Management (Spring 2024, 2025), and once in an undergraduate engineering course in the Transfer Pre-Engineering Program (Summer 2024). Table 4.1 summarizes the class sizes across these offerings. In the graduate course, the case was implemented as a six-week group project during the second half of the semester. Each student on average spent roughly 1–2 hours per week on the case, with more time devoted toward the end of the six-week period. In the undergraduate course, the case was delivered as a three-week (13-hour) intensive module in a summer session, during which students worked exclusively during scheduled class time and did not spend time offline.

Course	**Level**	**Academic Term**	**# of Students**	**# of Teams**
INDENG C253/CIVENG C258	Graduate	Spring 2024	34	15
INDENG C253/CIVENG C258	Graduate	Spring 2025	30	11
Transfer Pre-Engineering Program	Undergraduate	Summer 2024	14	5

Table 4.1: Summary of previous class implementations.

In following subsections, we review the student performance in the previous implemen-

tations, and analyze the key trade-offs reflected in their policies.

4.6.1 Performance

Here we look at students' overall performance, and their comparison with the two benchmark policies (Nick's and Thea's policies). This is intended to give the instructor a high-level idea of what to expect, but these results may not be representative of students in other courses and at other universities.

As shown in Figure 4.1, most of the student teams were able to design profitable policies, but did not necessarily improve over the benchmark of Thea's engine (red). To show how they achieved this performance, we display the median key performance indicators across the student teams, as compared to those for Nick's and Thea's engines, in Table 4.2. The student teams typically charged lower prices than Thea's engine ($0.73/mile instead of $0.83/mile) and thereby achieved higher throughput (8.15 riders/minute instead of 4.02 riders/minute). However, they were not able to capitalize on this higher throughput to match more effectively, as the average cost per rider was similar ($0.62/mile versus $0.63/mile) despite a higher match rate (0.71 versus 0.59). Meanwhile, Figure 4.1 presents high variance in student performance, highlighting the varying effectiveness of their policies in balancing revenue making and cost control.

Key Performance Indicator	**Nick's Engine**	**Thea's Engine**	**Team Median**
Profit ($/min)	-3.76	2.47	1.96
Throughput (#/min)	16.82	4.02	8.15
Match rate	0.80	0.59	0.71
Detour rate	0.14	0.14	0.10
Average quoted price ($/mile)	0.56	0.83	0.73
Average cost per rider ($/mile)	0.62	0.63	0.62

Table 4.2: Median KPIs across student teams, compared to those for Nick's and Thea's engines.

4.6.2 Trade-offs

The performance metrics of the teams reveal important trade-offs in both pricing and matching decisions. Note that the plots in this section have their x-axis values hidden

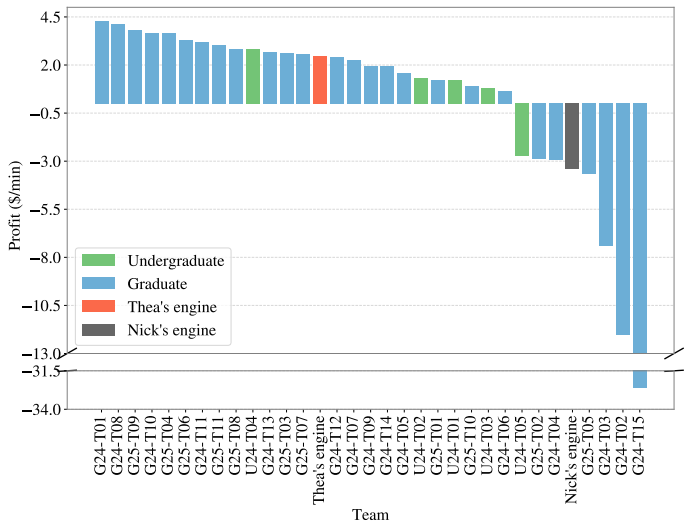


Figure 4.1: Student team profits, compared to Nick's and Thea's engines.
Note: Graduate teams are named as GYY-TXX, where YY is the last two digits of the year (e.g., 24 for 2024), and XX identifies the team. Similarly, undergraduate teams are labeled as U24-TXX (e.g., U24-T03 for Undergraduate Team 03 in 2024).

to avoid revealing sensitive details about the performance and specifications of student policies. Unredacted plots are available in the supplemental teaching note of [96].

Pricing trade-off. Figure 4.2 shows how profit varies with the average quoted price: profit tends to increase with price up to a point, and then decreases as prices continue to rise. Meanwhile, there is a clear boundary between teams that are profitable (blue) and unprofitable (orange)—teams quoting very low prices consistently ended up with negative profits as they fail to recover the costs. On the other hand, very high prices may suppress demand and matching efficiency, reducing overall profitability. This results in a concave trend in Figure 4.2, highlighting the importance of striking a balance in pricing.

Matching trade-off. We focus on the subset of teams that achieved positive profit, as the negative-profit teams generally failed due to underpricing, making their matching decisions less informative. Figure 4.3 presents a clear downward trend between profit and cost per rider; it indicates that reducing the cost per rider, potentially through an appropriate

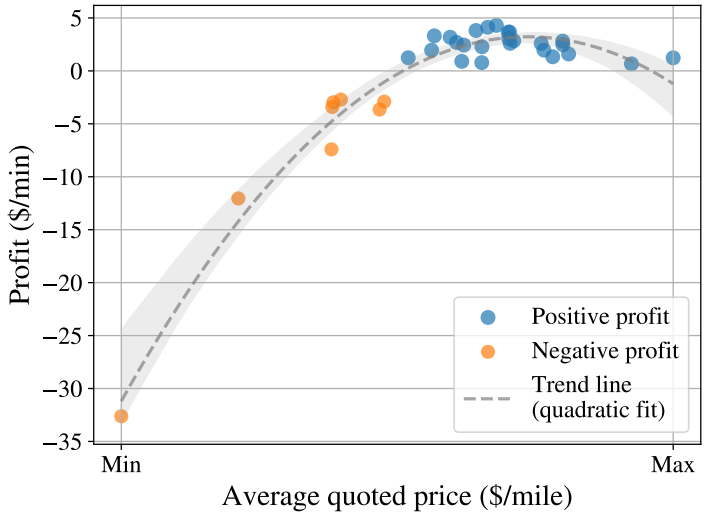


Figure 4.2: Profit vs. average quoted prices among student teams.
Note: Instructors can refer to Figure 3 in the teaching note of [95] for the unredacted plot showing actual values of the average quoted price.

amount of high-quality matching, generally improves profitability. Meanwhile, teams with a higher profit typically have an intermediate match rate (yellow), while those with excessively high (green) or low (red) match rates tend to be less profitable, probably as a result of inefficient or failed matching. Thus, achieving cost-effective matching requires not only maintaining a reasonably high match rate, but also a high match quality.

4.6.3 Improvements Since the Validation Round

The students also benefited from the optional validation round, using it to test and refine their policies. Of the 31 student teams, 20 participated in the validation round (16/26 graduate, 4/5 undergraduate). Of these 20 teams, 17 improved their profits (13/16 graduate, 4/4 undergraduate) typically by raising prices and/or improving matching quality.

4.6.4 Student Groups

As the undergraduate class size was relatively small with only five teams, it is hard to draw strong conclusions on the performance comparison between the undergraduate and graduate

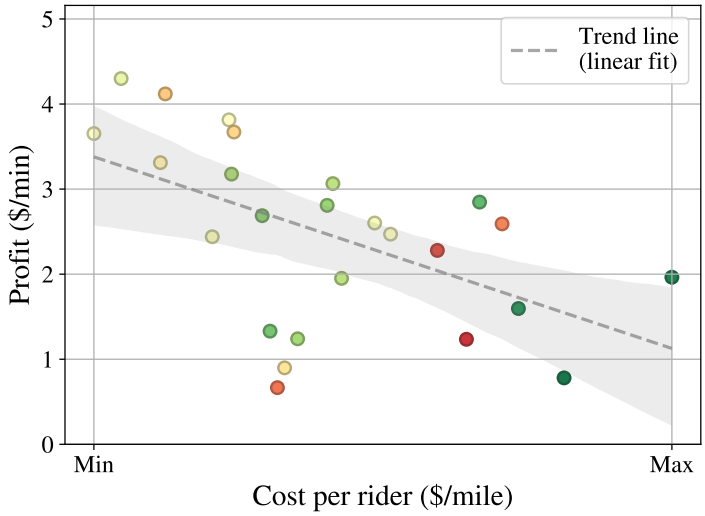


Figure 4.3: Profit vs. cost per rider among student teams.
Note: The node color represents the match rate (green for the highest, red for the lowest, and yellow for a medium level). Instructors can refer to Figure 4 in the teaching note of [95] for the unredacted plot showing actual values of cost per rider.

students. However, as shown in Figure 4.1, there is no notable performance gap between undergraduate and graduate teams. This suggests that the case is suitable for students across a wide range of academic levels and technical backgrounds. It also highlights that with appropriate additional guidance (such as the tutorial-style notebooks), undergraduate students can develop reasonable policies that achieve performance comparable to many graduate teams, even if their approaches are more intuitive and methodologically simpler.

4.6.5 Feedback

We report student feedback in Table 4.3 across three in-person offerings: the graduate course INDENG C253/CIVENG C258: Supply Chain and Logistics Management at UC Berkeley in Spring 2024 and Spring 2025, as well as the Transfer Pre-Engineering Program (T-PREP) Summer 2024 bridge program for incoming undergraduate transfer students to Berkeley Engineering. Across all cohorts, students consistently rated the case as **moderately to highly challenging**. The graduate sections reported average difficulty scores of **5.50/7**

(2024) and **4.89/7** (2025), while the T-PREP students—many of whom had limited prior exposure to Python, analytics, or operations research—reported a difficulty of **3.71/5**. Despite the challenge, written comments emphasized that the effort was rewarding: one group remarked that the project was "very time-consuming, but we really learned new things through discussion and collaboration," and multiple students across programs described the experience as "the highlight of the course."

Student endorsement of the pedagogical value was similarly strong. Overall, students recommended that the case be taught again, with scores of **6.44/7** in both graduate offerings and **4.36/5** in the T-PREP cohort. In Spring 2025, graduate students agreed that the case increased their awareness of real-world transportation issues, with a score of **6.25/7**; one student described the case as "a very valuable experience [that] piqued my interest in ridesharing and inspired me to explore this topic further". These students also reported strengthened understanding of analytics and operations research tools and how they are applied in practice (**6.13/7**). The T-PREP cohort also showed learning gains, although their more limited background could be a barrier: the case increased awareness of transportation issues (**4.14/5**) and improved understanding of analytics/OR tools (**3.86/5**).

Across cohorts, the written feedback reveals differences often rooted in students' background. Graduate students emphasized the excitement of seeing "heuristic and simple strategies perform super well," and then being able to "refine [the models] in a data-driven manner under [the instructor's] suggestions." In contrast, T-PREP students requested more instruction and guided preparation early on, and more opportunities for testing and feedback later. Nevertheless, students cited the relevance of the ridesharing context, the engaging format, and the sense that the case deepened their understanding of data-driven modeling and operations analytics. Together, the results demonstrate that the case not only meets its intended learning objectives but also scales effectively across diverse student populations.

4.7 Conclusion

This case introduces Calyber, a simulation-based game that challenges students to design data-driven pricing and matching policies to maximize the profitability of the shared rides platform. By placing students in the role of platform operators making real-time decisions,

Dimension / Question	INDENG C253 (Spring 2024)	INDENG C253 (Spring 2025)	T-PREP (Summer 2024)
Response Count / Enrollment	19 / 34	9 / 30	14 / 14
Response Rate	55.9%	30.0%	100.0%
How difficult was the game?	5.50 / 7	4.89 / 7	3.71 / 5
How interesting was the game?	6.39 / 7	—	—
How useful was the experience?	6.18 / 7	—	—
Increased awareness of real-world transportation issues	—	6.25 / 7	4.14 / 5
Improved understanding of analytics/OR tools	—	6.13 / 7	3.86 / 5
Recommend keeping/playing the game again	6.44 / 7	6.44 / 7	4.36 / 5
Familiarity with ridesharing (pre-game)	—	4.89 / 7	1.43 / 5
Familiarity with Python (pre-game)	—	—	2.79 / 5
Familiarity with analytics/OR (pre-game)	—	—	1.21 / 5
Evaluation process is fair/helpful	—	—	4.29 / 5

Table 4.3: Summary of student feedback across cohorts.

the case provides hands-on experience with key trade-offs in a dynamic, uncertain environment. Students engage with large-scale data, develop and test policies, and are evaluated through multiple rounds.

The case is suitable for a broad range of learners, from undergraduates with basic data analytics skills to graduate and doctoral students with stronger technical backgrounds. It has been implemented successfully in both student groups—students have created meaningful policies, shown improvement through multiple rounds, and expressed high levels of engagement and interest in the topic.

This case offers instructors an interactive, data-rich tool for teaching stochastic dynamic decision-making. It contributes a novel perspective to the growing set of simulation games, particularly by integrating both pricing and matching decisions—a realistic operational challenge in ridesharing platforms.

The model in the case simplifies ridesharing operations for classroom use. First, we only consider a shared rides product, whereas companies such as Uber and Lyft simultaneously offer a separate solo rides product. The price and associated travel time under this solo rides product then influences riders' conversion behavior to the shared rides product. We made this simplification to focus on the key pricing and matching trade-offs, and because operating a shared rides service in isolation is already challenging (as evidenced by the wide range of performances achieved by the different student groups). Second, riders are dispatched in a solo ride after their waiting patience is exceeded, whereas in practice, riders

might also cancel their requests. Future cases can highlight considerations such as the interaction between shared rides and solo rides products, or more complex models of rider patience.

Chapter 5

CONCLUSION

This dissertation examined how ridesharing platforms can design pricing and matching policies to make shared rides operationally sustainable. The three chapters approached this question from different perspectives: the first work focuses on pricing—why platforms struggle with pricing, and proposing an alternative adaptive pricing policy based on rider's matching outcome; beyond pre-trip matching that is assumed in the first work, the second work explicitly studies on-trip matching decisions—how to make both pre-trip and on-trip matching decisions, and contrast the behavior of each phase of matching; while the first two works develop analytical and computational contributions, the third work complements them with developing a teaching case that places students in the role of platform operators confronting the joint problem in real time.

The first work analyzes the challenges that platforms face in operating shared rides in terms of pricing policy as well as a path forward. We show that a heuristic, iterative approach to static pricing tends to overprice, suppressing demand and confining operations to dense downtown areas. Meanwhile, even the optimal static price falls short, with the risk of failing to reduce costs by efficient matching. We propose a match-based pricing policy that varies depending on the matching outcomes, by charging more when a rider is dispatched solo and offering a discount when it is matched. We establish theoretical results for a single origin-destination pair and validate them through large-scale simulations on real Chicago data using scalable, computationally simple methods. Relative to the optimal static pricing policy, our approach achieves higher match rate and cost efficiency while reducing payments for riders. The gains are most significant in low-demand, high-cost settings where shared rides have historically struggled to operate.

The second work focuses on studying when and how to apply on-trip matching. Pre-trip matching commits to matches before dispatch, so a rider dispatched solo stays solo

for the remaining trip. On-trip matching relaxes this constraint, allowing the platform to match riders even after one of them has been dispatched solo; but it requires tracking each rider throughout her entire journey, since her matching opportunities and value evolve with her position. In this work, we develop joint pre-trip and on-trip matching policies and characterize the conditions under which each is favorable. We then embed the matching model within an outer pricing optimization problem to study how pricing and matching decisions interact. Across both synthetic and real-world data, we find that pre-trip matching performs best in dense downtown areas that are close to the ideal single-origin-destination setting, while on-trip matching performs best in sparse outskirts where riders are spatially dispersed, allowing the platform to balance matching opportunity against matching value. Introducing on-trip matching raises both profit and efficiency, enabling lower rider prices for shared rides, particularly in the sparse outskirts. Since the demand in large metropolitan areas is substantially dispersed, sustaining shared rides at that scale requires both matching regimes together, highlighting why this new paradigm needs careful modeling.

The third work introduces Calyber, a simulation-based game in which students design data-driven pricing and matching policies to maximize the profitability of a shared-rides platform. By playing the role of platform operators making real-time decisions, students gain hands-on experience navigating key trade-offs in a dynamic, uncertain environment, working with large-scale data to develop and test policies that are evaluated across multiple rounds. The case suits a broad range of learners, from undergraduates with basic data-analytics skills to graduate and doctoral students with stronger technical backgrounds. It has been deployed successfully with both groups—students developed meaningful policies, improved over successive rounds, and reported strong engagement and interest with the case. For instructors, the case offers an interactive, data-rich tool for teaching stochastic dynamic decision-making, and contributes a fresh perspective to the growing body of simulation games by jointly incorporating pricing and matching—a realistic operational challenge for ridesharing platforms.

Taken together, these three works share a single conviction that pricing and matching in shared rides cannot be designed in isolation. The first work shows that conditioning prices on matching outcomes improves operation in high-cost, low-demand settings where

traditional static pricing typically fails. The second shows that combining pre-trip and on-trip matching, jointly optimized with pricing, is able to sustain shared rides across the spatially dispersed demand of a full metropolitan area. Beyond their individual results, the first two chapters identify the operating regimes that allow shared rides to extend beyond dense downtowns into the sparse outskirts where the service has historically been most challenging. The third turns the joint optimization problem into a hands-on simulation game, letting students experience the same pricing and matching trade-offs that the previous chapters study. Across all three, the message is consistent: the promise of shared rides lies not in any single lever but in the coordinated design of pricing and matching together.

BIBLIOGRAPHY

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[2] Deniz Akturk and Ozan Candogan. Mobismile Bike-Sharing. Unpublished teaching case, 2024.

[3] Gad Allon. Pool is Back. `https://gadallon.substack.com/p/pool-is-back`, 2022. Accessed: 2026-01-03.

[4] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[5] Javad Jomehpour Chahar Aman and Janille Smith-Colin. Transit deserts: Equity analysis of public transit accessibility. *Journal of Transport Geography*, 89:102869, 2020.

[6] Saurabh Amin, Patrick Jaillet, Haripriya Pulyassary, and Manxi Wu. Market design for dynamic pricing and pooling in capacitated networks. *arXiv preprint arXiv:2307.03994*, 2023.

[7] Priya Anand. Uber burned more than $1 million a week on UberPool to win San Francisco. `https://www.buzzfeednews.com/article/priya/uber-pool-burn-rate-frisco`, 2017. Accessed: 2025-06-19.

[8] Ivory Sherman Andrew Greiner, Matt McFarland and Jen Tse. A history of Lyft, from fuzzy pink mustaches to global ride share giant. `https://www.cnn.com/interactive/2019/03/business/lyft-history/index.html`, 2019. Accessed: 2025-06-19.

[9] Ali Aouad and Ömer Sarıtaç. Dynamic stochastic matching under limited time. *Operations Research*, 70(4):2349–2383, 2022.

[10] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. Edge-weighted online windowed matching. *Mathematics of Operations Research*, 48(2):999–1016, 2023.

[11] Xabi Azagirre, Akshay Balwally, Guillaume Candeli, Nicholas Chamandy, Benjamin Han, Alona King, Hyungjun Lee, Martin Loncaric, Sébastien Martin, Vijay Narasiman, et al. A better match for drivers and riders: Reinforcement learning at Lyft. *INFORMS Journal on Applied Analytics*, 54(1):71–83, 2024.

[12] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. Pricing in ride-sharing platforms: A queueing-theoretic approach. In *Proceedings of the 2015 ACM Conference on Economics and Computation*, page 639, 2015.

[13] Sergio Barbarino and Robert N Boute. Game—How to share the gains of collaboration? *INFORMS Transactions on Education*, 23(2):57–63, 2023.

[14] James Barron. Where Yellow Cabs Didn't Go, Green Cabs Were Supposed to Thrive. Then Came Uber. `https://www.nytimes.com/2018/09/03/nyregion/green-cabs-yellow-uber.html`, 2018. Accessed: 2025-08-07.

[15] Saif Benjaafar, Harald Bernhard, and Costas Courcoubetis. Drivers, riders and service providers: The impact of the sharing economy on mobility. *Management Science*, 68(1):123–142, 2022.

[16] Omar Besbes, Francisco Castro, and Ilan Lobel. Surge pricing and its spatial supply response. *Management Science*, 67(3):1350–1367, 2021.

[17] Kostas Bimpikis, Ozan Candogan, and Daniela Saban. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.

[18] Sebastian Birolini, Alexandre Jacquillat, Phillip Schmedeman, and Nuno Ribeiro.

Passenger-centric slot allocation at schedule-coordinated airports. *Transportation Science*, 57(1):4–26, 2023.

[19] Geoff Boeing. Modeling and analyzing urban networks and amenities with OSMnx. *Geographical Analysis*, 57:567–577, 2025.

[20] Rainer E Burkard, Bettina Klinz, and Rüdiger Rudolf. Perspectives of monge properties in optimization. *Discrete Applied Mathematics*, 70(2):95–161, 1996.

[21] Gerard P Cachon, Kaitlin M Daniels, and Ruben Lobel. The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management*, 19(3):368–384, 2017.

[22] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 241–242, 2017.

[23] Guoli Chen, Kuangzhen Wu, Tony Tong, and Xiaohua Su. Uber vs. DiDi: The Race for China's Ride-hailing Market. INSEAD Case IN1306, 2016.

[24] Junlin Chen, Chiwei Yan, and Hai Jiang. On the linear programming model for dynamic stochastic matching and its application to pricing. *arXiv preprint arXiv:2506.09924*, 2025.

[25] William Bohan Chinique. UberPOOL FAQs. `https://www.uber.com/newsroom/uberpool-faqs/`, 2015. Accessed: 2024-05-14.

[26] Natalie Collina, Nicole Immorlica, Kevin Leyton-Brown, Brendan Lucier, and Neil Newman. Dynamic weighted matching with heterogeneous arrival and departure rates. In *Web and Internet Economics: 16th International Conference, WINE 2020, Beijing, China, December 7–11, 2020, Proceedings 16*, pages 17–30, 2020.

[27] Carlos F Daganzo, Yanfeng Ouyang, and Haolin Yang. Analysis of ride-sharing with service time and detour guarantees. *Transportation Research Part B: Methodological*, 140:130–150, 2020.

[28] Sophie D'Amours, Philippe Marier, Mikael Rönnqvist, Riadh Azouzi, and Dag Fjeld. Game—The online wood supply game. *INFORMS Transactions on Education*, 18(1):71–87, 2017.

[29] Jackie Davalos. Lyft will discontinue pooled rides. `https://www.bloomberg.com/news/articles/2023-05-11/lyft-will-discontinue-pooled-rides-roll-out-new-features`, 2023. Accessed: 2023-07-10.

[30] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[31] Daniela Pucci de Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[32] Arjan de Ruijter, Oded Cats, Javier Alonso-Mora, and Serge Hoogendoorn. Ride-pooling adoption, efficiency and level of service under alternative demand, behavioural and pricing settings. *Transportation Planning and Technology*, 46(4):407–436, 2023.

[33] Saahil Desai. UberPool is a Zombie. `https://www.theatlantic.com/technology/archive/2022/07/uberx-share-carpooling-ride-app-cost/661483/`, 2022. Accessed: 2023-07-10.

[34] Tom Dotan. Uber insiders say the company's cheapest rides are being quietly killed off because they lost so much money. `https://www.businessinsider.com/uberpool-cheap-shared-rides-are-not-coming-back-after-pandemic-2021-5`, 2021. Accessed: 2024-05-13.

[35] Jonathan Dumas and François Soumis. Passenger flow model for airline networks. *Transportation Science*, 42(2):197–207, 2008.

[36] Myungeun Eom and Alejandro Toriello. Batching and greedy policies: How good are they in dynamic matching? *Manufacturing & Service Operations Management*, 28(2):479–495, 2026.

[37] Alireza Ermagun and Nebiyou Tilahun. Equity of transit accessibility across Chicago. *Transportation Research Part D: Transport and Environment*, 86:102461, 2020.

[38] Vivek F Farias and Benjamin Van Roy. Tetris: A study of randomized constraint sampling. In *Probabilistic and randomized methods for design under uncertainty*, pages 189–201. 2006.

[39] Chiara Farronato, Alan MacCormack, and Sarah Mehta. Innovation at Uber: The launch of Express POOL. Harvard Business School Case 619-003, 2018. (Revised July 2023).

[40] Yiding Feng, Rad Niazadeh, and Amin Saberi. Two-stage stochastic matching and pricing with applications to ride hailing. *Operations Research*, 72(4):1574–1594, 2024.

[41] Andres Fielbaum, Xiaoshan Bai, and Javier Alonso-Mora. On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, 126:103061, 2021.

[42] Andres Fielbaum, Rafał Kucharski, Oded Cats, and Javier Alonso-Mora. How to split the costs and charge the travellers sharing a ride? aligning system's optimum with users' equilibrium. *European Journal of Operational Research*, 301(3):956–973, 2022.

[43] Masabumi Furuhata, Kenny Daniel, Sven Koenig, Fernando Ordonez, Maged Dessouky, Marc-Etienne Brunet, Liron Cohen, and Xiaoqing Wang. Online cost-sharing mechanism design for demand-responsive transport systems. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):692–707, 2014.

[44] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

[45] Gamelab Education. PricinGame, 2025.

[46] Noah Garfinkel. Uber brings back shared rides to nine U.S. cities this summer. `https:`

//www.axios.com/2022/06/21/uber-shared-rides-2022-uberx-share, 2022. Accessed: 2024-05-14.

[47] Nikhil Garg and Hamid Nazerzadeh. Driver surge pricing. *Management Science*, 68(5):3219–3235, 2022.

[48] Jose Gomez-Ibanez and Trevor Johnston. Hubway (A): Bike Sharing in Boston. Harvard Kennedy School Case KS1180, 2016.

[49] John T. Gourville, Thomas T. Nagle, and John Hogan. Pricing Simulation: Universal Rental Car V2. Harvard Business School Simulation, 2011.

[50] Adam Greenhall. Experimentation in a ridesharing marketplace. https://eng.lyft.com/https-medium-com-adamgreenhall-simulating-a-ridesharing-marketplace-36007a8a31f2, 2016. Accessed: 2023-06-29.

[51] Ranjay Gulati and Jeffrey Huizinga. Lyft 2023: Roads to Growth and Differentiation. Harvard Business School Case 424-060, 2024.

[52] Owen Hall and Kenneth Ko. Equuleus Car Sharing Inc.: Revenue Management. Ivey Publishing Case W18404, 2018.

[53] Ming Hu, Jianfu Wang, and Hengda Wen. Share or solo? individual and social choices in ride-hailing. *Available at SSRN 3675050*, 2020.

[54] Ming Hu and Yun Zhou. Dynamic type matching. *Manufacturing & Service Operations Management*, 24(1):125–142, 2022.

[55] Shichun Hu, Maged M Dessouky, Nelson A Uhan, and Phebe Vayanos. Cost-sharing mechanism design for ride-sharing. *Transportation Research Part B: Methodological*, 150:410–434, 2021.

[56] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. How to match when all vertices arrive online. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 17–29, 2018.

[57] Marco Iansiti, Michael W. Toffel, and Amram Migdal. uberPOOL. Harvard Business School Case 617-009, 2016. (Revised March 2017).

[58] Jagan Jacob and Ricky Roet-Green. Ride solo or pool: Designing price-service menus for a ride-sharing platform. *European Journal of Operational Research*, 295(3):1008–1024, 2021.

[59] Florencia Jaureguiberry and Mariano Tappata. Game—The Hotel Game: Pricing Simulations with Opaque and Transparent Channels. *INFORMS Transactions on Education*, 16(1):24–38, 2015.

[60] Junfeng Jiao and Maxwell Dillivan. Transit deserts: The gap between demand and supply. *Journal of Public Transportation*, 16(3):23–39, 2013.

[61] John Zimmer. The end of traffic. `https://medium.com/@johnzimmer/the-end-of-traffic-6d255c03207d`, 2017. Accessed: 2025-06-19.

[62] Paul Karaenke, Maximilian Schiffer, and Stefan Waldherr. On the benefits of ex-post pricing for ride-pooling. *Transportation Research Part C: Emerging Technologies*, 155:104290, 2023.

[63] Jintao Ke, Hai Yang, Xinwei Li, Hai Wang, and Jieping Ye. Pricing and equilibrium in on-demand ride-pooling markets. *Transportation Research Part B: Methodological*, 139:411–431, 2020.

[64] Nan Kong. Active game-based learning of dynamics modeling and simulation in biomedical systems engineering. *INFORMS Transactions on Education*, 20(1):16–25, 2019.

[65] Leo Kroon, Gábor Maróti, and Lars Nielsen. Rescheduling of railway rolling stock with dynamic passenger flows. *Transportation Science*, 49(2):165–184, 2015.

[66] Tatsiana Levina, Yuri Levin, Jeff McGill, and Mikhail Nediak. Network cargo capacity management. *Operations Research*, 59(4):1008–1023, 2011.

[67] Jenny Lo and Steve Morseman. The Perfect uberPOOL: A Case Study on Trade-Offs. *Ethnographic Praxis in Industry Conference Proceedings*, 2018(1):195–223, 2018.

[68] Ilan Lobel and Sebastien Martin. Detours in shared rides. *Management Science*, 71(2):1716–1736, 2025.

[69] Alice Lu, Peter I Frazier, and Oren Kislev. Surge pricing moves Uber's driver partners. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 3–3, 2018.

[70] Ingrid Lunden. Uber says that 20% of its rides globally are now on UberPool. `https://techcrunch.com/2016/05/10/uber-says-that-20-of-its-rides-globally-are-now-on-uber-pool`, 2016. Accessed: 2023-07-10.

[71] Natalie Lung. Lyft revives pooled rides at airports in push for cheaper trips. `https://fortune.com/2025/05/19/lyft-pooled-rides-at-airports-cheaper-trips/`, 2025. Accessed: 2026-01-03.

[72] Lyft. Lyft's New App Creates Positive Change for Passengers and Cities. `https://www.lyft.com/blog/posts/new-app`, 2018. Accessed: 2023-07-10.

[73] Lyft. Shared Rides Return to More Cities. `https://www.lyft.com/blog/posts/shared-rides-return-to-more-cities`, 2021. Accessed: 2023-07-10.

[74] Lyft. Lyft shared ride pricing. `https://help.lyft.com/hc/et/all/articles/115013080888-Lyft-Shared-ride-pricing`, 2024. Accessed: 2024-05-14.

[75] Hongyao Ma, Fei Fang, and David C Parkes. Spatio-temporal pricing for ridesharing platforms. *Operations Research*, 70(2):1025–1041, 2022.

[76] Victor Martinez de Albeniz Margalef. Bicing: Bike-Sharing in Barcelona. IESE Business School Case IES657, 2017.

[77] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[78] MobLab. Production, Entry & Exit. MobLab Simulation ML0009-HTM-ENG, 2018.

[79] Youngme Moon. Uber: Changing the Way the World Moves. Harvard Business School Case 316-101, 2015. (Revised January 2017).

[80] Morning Brew Daily. Uber CEO unveils plan to make rides cheaper, reveals own uber rating, and launches "party bus", 2024. Accessed: 2025-07-25.

[81] Almir Mutapcic and Stephen Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*, 24(3):381–406, 2009.

[82] Joe Naoum-Sawaya and Abby Xiong. Capital Bikeshare: Analyzing Bike Rental Demand. Ivey Publishing Case W21108, 2021.

[83] Amir Ali Nasrollahzadeh, Amin Khademi, and Maria E Mayorga. Real-time ambulance dispatching and relocation. *Manufacturing & Service Operations Management*, 20(3):467–480, 2018.

[84] New York Times. What Will New York Do About Its Uber Problem? `https://www.nytimes.com/2018/05/07/opinion/new-york-uber-problem.html`, 2018. Accessed: 2025-08-07.

[85] Fernando Ordóñez and Maged Dessouky. Dynamic ridesharing. In *INFORMS TutORials in Operations Research*, pages 212–236. INFORMS, 2017.

[86] Michael Ostrovsky and Michael Schwarz. Carpooling and the economics of self-driving cars. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 581–582, 2019.

[87] Erhun Özkan and Amy R Ward. Dynamic matching for real-time ride sharing. *Stochastic Systems*, 10(1):29–70, 2020.

[88] Marco Pavone, Amin Saberi, Maximilian Schiffer, and Matt Wu Tsao. Online hypergraph matching with delays. *Operations Research*, 70(4):2194–2212, 2022.

[89] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[90] Balaraman Rajan and N Ravichandran. Case—Tactical Decisions at Vastrapur Car Rental Services. *INFORMS Transactions on Education*, 18(1):52–55, 2017.

[91] Sachin Kansal. For every day. `https://www.uber.com/newsroom/go-get-2025/`, 2025. Accessed: 2025-06-26.

[92] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[93] Antoine Sauré and Martin L Puterman. The appointment scheduling game. *INFORMS Transactions on Education*, 14(2):73–85, 2014.

[94] Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

[95] Yifan Shen, Chiwei Yan, and Julia Yan. Case Article—Calyber: A Ridesharing Game. *INFORMS Transactions on Education*, 26(3):218–225, 2026.

[96] Yifan Shen, Chiwei Yan, and Julia Yan. Case—Calyber: A Ridesharing Game. *INFORMS Transactions on Education*, 26(3):226–232, 2026.

[97] Andrea Simonetto, Julien Monteil, and Claudio Gambella. Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies*, 101:208–232, 2019.

[98] John D Sterman. Instructions for running the beer distribution game. *System Dynamics Group Working Paper D-3679, MIT, Sloan School of Management, Cambridge, MA*, 1984.

[99] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological*, 82:36–53, 2015.

[100] Kalyan Talluri. The Customer Valuations Game as a Basis for Teaching Revenue Management. *INFORMS Transactions on Education*, 9(3):117–123, 2009.

[101] Terry A Taylor. Shared-ride efficiency of ride-hailing platforms. *Manufacturing & Service Operations Management*, 26(5):1945–1961, 2024.

[102] Thompson S. H. Teo, Sheryl E. Kimes, and Zhiyi Yong. Case Article—The Rise and Fall of Taxi Club Management in New York. *INFORMS Transactions on Education*, 20(1):26–27, 2019.

[103] The Verge. Uber CEO Dara Khosrowshahi on the company's new Route Share feature. `https://www.youtube.com/shorts/ERSAJRtrc2U`, 2025. Accessed: 2025-07-24.

[104] The Verge. Why Uber CEO Dara Khosrowshahi wants ride sharing to be "cool" again. `https://www.fastcompany.com/91056936/sxsw-uber-ceo-dara-khosrowshahi-sharing-rides-cool`, 2025. Accessed: 2025-07-24.

[105] TomTom. Chicago Traffic Report. `http://tinyurl.com/4aah3fae`, 2024. Accessed: 2024-01-31.

[106] Travis Kalanick. Uber's plan to get more people into fewer cars. `https://www.ted.com/talks/travis_kalanick_uber_s_plan_to_get_more_people_into_fewer_cars/`, 2016. Accessed: 2025-07-08.

[107] Uber. Meet Michelangelo: Uber's machine learning platform. `https://www.uber.com/blog/michelangelo-machine-learning-platform/`, 2017. Accessed: 2023-06-29.

[108] Uber. Introducing Express POOL: Walk a little to save a lot. `https://www.uber.com/newsroom/expresspool/`, 2018. Accessed: 2023-06-29.

[109] Uber. How does route-based pricing help? https://www.uber.com/us/en/marketplace/pricing/route-based-pricing/, 2019. Accessed: 2023-06-29.

[110] Uber. UberX Share. https://www.uber.com/us/en/ride/uberx-share/, 2022. Accessed: 2024-05-14.

[111] Uber. The history of Uber. https://www.uber.com/en-CA/newsroom/history/, 2025. Accessed: 2025-06-19.

[112] Uber. Introducing UberX Share. https://www.uber.com/newsroom/uberx-share/, 2026. Accessed: 2026-01-03.

[113] Uber. UberX Share. https://www.uber.com/gb/en/ride/uberx-share/, 2026. Accessed: 2026-01-03.

[114] Siying Wang, Xiaolei Wang, Chen Yang, Xiaoning Zhang, and Wei Liu. Optimizing OD-based up-front discounting strategies for enroute ridepooling services. *Transportation Research Part B: Methodological*, 189:103013, 2024.

[115] Xing Wang, Niels Agatz, and Alan Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 52(4):850–867, 2018.

[116] Xuan Wang and Renyu Zhang. Carpool services for ride-sharing platforms: Price and welfare implications. *Naval Research Logistics (NRL)*, 69(4):550–565, 2022.

[117] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 905–913, 2018.

[118] Chiwei Yan, Julia Yan, and Yifan Shen. Pricing Shared Rides. *Operations Research*, 2026.

[119] Chiwei Yan, Helin Zhu, Nikita Korolko, and Dawn Woodard. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)*, 67(8):705–724, 2020.

[120] Mischa Young, Steven Farber, and Matthew Palm. The true cost of sharing: A detour penalty analysis between UberPool and UberX trips in Toronto. *Transportation Research Part D: Transport and Environment*, 87:102540, 2020.

[121] Chenkai Yu and Hongyao Ma. Iterative network pricing for ridesharing platforms. *arXiv preprint arXiv:2311.08392*, 2023.

[122] Kenan Zhang and Yu Marco Nie. To pool or not to pool: Equilibrium, pricing and regulation. *Transportation Research Part B: Methodological*, 151:59–90, 2021.

[123] Wei Zhang, Alexandre Jacquillat, Kai Wang, and Shuaian Wang. Routing optimization with vehicle–customer coordination. *Management Science*, 69(11):6876–6897, 2023.

[124] Yao Zhao, Arim Park, Olena Rudna, and Ju Myung Song. Game—Florapark (the flower game): A supply chain contract and collaboration simulation. *INFORMS Transactions on Education*, 24(1):105–117, 2023.

[125] Feng Zhu and Angela Acocella. Fasten: Challenging Uber and Lyft with a New Business Model. Harvard Business School Case 616-062, 2016. (Revised March 2020).

[126] Pengbo Zhu, Giancarlo Ferrari-Trecate, and Nikolas Geroliminis. En Route Path-planning for Partially Occupied Vehicles in Ride-pooling Systems. In *2025 European Control Conference (ECC)*, pages 2952–2957, 2025.

Appendix A

SUPPLEMENTARY MATERIALS FOR CHAPTER 2

In this section, we discuss and provide analyses to extensions of our basic model of the single origin-destination setting in Section 2.3.

A.1 Solo, Shared Rides and Sharing Disutility

We analyze a single origin-destination setting with a rider demand model where riders choose between solo and shared rides and sharing is associated with a disutility $\beta > 0$. Since there is no detour, this disutility represents the loss of privacy and comfort.

Match-Based Pricing. As before, riders make requests if their willingness to pay is at least as high as the price, which is p_X for solo rides (X as in "UberX"), $p(0)$ for riders who arrive with no requests waiting on the platform, and $p(1)$ for riders who arrive with one request waiting on the platform. Specifically, let v be the rider's willingness to pay, and let u_X, $u(0)$, and $u(1)$ be the expected utilities for a solo ride, a shared ride requested when no shared requests are waiting in the system (state 0), and a shared ride requested when another shared request is waiting in the system (state 1), respectively.

$$u_X = v - p_X, \tag{A.1a}$$

$$u(0) = v - p(0) - \beta \frac{\lambda(1)}{\lambda(1) + \theta}, \tag{A.1b}$$

$$u(1) = v - p(1) - \beta. \tag{A.1c}$$

For (A.1b), recall from Section 2.3.1 that for a rider who arrives with no other request waiting on the platform, the probability of being matched with another request is $\lambda(1)/(\lambda(1) + \theta)$, where $\lambda(1)$ is the conversion rate of shared requests at state 1. Hence, the term $\beta\lambda(1)/(\lambda(1)+\theta)$ represents an expected sharing disutility in state 0.

From (A.1), the conversion rates given prices p_X, $p(0)$, and $p(1)$ are as follows. If a rider arrives with one shared request waiting on the platform (state 1), she chooses a shared ride at rate

$$\lambda(1) = \begin{cases} 1 - \beta - p(1), & \text{if } p_X > p(1) + \beta, \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

Similarly, if a rider arrives with no shared requests waiting on the platform (state 0), she chooses a shared ride at rate

$$\lambda(0) = \begin{cases} 1 - \beta \dfrac{\lambda(1)}{\lambda(1) + \theta} - p(0), & \text{if } p_X > p(0) + \beta \dfrac{\lambda(1)}{\lambda(1) + \theta}, \\ 0, & \text{otherwise.} \end{cases} \tag{A.3}$$

Finally, riders choose solo rides at rate

$$\lambda_X = (1 - p_X)\left(\pi_0 \mathbb{1}\left\{p_X \le p(1) + \beta\right\} + \pi_1 \mathbb{1}\left\{p_X \le p(0) + \beta \frac{\lambda(1)}{\lambda(1) + \theta}\right\}\right),$$

recalling from (2.5) that π_0 and π_1 are the steady-state probabilities of states 0 and 1, respectively. The platform then seeks to optimize the total profit,

$$\max_{p_X,\, p(0),\, p(1)} \quad \frac{\lambda(0)}{\lambda(0) + \lambda(1) + \theta}\left(\lambda(1)\left(p(0) + p(1) - c\right) + \theta\left(p(0) - c\right)\right) + \lambda_X(p_X - c). \quad \text{(A.4)}$$

It can be seen that all conversion decisions do not depend on the rider's valuation v. Therefore, at a given system state, all riders will either entirely convert to solo rides or shared rides. Then, the platform operates shared rides only if all riders convert to shared rides at both states 0 and 1. Otherwise, if (1) all riders convert to solo rides in state 0 and state 1 is never reached, or if (2) all riders convert to shared rides in state 0 and solo rides in state 1, then the platform only operates solo rides. Note that the second case cannot be more profitable than the first case, since the platform fails to match any shared requests in state 1. The platform either operates only shared rides, only solo rides, or shuts down service entirely. The following proposition characterizes the conditions for each situation.

Theorem 4. *For any $c \in (0, 2)$, $\theta > 0$, and $\beta \ge 0$, there exists a triplet of profit-maximizing conversion rates $(\lambda^*(0), \lambda^*(1), \lambda_X^*)$. In particular, the following triplet is an optimal solution:*

$$(\lambda^*(0), \lambda^*(1), \lambda_X^*) = \begin{cases} \left(\dfrac{(1 - c/2 - \beta)^2 + \theta(1 - c)}{2 - c - 2\beta + 2\theta},\ \dfrac{(1 - c/2 - \beta)^2 + \theta(1 - 2\beta)}{2 - c - 2\beta + 2\theta},\ 0\right), & \text{if } c \in (0, 1], \beta \in [0, c/2], \theta > 0, \\ & \quad \text{or } c \in (1, 2), \beta \in [0, 1 - c/2), \\ & \quad \theta < (1 - c/2 - \beta)^2/(c - 1), \\ (0,\ \text{any } \lambda(1) \in [0, 1],\ (1 - c)/2), & \text{if } c \in (0, 1), \beta > c/2, \theta > 0, \\ (0,\ \text{any } \lambda(1) \in [0, 1],\ 0), & \text{otherwise.} \end{cases}$$

The proof is provided in Online Appendix A.3.

Static Pricing. The static pricing counterpart can be described with a solo price p_X and a static shared price p_S, leading to the following expected utilities for solo ride u_X and shared ride u_S:

$$u_X = v - p_X, \quad \text{(A.5a)}$$

$$u_S = v - \beta\rho - p_S = v - \beta \frac{2\lambda_S}{2\lambda_S + \theta} - p_S. \quad \text{(A.5b)}$$

For (A.5b), recall from Section 2.3.1 that the match probability of a shared request is $2\lambda_S/(2\lambda_S + \theta)$ where the conversions are given by:

$$\lambda_S = \mathbb{P}(u_S > [u_X]^+)$$

$$
\begin{aligned}
&= \mathbb{P}\left(v - \beta\frac{2\lambda_S}{2\lambda_S+\theta} - p_S > [v-p_X]^+\right) \\
&= \begin{cases} 1 - \dfrac{2\beta\lambda_S}{(2\lambda_S+\theta)} - p_S, & \text{if } p_X > \dfrac{2\beta\lambda_S}{(2\lambda_S+\theta)} + p_S \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{A.6}
$$

$$
\begin{aligned}
\lambda_X &= \mathbb{P}(u_X \geq [u_S]^+) \\
&= \mathbb{P}\left(v - p_I \geq \left[v - \beta\frac{2\lambda_S}{2\lambda_S+\theta} - p_S\right]^+\right) \\
&= \begin{cases} 1 - p_X, & \text{if } p_X \leq \dfrac{2\beta\lambda_S}{(2\lambda_S+\theta)} + p_S \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
$$

Equation (A.6) implies that λ_S is the solution of a quadratic equation given $p_X > 2\beta\lambda_S/(2\lambda_S+\theta) + p_S$.

$$
\begin{aligned}
& 2\lambda_S^2 + (\theta + 2\beta - 2(1-p_S))\lambda_S - \theta(1-p_S) = 0 \\
\implies \quad & \lambda_S = \frac{1}{4}(2 - 2\beta - 2p_S - \theta) + \frac{1}{2}\sqrt{(1-p_S-\beta)^2 + (1-p_S+\beta)\theta + \theta^2/4}. \\
& \qquad \text{(the other root is negative; when } \beta = 0 \text{ it is reduced to } \lambda_S = 1 - p_S)
\end{aligned}
$$

We let $\tilde{\lambda}_S := (2 - 2\beta - 2p_S - \theta)/4 + \sqrt{(1-p_S-\beta)^2 + (1-p_S+\beta)\theta + \theta^2/4}/2$. It can be shown that $\tilde{\lambda}_S \in (0,1)$ for all $p_S \in (0,1), \beta > 0, \theta > 0$. Also, $\tilde{\lambda}_S$ decreases with p_S for any $\beta, \theta > 0$. Equation (A.6) can thus be written as:

$$
\lambda_S = \tilde{\lambda}_S \mathbb{1}_{p_X > 1-\tilde{\lambda}_S}, \quad \lambda_X = (1-p_X)\mathbb{1}_{p_X \leq 1-\tilde{\lambda}_S}.
$$

The optimal static prices solve

$$
\begin{aligned}
\max_{p_X, p_S} \quad & \lambda_S\left(p_S - \frac{c}{2}\Big(1 + \frac{\theta}{2\lambda_S+\theta}\Big)\right) + \lambda_X(p_X - c) \\
= & \tilde{\lambda}_S\left(p_S - \frac{c}{2}\Big(1 + \frac{\theta}{2\tilde{\lambda}_S+\theta}\Big)\right)\mathbb{1}_{p_X > 1-\tilde{\lambda}_S} + (1-p_X)(p_X - c)\mathbb{1}_{p_X \leq 1-\tilde{\lambda}_S}
\end{aligned}
$$

Similarly, all conversion decisions do not depend on the rider's valuation v. However, because of the complexity of λ_S, we are not able to get a closed-form solution for the optimal static price, nor can we analytically compare it to the optimal match-based prices. Instead, we present exact numerical results.

Figure A.1 reproduces Figure 2.5, now with $\beta = 0.1$, and compares the optimal static and match-based conversion rates. In the striped area where $c \leq 2\beta$, it is optimal to operate solo rides only, regardless of whether shared rides utilize static pricing or match-based pricing. Meanwhile, when $c > 2\beta$, it is better to operate shared rides only, as long as they are profitable. Otherwise, the relationship between the optimal match-based and static

conversion rates is consistent with Figure 2.5. (Note that when $\beta = 0$, comparing conversion rates is equivalent to comparing prices. For example, $\lambda^* = \bar{\lambda}^*$ is equivalent to $p^* = \bar{p}^*$. This is no longer true when $\beta > 0$; nevertheless, the trends and qualitative observations for the price comparisons are similar to the conversion comparisons in Figure A.1.) Similarly, Figure A.2 reproduces Figure 2.6 for $\beta = 0.1$. As before, match-based pricing has a larger profitablility region, lower average payment, and lower cost per request than static pricing. The improvement in payment and cost per request is especially pronounced for high c and θ, when shared rides are most challenging to operate.

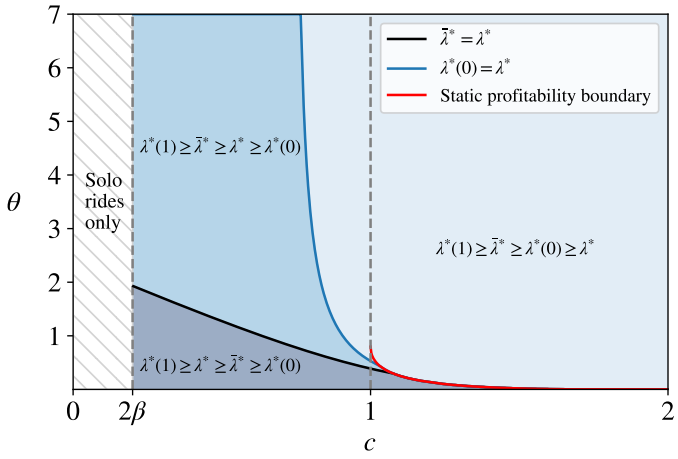


Figure A.1: Comparisons between optimal match-based conversions and static conversions with $\beta = 0.1$.

A.1.1 Rider and Platform Risk

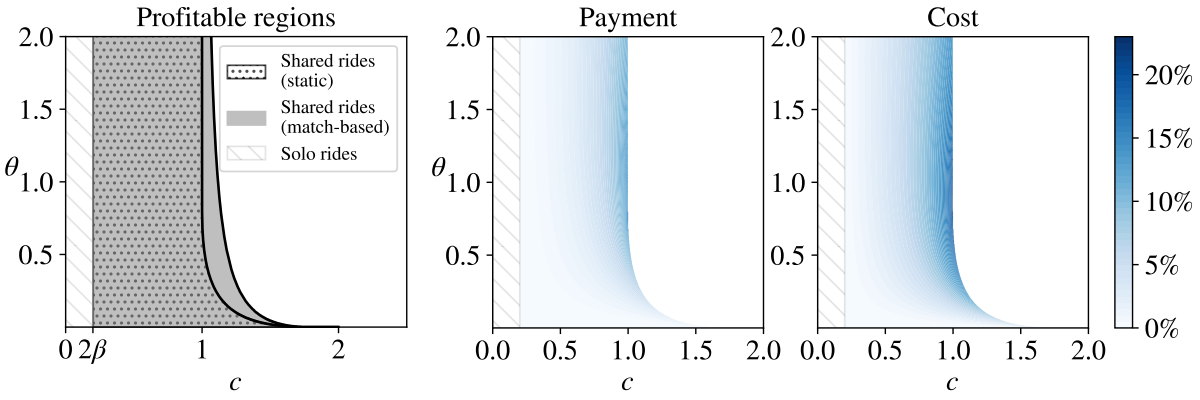
We start by providing a discussion of risk for platform and riders to illustrate the trade-offs between various pricing policies. We quantify the risk for the platform as the *variance in profit per request*. Under static pricing, the platform's profit per request is

$$\text{Profit/Request (Static)} = \begin{cases} 1 - \lambda^* - c/2, & \text{with probability } \rho(\lambda^*), \\ 1 - \lambda^* - c, & \text{with probability } 1 - \rho(\lambda^*), \end{cases}$$

where $\rho(\lambda^*) = 2\lambda^*/(2\lambda^* + \theta)$ is the match rate under optimal static conversion rate λ^*. Then, the platform's risk under static pricing is

$$\text{Platform Risk (Static)} = \frac{\lambda^* \theta}{2} \left(\frac{c}{2\lambda^* + \theta} \right)^2. \tag{A.7}$$

From the rider's perspective, we quantify her risk as the *variance in payment*. As static



(a) Profitable regions of policies

(b) Percentage decrement of payment and cost per request

Figure A.2: Profitable regions of different pricing policies ($\beta = 0.1$); percentage decrement of payment and cost per request (match-based pricing over static pricing)

pricing always quotes the same price $1 - \lambda^*$ to all riders,

$$\text{Rider Risk (Static)} = 0. \tag{A.8}$$

Therefore, a single static price is risky for the platform, but not for the rider.

However, by charging *two* prices, the platform can transfer some of its risk to the rider. To illustrate this point, we consider a *state-independent* variant of match-based pricing. The key difference between this state-independent variant and the one described in the main text is that under this variant, the platform quotes two prices to all riders regardless of system state: the price is $1 - \lambda'(0)$ if the rider is dispatched alone, and $1 - \lambda(1)$ if the rider is matched with another rider. This conversion behavior resembles the user interface in Figure 2.4a. Because the quoted prices are state-independent, the conversion is as well, so we write the conversion simply as λ.

The platform's expected profit per unit time is then

$$\Pi(\lambda'(0), \lambda(1)) = \frac{\lambda}{2\lambda + \theta} \left(\lambda(2 - 2\lambda(1) - c) + \theta(1 - \lambda'(0) - c)\right). \tag{A.9}$$

We assume that riders are risk-neutral, so that they convert based on the expected price

$$1 - \lambda = (1 - \lambda(1))\rho(\lambda) + (1 - \lambda'(0))(1 - \rho(\lambda)), \tag{A.10}$$

where $\rho = 2\lambda/(2\lambda + \theta)$ is the match rate. Using (A.10), the profit function (A.9) simplifies to

$$\Pi(\lambda) = \frac{\lambda}{2\lambda + \theta} \left(\lambda(2 - 2\lambda - c) + \theta(1 - \lambda - c)\right),$$

which equals the objective function of (2.2). Hence, this state-independent variant of match-based pricing achieves the same profit as static pricing. Setting $\lambda = \lambda^*$ (the optimal static conversion) and rearranging (A.10), the equation

$$\lambda'(0) = \lambda^* \left(1 - \frac{2}{\theta}(\lambda(1) - \lambda^*)\right) \tag{A.11}$$

defines a set of values for $\lambda'(0)$ and $\lambda(1)$ that all achieve the optimal static profit $\Pi(\lambda^*)$.

Although the $\lambda'(0)$ and $\lambda(1)$ satisfying (A.11) all achieve the same profit, they vary in their balance of platform versus rider risk. Concretely, under state-independent match-based pricing, and given conversion rates $\lambda'(0)$ and $\lambda(1)$ satisfying (A.11), the platform's profit per request is

$$\text{Profit/Request (Match, State-Ind.)} = \begin{cases} 1 - \lambda(1) - c/2, & \text{with probability } \rho(\lambda^*), \\ 1 - \lambda'(0) - c, & \text{with probability } 1 - \rho(\lambda^*). \end{cases}$$

Then, the platform's risk under state-independent match-based pricing is

$$\text{Platform Risk (Match, State-Ind.)} = \frac{\lambda^* \theta}{2} \left(\frac{c - 2\left(\lambda(1) - \lambda'(0)\right)}{2\lambda^* + \theta}\right)^2. \tag{A.12}$$

Riders arriving on the platform perceive their future payment (if they convert) as a random variable

$$\text{Payment (Match, State-Ind.)} = \begin{cases} 1 - \lambda(1), & \text{with probability } \rho(\lambda^*), \\ 1 - \lambda'(0), & \text{with probability } 1 - \rho(\lambda^*), \end{cases}$$

so the rider's risk is

$$\text{Rider Risk (Match, State-Ind.)} = \frac{\lambda^* \theta}{2} \left(\frac{2(\lambda(1) - \lambda'(0))}{2\lambda^* + \theta}\right)^2. \tag{A.13}$$

The expressions (A.12) and (A.13) show that by charging two prices, the platform can strike a trade-off between risk to itself and riders. Noting from (A.11) that $\lambda(1) \geq \lambda^* \geq \lambda'(0)$, there are three cases.

- When $\lambda(1) - \lambda'(0) = 0$, the platform offers static prices. Consistent with (A.7) and (A.8), the platform risk is $\lambda^*(\theta/2) \cdot (c/(2\lambda^* + \theta))^2$ and the rider risk is zero.

- As $\lambda(1) - \lambda'(0)$ increases to $c/2$, the platform risk decreases to zero, while rider risk increases to $\lambda^*(\theta/2) \cdot (c/(2\lambda^* + \theta))^2$.

- The platform would not increase $\lambda(1) - \lambda'(0)$ beyond $c/2$, as this increases risk for both platform and rider alike.

For illustration, platform and rider risks under for various pricing policies are plotted in Figure A.3 for $c = 0.8$ and $\theta = 0.5, 1, 2$ (the plots are similar for other values of c and θ).

The state-independent match-based prices satisfying (A.11) define a Pareto frontier (blue line) of platform versus rider risk. Since static pricing (red point) is a special case of state-independent match-based pricing, it lies directly on the Pareto frontier, where the highest platform risk is incurred. Thus, although the state-independent variant of match-based pricing does not increase profit relative to static pricing, the flexibility of two prices allows the platform to transfer its risk to riders. However, this risk transfer could be unappealing to riders.

We now discuss the risk aspect of the proposed *state-dependent* match-based pricing in Section 2.3.1. (As this is the focus of our paper, we will refer to it simply as match-based pricing thereafter unless clarification is needed.) Recall that, in contrast with the state-independent variant, the platform quotes the two prices $1 - \lambda'^*(0)$ and $1 - \lambda^*(1)$ in state 0, when there is no rider waiting on the platform; it quotes the price $1 - \lambda^*(1)$ in state 1, when another rider is waiting on the platform. Then, in state 0, riders convert at rate $\lambda^*(0)$, which satisfies (2.4); in state 1, riders convert at rate $\lambda^*(1)$. Because riders are shown a single, lower price in state 1, their conversion increases and enables higher platform profit.

Under match-based pricing, the platform's profit per request is

$$\text{Profit/Request (Match)} = \begin{cases} 1 - \lambda^*(1) - c/2, & \text{with probability } 2\lambda^*(1)/(2\lambda^*(1) + \theta), \\ 1 - \lambda'^*(0) - c, & \text{with probability } \theta/(2\lambda^*(1) + \theta), \end{cases}$$

where $2\lambda^*(1)/(2\lambda^*(1) + \theta)$ is the match rate under optimal match-based prices. Applying Theorem 3 and eq. (2.5), the platform's risk under match-based pricing is then

$$\text{Platform Risk (Match)} = \frac{\theta}{8} \frac{1}{(2 - c + 2\theta)^3} \left(\frac{(2-c)^2 + 4\theta(1-c)}{(2-c)^2 + 2\theta(4 - c + 2\theta)} \right)^2. \tag{A.14}$$

Similarly, riders arriving on the platform perceive their future payment (if they convert) as

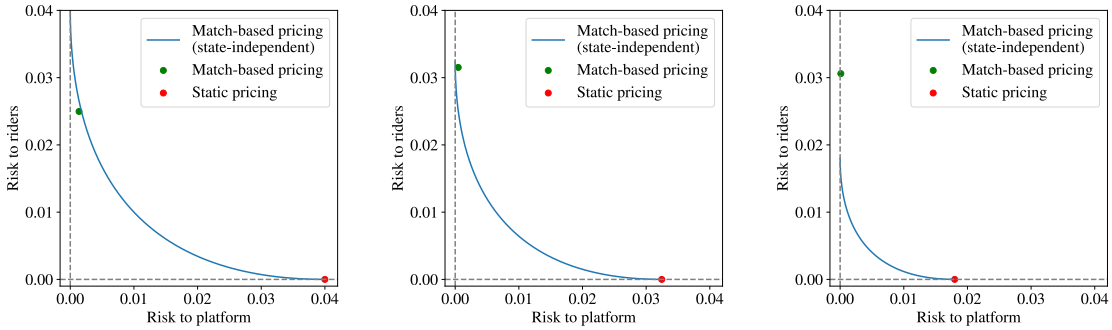
$$\text{Payment (Match)} = \begin{cases} 1 - \lambda^*(1), & \text{with probability } 2\lambda^*(1)/(2\lambda^*(1) + \theta), \\ 1 - \lambda'^*(0), & \text{with probability } \theta/(2\lambda^*(1) + \theta), \end{cases}$$

so the rider's risk is

$$\text{Rider Risk (Match)} = \frac{c^2\theta}{8} \frac{(2-c)^2 + 4\theta}{(2 - c + 2\theta)^3} \left(\frac{(2-c)^2 + 4\theta(3 - c + 2\theta)}{(2-c)^2 + 2\theta(4 - c + 2\theta)} \right)^2. \tag{A.15}$$

The quantities from (A.14) and (A.15) are plotted in Figure A.3 (green points) under the same set of parameters, showing that state-dependent match-based pricing tends to trade platform risk for rider risk. In all cases, match-based pricing lies close to the Pareto frontier, and notably in one case (Figure A.3a), it *Pareto dominates* the static and state-independent variant of the match-based pricing in terms of both risks and expected profit.

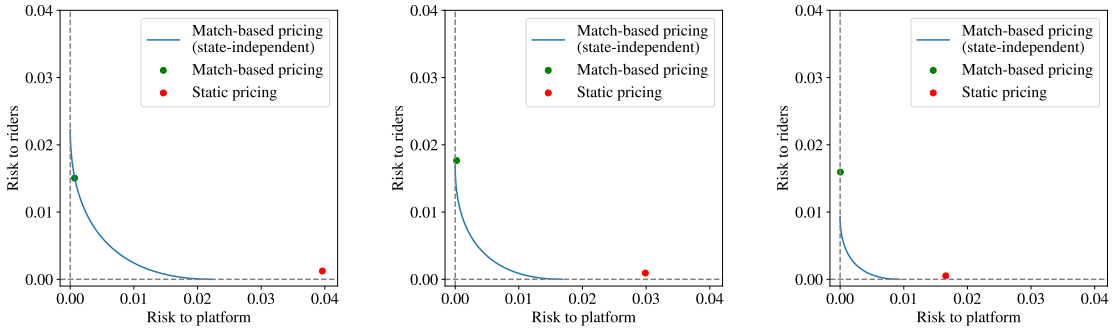
As with Appendix A.1, we can repeat this analysis for the case with a sharing disutility $\beta > 0$; now the risk is quantified as the variance in disutility. As the analysis is similar to the case for $\beta = 0$, it is omitted for brevity. The main difference is that static pricing no longer has zero risk for the rider, as a rider's utility changes depending on whether she is



(a) $\theta = 0.5$ (b) $\theta = 1$ (c) $\theta = 2$

Figure A.3: Risks under different pricing policies for $c = 0.8$ and varying θ.

matched — in fact, static pricing now lies outside of the Pareto frontier, as shown in Figure A.4. Intuitively, when riders experience disutility from being matched with another rider, static pricing makes riders prefer to remain unmatched. By offering two prices, the platform can compensate the riders experiencing sharing disutility.



(a) $\theta = 0.5$ (b) $\theta = 1$ (c) $\theta = 2$

Figure A.4: Risks under different pricing policies for $c = 0.8$, sharing disutility $\beta = 0.1$ and varying θ.

In summary, offering two prices allows the platform to decrease its risk while also increasing profits under (state-dependent) match-based pricing. Riders often shoulder the increase in risk. However, match-based pricing also allows the platform to drop prices in state 1. Riders then benefit from lower average payments, as well as lower average quoted prices if c and θ are high enough (Proposition 2), which can compensate for the increase in risk.

A.2 Computing the Static Pricing Policy

We give a mathematical program whose optimal value upper bounds the optimal profit rate under the optimal static pricing and matching policy. Following the notion used in [56] and [9], a rider is called "passive" if it is matched with a rider arriving earlier; otherwise, it is called "active". With a bit of abuse of notation, let $x_{i,j}$ be the average rate of dispatches between type i active riders with type j passive riders. Denote the average rate of solo dispatch of type i riders by y_i. Let $\mathbf{x} := [x_{i,j}]_{i,j\in[N]}$ and $\mathbf{y} := [y_i]_{i\in[N]}$. Assume the arrival rate of rider type i is Λ_i for $i \in [N]$, and the request conversion rate is $\lambda_i \in [0, \Lambda_i]$. Let $\boldsymbol{\lambda} := [\lambda_i]_{i\in[N]}$ be a vector of static request conversion rates of all rider types. Let ρ_i denote the match rate of an incoming rider type i, and it can be written as the closed form below given the rate variables:

$$\rho_i := (1 - y_i/\lambda_i). \tag{A.16}$$

Let $\mathbb{E}(t_i)$ be the expected detour rate for rider type i given that it is matched. Recall that t_{ij} is the detour rate of rider type i when matched with rider type j. $\mathbb{E}(t_i)$ can also be written as the closed form below:

$$\mathbb{E}(t_i) := \sum_{j=1}^{N}(x_{i,j} + x_{j,i})t_{ij}/(\lambda_i - y_i). \tag{A.17}$$

Rider type i has an expected sharing and detour disutility $\bar{u} = \rho_i(\beta_0 + \beta_1\mathbb{E}(t_i))$, and the static price $p_i(\lambda_i)$ is then given by

$$p_i(\lambda_i) = (1 - \bar{u} - \lambda_i/\Lambda_i)\ell_i, \tag{A.18}$$

where we assume uniform distribution of willingness to pay. Based on (A.16), (A.17) and (A.18), the overall profit given conversion and matching rates is then expressed as

$$\begin{aligned}
&\left(\sum_{i=1}^{N}\lambda_i p_i(\lambda_i) - \sum_{i=1}^{N}\sum_{j=1}^{N} c_{(i,j)}x_{i,j} - \sum_{i=1}^{N} c_{(i)}y_i\right) \\
&= \sum_{i=1}^{N}\lambda_i\Big(1 - \beta_0 - \lambda_i/\Lambda_i\Big)\ell_i - \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\Big(\beta_1 t_{ij}(\ell_i + \ell_j) + c\ell_{ij}\Big)x_{i,j} - \sum_{i=1}^{N}(c - \beta_0)\ell_i y_i\right).
\end{aligned} \tag{A.19}$$

Consider the following linear program $\mathsf{PB}(\boldsymbol{\lambda})$ parameterized by a vector of static request conversion rates of all rider types and denote its optimal value by $g(\boldsymbol{\lambda})$.

$$\begin{aligned}
&(\mathsf{PB}(\boldsymbol{\lambda})) \\
&g(\boldsymbol{\lambda}) := \max_{x,\,y} \qquad \sum_{i=1}^{N}\lambda_i\Big(1 - \beta_0 - \lambda_i/\Lambda_i\Big)\ell_i
\end{aligned}$$

$$
\begin{aligned}
& - \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \Big(\beta_1 t_{ij} (\ell_i + \ell_j) + c\ell_{ij} \Big) x_{i,j} - \sum_{i=1}^{N} (c - \beta_0) \ell_i y_i \right) \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{j,i} + \sum_{j=1}^{N} x_{i,j} + y_i = \lambda_i, \qquad \forall i \in [N], \qquad (\gamma_i) \\
& \theta_i x_{i,j} \le \lambda_j y_i, \qquad \forall i, j \in [N], \qquad (\eta_{i,j}) \\
& x_{i,j} \ge 0, \qquad \forall i, j \in [N], \\
& y_i \ge 0, \qquad \forall i \in [N].
\end{aligned}
$$

Corollary 1. *Let $\Pi(\boldsymbol{\lambda})$ be the maximum expected profit under static request conversion rates $\boldsymbol{\lambda}$ and the corresponding optimal matching policy. The following holds:*

$$\max_{\mathbf{0} \le \boldsymbol{\lambda} \le \boldsymbol{\Lambda}} g(\boldsymbol{\lambda}) \ge \max_{\mathbf{0} \le \boldsymbol{\lambda} \le \boldsymbol{\Lambda}} \Pi(\boldsymbol{\lambda}).$$

Proof. The proof follows from a straightforward adaptation of Lemma 1 of [9] where they show $g(\boldsymbol{\lambda}) \ge \Pi(\boldsymbol{\lambda})$ for all $\boldsymbol{\lambda} \in [\mathbf{0}, \boldsymbol{\Lambda}]$. Essentially, given conversions, one can show that any feasible matching policy induces average matching rates that are feasible solutions of $\mathsf{PB}(\boldsymbol{\lambda})$. Moreover, its objective value coincides with the expected profit of the matching policy. □

We solve the problem $\max_{\mathbf{0} \le \boldsymbol{\lambda} \le \boldsymbol{\Lambda}} g(\boldsymbol{\lambda})$ using a projected subgradient ascent approach where the subgradient of $g(\boldsymbol{\lambda})$ can be obtained using the optimal dual value of the linear program $\mathsf{PB}(\boldsymbol{\lambda})$ under certain regularity conditions (see e.g., [77, Corollary 5]). Let γ_i^* and $\eta_{i,j}^*$ be the corresponding optimal dual solutions of the first and second sets of constraints of $\mathsf{PB}(\boldsymbol{\lambda})$. Note that projection onto the feasible set $[\mathbf{0}, \boldsymbol{\Lambda}]$ can be done with a closed form. Let y_i^* be the optimal primal solution of $\mathsf{PB}(\boldsymbol{\lambda})$. We have the following subgradient by standard linear programming duality:

$$\left[\frac{\partial \lambda_i (1 - \beta_0 - \lambda_i / \Lambda_i) \ell_i}{\partial \lambda_i} + \gamma_i^* + \sum_{j \in [N]} y_j^* \eta_{j,i}^* \right]_{i \in [N]} \in \partial g(\boldsymbol{\lambda}).$$

We can get a few quantities from the optimal solution of the problem $\max_{\mathbf{0} \le \boldsymbol{\lambda} \le \boldsymbol{\Lambda}} g(\boldsymbol{\lambda})$ that are used in our computational experiments in Section 2.4. For a type i request who is let wait upon arrival, her match rate ρ_i and expected detour rate given matched $\mathbb{E}(t_i^0)$ are:

$$\rho_i = \frac{\sum_{j=1}^{N} x_{i,j}}{\sum_{j=1}^{N} x_{i,j} + y_i}, \quad \mathbb{E}(t_i^0) = \frac{\sum_{j=1}^{N} x_{i,j} t_{ij}}{\sum_{j=1}^{N} x_{i,j}}, \quad \forall i \in [N].$$

Similarly, for a type i request who gets matched immediately upon arrival, her match rate is 1, and her average detour rate given matched $\mathbb{E}(t_i^1)$ is:

$$\mathbb{E}(t_i^1) = \frac{\sum_{j=1}^{N} x_{j,i} t_{ij}}{\sum_{j=1}^{N} x_{j,i}}, \quad \forall i \in [N].$$

These quantities are used to initialize the demand functions in ALP.

In Section A.3 we provide omitted proofs. In Section A.5 we give a few auxiliary results that are mentioned in the main text.

A.3 Proofs

Proof of Proposition 1. From eq. (2.2), the first- and second-order derivatives of $\Pi(\lambda)$ are:

$$\Pi'(\lambda) = -2\lambda - \frac{c\theta^2}{2(2\lambda+\theta)^2} + \left(1 - \frac{c}{2}\right), \tag{A.21a}$$

$$\Pi''(\lambda) = -2 + \frac{2c\theta^2}{(2\lambda+\theta)^3}. \tag{A.21b}$$

Using (A.21b), there is an inflection point $\hat{\lambda} = (\sqrt[3]{c\theta^2} - \theta)/2$. When $\theta \geq c$, then $\hat{\lambda} \leq 0$. Since $\Pi''(\lambda)$ is decreasing, it follows that $\Pi''(\lambda) \leq \Pi''(\hat{\lambda}) = 0$ for $\lambda \geq 0$, so $\Pi(\lambda)$ is concave. There are two cases.

- If $c < 1$, then setting (A.21a) equal to zero and applying the fact $\theta^2/(2\lambda+\theta)^2 \in [0,1]$, we find that there is one critical point $\tilde{\lambda} \in [1/2 - c/2, 1/2 - c/4] \subset [0,1]$. Therefore, $\lambda^* = \tilde{\lambda} > 0$.

- If $c \geq 1$, then for $\lambda \in [0,1]$, we have $\Pi'(\lambda) \leq \Pi'(0) = 1 - c \leq 0$. Therefore, $\lambda^* = 0$.

When $\theta < c$, then the inflection point $\hat{\lambda} > 0$, possibly allowing $\Pi(\lambda)$ to have another critical point. The gradient $\Pi'(\lambda)$ takes the following values at its inflection point ($\hat{\lambda}$) and boundary points (0 and 1):

$$\Pi'(0) = 1 - c, \quad \Pi'(\hat{\lambda}) = \frac{2 - c - 3\sqrt[3]{c\theta^2} + 2\theta}{2}, \quad \Pi'(1) = -\frac{c\theta^2}{2(2+\theta)^2} - 1 - \frac{c}{2} < 0.$$

From these values, we have the following cases.

1. If $c \leq 1$, then $\Pi(\lambda)$ has one critical point in $(0,1)$, which we denote $\tilde{\lambda}$. Since $\Pi'(\lambda) \geq 0$ for all $\lambda \in [0, \tilde{\lambda}]$ and $\Pi'(\lambda) \leq 0$ for all $\lambda \in [\tilde{\lambda}, 1]$, then $\lambda^* = \tilde{\lambda} > 0$. (Note that unlike the case $\theta \geq c$, $\Pi(\lambda)$ is not concave.)

2. If $c > 1$ and $\Pi'(\hat{\lambda}) \leq 0$, then $\Pi'(\lambda) \leq 0$ for all $\lambda \in [0,1]$, and $\lambda^* = \Pi(0) = 0$.

3. If $c > 1$ and $\Pi'(\hat{\lambda}) > 0$, then $\Pi(\lambda)$ has two critical points: it decreases from 0 to a local minimum, then increases to a local maximum. We use $\tilde{\lambda}$ to denote the local maximum; note that $\tilde{\lambda} > 0$.

When $c > 1$ and $\Pi'(\hat{\lambda}) > 0$, then $\lambda^* = \tilde{\lambda}$ if $\Pi(\tilde{\lambda}) > 0$, and $\lambda^* = 0$ otherwise. To identify the conditions when $\lambda^* = \tilde{\lambda}$ and when $\lambda^* = 0$, we parametrize problem (2.2)

by c and θ. The optimal solution $\lambda^*(c,\theta)$ is discontinuous at the point $(c,\tilde{\theta}(c))$ where $\Pi(\tilde{\lambda}(c,\tilde{\theta}(c))) = \Pi(0) = 0$, or:

$$-(\tilde{\lambda}(c,\tilde{\theta}(c))^2 + \left(1-\frac{c}{2}\right)\tilde{\lambda}(c,\tilde{\theta}(c)) - \frac{c\tilde{\theta}(c)\tilde{\lambda}(c,\tilde{\theta}(c))}{2(2\tilde{\lambda}(c,\tilde{\theta}(c)) + \tilde{\theta}(c))} = 0. \tag{A.22}$$

Equation (A.22) has two roots:

$$\tilde{\lambda}(c,\tilde{\theta}(c)) = \frac{1}{4}\left(2 - c - \tilde{\theta}(c) \pm \sqrt{4 - 4c + c^2 + 4\tilde{\theta}(c) - 6c\tilde{\theta}(c) + \tilde{\theta}(c)^2}\right). \tag{A.23}$$

In addition, equation (A.22) can be rearranged as:

$$2\tilde{\lambda}(c,\tilde{\theta}(c)) + \tilde{\theta}(c) = \frac{c\tilde{\theta}(c)}{2 - c - 2\tilde{\lambda}(c,\tilde{\theta}(c))}.$$

Plugging this into the first-order condition for $\Pi(\lambda)$, we have:

$$\Pi'(\tilde{\lambda}(c,\tilde{\theta}(c))) = -2\tilde{\lambda}(c,\tilde{\theta}(c)) - \frac{c\tilde{\theta}(c)^2}{2\left(c\tilde{\theta}(c)/(2 - c - 2\tilde{\lambda}(c,\tilde{\theta}(c)))\right)^2} + \left(1-\frac{c}{2}\right) = 0,$$

which reduces to:

$$\tilde{\lambda}(c,\tilde{\theta}(c)) = 1 - c + \sqrt{\frac{c(c-1)}{2}}. \tag{A.24}$$

Equation (A.24) implies that the roots (A.23) are in fact the following single root:

$$\tilde{\lambda}(c,\tilde{\theta}(c)) = \frac{1}{4}(2 - c - \tilde{\theta}(c)). \tag{A.25}$$

Combining equations (A.24) and (A.25), we find that:

$$\tilde{\theta}(c) = 3c - 2 - 2\sqrt{2c(c-1)}.$$

Having found the point $\tilde{\theta}(c)$ at which $\Pi(\tilde{\lambda}(c,\tilde{\theta}(c))) = \Pi(0)$, it follows that $\lambda^*(c,\tilde{\theta}(c))$ can be either 0 or $\tilde{\lambda}(c,\tilde{\theta}(c))$; we arbitrarily choose $\lambda^*(c,\tilde{\theta}(c)) = 0$. Now, we determine the values of $\lambda^*(c,\theta)$ for $\theta \neq \tilde{\theta}(c)$. When $\theta < \tilde{\theta}(c)$, we have:

$$\begin{aligned}\Pi(\lambda^*(c,\theta)) &= -(\lambda^*(c,\theta))^2 + \left(1-\frac{c}{2}\right)\lambda^*(c,\theta) - \left[\frac{c\theta}{2}\frac{\lambda^*(c,\theta)}{2\lambda^*(c,\theta)+\theta}\right] \\ &> -(\lambda^*(c,\tilde{\theta}(c)))^2 + \left(1-\frac{c}{2}\right)\lambda^*(c,\tilde{\theta}(c)) - \left[\frac{c\theta}{2}\frac{\lambda^*(c,\tilde{\theta}(c))}{2\lambda^*(c,\tilde{\theta}(c))+\theta}\right] \\ &\qquad (\lambda^*(c,\tilde{\theta}(c)) \text{ is suboptimal for } \theta \neq \tilde{\theta}(c))\end{aligned}$$

$$> -(\lambda^*(c,\tilde{\theta}(c)))^2 + \left(1-\frac{c}{2}\right)\lambda^*(c,\tilde{\theta}(c)) - \left[\frac{c\tilde{\theta}(c)}{2}\frac{\lambda^*(c,\tilde{\theta}(c))}{2\lambda^*(c,\tilde{\theta}(c))+\tilde{\theta}(c)}\right]$$

(since $\theta < \tilde{\theta}(c)$)

$$= \Pi(0) = 0. \tag{A.22}$$

Since $\Pi(\lambda^*(c,\theta)) > 0$, then $\lambda^*(c,\theta) = \tilde{\lambda}(c,\theta)$.

When $\theta > \tilde{\theta}(c)$, we aim to show that $\Pi(\lambda^*(c,\theta)) < 0 = \Pi(0)$. Following (A.23), this is equivalent to:

$$\lambda^*(c,\theta) \geq \frac{1}{4}\left(2-c-\theta+\sqrt{4-4c+c^2+4\theta-6c\theta+\theta^2}\right), \text{ or} \tag{A.26a}$$

$$\lambda^*(c,\theta) \leq \frac{1}{4}\left(2-c-\theta-\sqrt{4-4c+c^2+4\theta-6c\theta+\theta^2}\right) \tag{A.26b}$$

when $4-4c+c^2+4\theta-6c\theta+\theta^2 \geq 0$, or

$$4-4c+c^2+4\theta-6c\theta+\theta^2 < 0. \tag{A.27}$$

For $\theta > \tilde{\theta}(c)$, there are two cases. First, if $\theta \in (\tilde{\theta}(c), 3c-2+2\sqrt{2c(c-1)})$, then (A.27) is satisfied. Second, if $\theta \geq 3c-2+2\sqrt{2c(c-1)}$, then the right-hand side of (A.26a) is negative. Therefore, for any $\theta > \tilde{\theta}(c)$, we have $\Pi(\lambda^*(c,\theta)) < 0 = \Pi(0)$, so $\lambda^*(c,\theta) = 0$.

□

Proof of Theorem 1. From Algorithm 1 and eq. (2.1), the iterative equation of $\lambda^{(k)}$ is:

$$\lambda^{(k+1)} = \left[\frac{1-\bar{c}^{(k)}}{2}\right]^+ = \left[\frac{1}{2}\left(1-\frac{c}{2}-\frac{c\theta}{2(2\lambda^{(k)}+\theta)}\right)\right]^+. \tag{A.28}$$

Since Algorithm 1 returns 0 if $\lambda^{(k+1)} < 0$, we can ignore the $[\ ]^+$ when analyzing (A.28), only enforcing nonnegativity at termination. The equilibrium condition is then:

$$Q(\lambda^{(k)}) = (\lambda^{(k)})^2 + \frac{2\theta+c-2}{4}\lambda^{(k)} + \frac{\theta(c-1)}{4} = 0. \tag{A.29}$$

The roots of (A.29) are:

$$\lambda^+, \lambda^- = \frac{1}{4}\left(1-\theta-\frac{c}{2} \pm \sqrt{\left(1+\theta-\frac{3c}{2}\right)^2 - 2c(c-1)}\right).$$

The convergence of $\lambda^{(k)}$ depends on the location of $\lambda^{(1)} \in (0,1]$ relative to these roots. We make use of the following lemmas.

Lemma 2. *Let r denote λ^+ or λ^-. If $\lambda^{(1)} < r$, then $\lambda^{(k)} < r$ for all $k \geq 1$. If $\lambda^{(1)} > r$, then $\lambda^{(k)} > r$ for all $k \geq 1$.*

Lemma 3. *If $Q(\lambda^{(k)}) > 0$, then $\lambda^{(k+1)} < \lambda^{(k)}$. If $Q(\lambda^{(k)}) < 0$, then $\lambda^{(k+1)} > \lambda^{(k)}$.*

Lemma 2 is proved by induction, and Lemma 3 is straightforward algebra. Both proofs are provided later in the e-companion.

As $\lambda^{(1)}$ is already at equilibrium at λ^+ or λ^-, we focus on the remainder of $(0, 1]$.

- First, consider $\lambda^{(1)} > \lambda^+$. By Lemma 2, $\lambda^{(k)} > \lambda^+$ for $k \geq 1$. Then, $Q(\lambda^{(k)}) > 0 \; \forall k \geq 1$, and $\lambda^{(k)}$ is monotone decreasing by Lemma 3. Therefore, $\lambda^{(k)}$ converges to $\max\{\lambda^+, 0\}$.

- Next, consider $\lambda^{(1)} \in (\lambda^-, \lambda^+)$, which requires $\lambda^+ > 0$ and $\lambda^+ \neq \lambda^-$. By Lemma 2, $\lambda^{(k)} \in (\lambda^-, \lambda^+) \; \forall k \geq 1$. Then, $Q(\lambda^{(k)}) < 0 \; \forall k \geq 1$, and $\lambda^{(k)}$ is monotone increasing by Lemma 3, converging to λ^+.

- Next, consider $\lambda^{(1)} < \lambda^-$, which requires $\lambda^- > 0$. By Lemma 2, $\lambda^{(k)} < \lambda^- \; \forall k \geq 1$. Then, $Q(\lambda^{(k)}) > 0 \; \forall k \geq 1$, and $\lambda^{(k)}$ is monotone decreasing by Lemma 3, converging to 0. Finally, consider imaginary λ^+ and λ^-. Then, $Q(\lambda) > 0$ everywhere. By Lemma 3, $\lambda^{(k)}$ is monotone decreasing and converges to 0.

We now determine the conditions on c and θ that lead to the above scenarios.

- First, if $c \in (0, 1)$ and $\theta \in (0, \infty)$, then $\lambda^- < 0 < \lambda^+ < 1$. Then, for $\lambda^{(1)} \in (0, 1]$, $\lambda^{(k)}$ converges to λ^+.

- Next, if $c \in [1, 2)$ and $\theta \in (0, \tilde{\theta}(c)/2]$, then $0 \leq \lambda^- \leq \lambda^+ < 1$, dividing the domain into the intervals $(0, \lambda^-)$ (if $\lambda^- > 0$), $[\lambda^-, \lambda^+)$ (if $\theta \neq \tilde{\theta}(c)/2$), and $[\lambda^+, 1]$. If $\lambda^{(1)} \in (0, \lambda^-)$, then $\lambda^{(k)}$ converges to 0. If $\lambda^{(1)} = \lambda^-$, then $\lambda^{(k)}$ remains at λ^-. Otherwise, $\lambda^{(k)}$ converges to λ^+.

- Finally, if $c \in [1, 2)$ and $\theta \in (\tilde{\theta}(c)/2, \infty)$, then (A.29) does not have any positive, real roots. For $\lambda^{(1)} \in (0, 1]$, $\lambda^{(k)}$ converges to 0.

□

Proof of Lemma 2. First, consider $\lambda^{(1)} < r$. We prove that $\lambda^{(k)} < r$ for all $k \geq 1$ by induction. The base case is given. Now assume that $\lambda^{(k)} < r$ for some k. Then, for $\lambda^{(k+1)}$, we have:

$$\lambda^{(k+1)} = \frac{1}{2}\left(1 - \frac{c}{2} - \frac{c\theta}{2(2\lambda^{(k)} + \theta)}\right) < \frac{1}{2}\left(1 - \frac{c}{2} - \frac{c\theta}{2(2r + \theta)}\right) = r,$$

where the first step is from the iterative equation (A.28), the second step is from the inductive hypothesis, and the third step is by definition of the root r. Therefore, $\lambda^{(k)} < r$ for all $k \geq 1$.

The proof is similar for $\lambda^{(1)} > r$. □

Proof of Lemma 3. First, consider $Q(\lambda^{(k)}) > 0$. That is,

$$(\lambda^{(k)})^2 + \frac{2\theta + c - 2}{4}\lambda^{(k)} + \frac{\theta(c-1)}{4} > 0.$$

Dividing both sides by $2\lambda^{(k)} + \theta$, it then follows that:

$$\lambda^{(k)} > \frac{1}{2}\left(1 - \frac{c}{2} - \frac{c\theta}{2(2\lambda^{(k)} + \theta)}\right) = \lambda^{(k+1)}.$$

The proof is similar for $Q(\lambda^{(k)}) < 0$. □

Proof of Theorem 2. For convenience, let $\lambda^I := \lim_{k\to\infty} \lambda^{(k)}$.

From Theorem 1, we know that $\lambda^I \in \{\lambda^+, \lambda^-, 0\}$ and $\lambda^+ \geq \lambda^-$. Since $\lambda^* \geq 0$, it suffices to show that $\lambda^* \geq \lambda^+$ and focus on the conditions when $\lambda^+ > 0$. From Theorem 1, these conditions are: $c \in (0,1), \theta \in (0,\infty)$ or $c \in [1,2), \theta \in (0, \tilde{\theta}(c)/2]$.

To analyze Algorithm 1, we rewrite $\Pi(\lambda)$ as $f(\lambda, \bar{c}(\lambda)) := (p(\lambda) - \bar{c}(\lambda))\lambda$. Since each iteration of Algorithm 1 solves the problem $\max_{\lambda^{(k+1)}\in[0,1]}(1 - \lambda^{(k+1)} - \bar{c}^{(k)})\lambda^{(k+1)}$, the equilibrium condition is:

$$\left.\frac{df(\lambda, \bar{c}^I)}{d\lambda}\right|_{\lambda=\lambda^I} = 0,$$

where $\bar{c}^I := \bar{c}(\lambda^I)$. (Note that this equilibrium condition relies on $\lambda^I > 0$, as the first-order condition might not hold if λ^I is on the boundary. On the other hand, if $\lambda^I = 0$, the statement is trivial to prove). Therefore,

$$\left.\frac{df(\lambda, \bar{c}(\lambda))}{d\lambda}\right|_{\lambda=\lambda^I} = \underbrace{\left.\frac{df(\lambda, \bar{c}^I)}{d\lambda}\right|_{\lambda=\lambda^I}}_{=0} + \left(\underbrace{\frac{\partial f(\lambda, \bar{c}(\lambda))}{\partial \bar{c}}}_{\leq 0}\underbrace{\frac{d\bar{c}(\lambda)}{d\lambda}}_{<0}\right)\Bigg|_{\lambda=\lambda^I, \bar{c}=\bar{c}^I} \geq 0. \tag{A.30}$$

As a result, λ can be increased towards a local maximum. From Proposition 1, we know that under the conditions $c \in (0,1), \theta \in (0,\infty)$ or $c \in [1,2), \theta \in (0, \tilde{\theta}(c)/2]$, there is one local maximum $\tilde{\lambda} > 0$, and it is the global maximum λ^*. Therefore, $\lambda^I \leq \tilde{\lambda} = \lambda^*$.

Note that this proof requires Proposition 1 and Theorem 1 to ensure that λ is increased towards the global maximum rather than some suboptimal local maximum. Proposition 1 and Theorem 1 in turn require the uniform willingness-to-pay assumption (i.e., $\lambda = 1 - p$). However, the signs $\partial f(\lambda, \bar{c}(\lambda))/\partial \bar{c} \leq 0$ and $d\bar{c}(\lambda)/d\lambda < 0$ in (A.30) would be expected to hold in other sensical models of shared rides. The former means that profit is decreasing in cost. The latter means that the average cost per request is decreasing in demand, which is reasonable because requests are more likely to be matched as demand increases.

□

Proof of Theorem 3. The proof is a special case of the proof of Theorem 4 with $\beta = 0$. □

Proof of Proposition 2. We will use the following facts, which we discussed with Proposition 1. For $c \in (0, 1)$,

- $\lambda^{\#} < \lambda^*$ if and only if $\Pi'(\lambda^{\#}) > 0$; and
- $\lambda^{\#} > \lambda^*$ if and only if $\Pi'(\lambda^{\#}) < 0$.

For $c \in [1, 2)$ and $\theta \in (0, \tilde{\theta}(c))$,

- $\lambda^{\#} < \lambda^*$ if and only if $\Pi'(\lambda^{\#}) > 0$; and
- $\lambda^{\#} > \lambda^*$ if and only if $\Pi'(\lambda^{\#}) < 0$ *and* $\Pi''(\lambda^{\#}) < 0$.

Finally, for $c \in [1, 2)$ and $\theta \in [\tilde{\theta}(c), \infty)$, $\lambda^* = 0$.

Note that the case when $c \in [1, 2)$ and $\theta \in (0, \tilde{\theta}(c))$ requires looking at the sign of the second derivative. We have the following lemmas, which we prove later in the e-companion, and which allow us to focus only on the sign of the first derivative.

Lemma 4. $\Pi''(\lambda^*(0)) < 0$ $\forall c \in [1, 2)$ *and* $\theta \in (0, \tilde{\theta}(c))$.

Lemma 5. $\Pi''(\bar{\lambda}^*) < 0$ $\forall c \in [1, 2)$ *and* $\theta \in (0, \tilde{\theta}(c))$.

Relationship between $\lambda^*(1)$ and λ^*. From (2.7), we have that $\lambda^*(0) + \lambda^*(1) = 1 - c/2$. From (A.21a) and the first-order condition, it follows that $2\lambda^* \leq 1 - c/2$. Therefore,

$$\lambda^* \leq \frac{1}{2}\left(\lambda^*(0) + \lambda^*(1)\right) \leq \lambda^*(1),$$

where the last step comes from $\lambda^*(0) \leq \lambda^*(1)$.

Relationship between $\lambda^*(0)$ and λ^*. Using Theorem 3 and (A.21a), we have:

$$\Pi'(\lambda^*(0)) = c\theta \frac{(c-2)^4 - 2(c-2)^2(5c-6)\theta + 16(c-1)(2c-3)\theta^2 - 8(3c-2)\theta^3}{(2+2\theta-c)(4-4c+c^2+8\theta-6c\theta+4\theta^2)^2}. \quad \text{(A.31)}$$

For $c \in (0, 2/3]$, (A.31) is positive, so $\lambda^*(0) < \lambda^*$. If $c \in (2/3, 1)$, then there is a threshold of θ before which (A.31) is positive and $\lambda^*(0) < \lambda^*$, and after which (A.31) is negative and $\lambda^*(0) > \lambda^*$.

Now, consider $c \in [1, 2)$ and $\theta \in (0, \tilde{\theta}(c))$. Since the denominator of (A.31) is positive, we focus on the numerator, which we call $h(\theta)$, and observe that:

$$h(0) > 0, \quad \text{(A.32a)}$$

$$h(\tilde{\theta}(c)) < 0 \quad \text{if } c \in \left[1, \frac{10 + 8\sqrt{3}}{23}\right), \quad \text{(A.32b)}$$

$$h(\tilde{\theta}(c)) \geq 0 \quad \text{if } c \in \left[\frac{10+8\sqrt{3}}{23}, 2\right). \tag{A.32c}$$

(*Note:* In (A.32b) and (A.32c), the threshold $(10+8\sqrt{3})/23$ comes from the following:

$$\begin{aligned} h(\tilde{\theta}(c)) \geq 0 &\iff 4(329c^3 - 30c^2 + 236c - 24)\sqrt{2c(c-1)} \geq 1,861c^4 - 3,928c^3 + 2,600c^2 - 544c + 16 \\ &\iff 16(329c^3 - 30c^2 + 236c - 24)2(2c)(c-1) \geq (1,861c^4 - 3,928c^3 + 2,600c^2 - 544c + 16)^2 \\ &\iff (c-2)^4(17c^2 + 4c + 4)(23c^2 - 20c - 4) \geq 0 \\ &\iff c \geq \frac{10+8\sqrt{3}}{23}. \end{aligned}$$

where the second line is because both sides of the first line are positive when $c \in (1,2)$, and the last line applies $c \in (1,2)$ to discard the other root.)

Furthermore, from $h'(\theta)$, it is straightforward to verify that, for $\theta \in (0, \tilde{\theta}(c))$:

$$h(\theta) \text{ is increasing then decreasing} \qquad \text{if } c \in [1, 6/5), \tag{A.33a}$$

$$h(\theta) \text{ is decreasing} \qquad \text{if } c \in [6/5, 2). \tag{A.33b}$$

Combining (A.32a), (A.32b) and (A.33a), we have that for $c \in [1, (10+8\sqrt{3})/23)$, there exists some threshold $\hat{\theta}(c) \in (0, \tilde{\theta}(c))$ before which $h(\theta) > 0$ and $\lambda^*(0) < \lambda^*$, and after which $h(\theta) < 0$ and $\lambda^*(0) > \lambda^*$. Combining (A.32a), (A.32c) and (A.33), we have that for $c \in [(10+8\sqrt{3})/23, 2)$ and $\theta \in (0, \tilde{\theta}(c))$, then $h(\theta) > 0$ and $\lambda^*(0) < \lambda^*$. From Proposition 1 and Theorem 3, if $c \in [1,2)$ and $\theta \in [\tilde{\theta}(c), (1-c/2)^2/(c-1))$, then $\lambda^*(0) > 0 = \lambda^*$; and if $\theta \geq (1-c/2)^2/(c-1)$, then $\lambda^*(0) = \lambda^* = 0$.

Relationship between λ^* and $\bar{\lambda}^*$. From Theorem 3, $\bar{\lambda}^*$ can be written as:

$$\bar{\lambda}^* = \pi_0\lambda^*(0) + \pi_1\lambda^*(1) = \frac{\big((2-c)^2 + 4\theta(1-c)\big)\big((2-c)^2 + 2\theta(4-c+2\theta)\big)}{4(2-c+2\theta)^3}. \tag{A.34}$$

The logic for $\bar{\lambda}^*$ is similar to that for $\lambda^*(0)$; we seek to show that there exists a threshold of θ before which $\Pi'(\bar{\lambda}^*) \geq 0$ and after which $\Pi'(\bar{\lambda}^*) < 0$. Plugging (A.34) into (A.21b), we have:

$$\begin{aligned} \Pi'(\bar{\lambda}^*) = 2c^2\theta^2\Big((2-c)^8 &+ 16(2-c)^7\theta + 16(2-c)^4(26 - 28c + 7c^2)\theta^2 + 64(2-c)^3(22 - 27c + 7c^2)\theta^3 \\ &+ 64(160 - 398c + 348c^2 - 128c^3 + 17c^4)\theta^4 - 128(-32 + 85c - 60c^2 + 12c^3)\theta^5 \\ &+ 128(-4 - 13c + 8c^2)\theta^6 + 128(-8 + c)\theta^7 - 256\theta^8\Big) \\ /\Big((2-c+2\theta)^3&\big((2-c)^4 + 8(2-c)^3\theta + 8(12 - 13c + 3c^2)\theta^2 + (64 - 40c)\theta^3 + 16\theta^4\big)^2\Big). \end{aligned} \tag{A.35}$$

Since $2c^2\theta^2$ and the denominator of (A.35) are both positive, we focus on the remainder of the function, calling it $q(\theta)$. Observe from (A.35) that $q(0) = (2-c)^8 > 0$ and $\lim_{\theta\to\infty} q(\theta) = -\infty$. We show that $q(\theta)$ only has one root for $\theta \in (0, \infty)$ by analyzing the higher-order

derivatives of $q(\theta)$ up to $q^{(7)}(\theta)$. The derivatives for $c \in (0,1)$ are summarized in Table A.1, from which we conclude that $q(\theta)$ starts from a point $q(0) > 0$, increases, and then decreases toward $\lim_{\theta\to\infty} q(\theta) = -\infty$. Therefore, when $c \in (0,1)$, there exists a threshold of θ before which $q(\theta) > 0$ is positive and after which $q(\theta) < 0$.

Derivative	**Shape on** $(0,\infty)$	$q^{(i)}(0)$	$\lim_{\theta\to\infty} q^{(i)}(\theta)$
$q^{(7)}(\theta)$	$\searrow$	$-$	$-\infty$
$q^{(6)}(\theta)$	$\searrow$	$-$	$-\infty$
$q^{(5)}(\theta)$	$\searrow$	$\begin{cases} + & \text{if } c \in (0, 0.60), \\ - & \text{if } c \in (0.60, 1) \end{cases}$	$-\infty$
$q^{(4)}(\theta)$	$\begin{cases} \nearrow\searrow & \text{if } c \in (0, 0.60), \\ \searrow & \text{if } c \in (0.60, 1) \end{cases}$	$\begin{cases} + & \text{if } c \in (0, 0.95), \\ - & \text{if } c \in (0.95, 1) \end{cases}$	$-\infty$
$q^{(3)}(\theta)$	$\begin{cases} \nearrow\searrow & \text{if } c \in (0, 0.95), \\ \searrow & \text{if } c \in (0.95, 1) \end{cases}$	$+$	$-\infty$
$q^{(2)}(\theta)$	$\nearrow\searrow$	$+$	$-\infty$
$q^{(1)}(\theta)$	$\nearrow\searrow$	$+$	$-\infty$
$q(\theta)$	$\nearrow\searrow$	$+$	$-\infty$

Table A.1: Derivatives of $q(\theta)$ on $(0,\infty)$ for $c \in (0,1)$.

We omit the case when $c \in [1,2)$ for brevity: it follows a similar structure as the case of $c \in (0,1)$, except that it focuses on $\theta \in (0, \tilde{\theta}(c))$ and involves numerous cases. □

Proof of Lemma 4. For convenience, we prove $\Pi''(\lambda^*(0)) < 0$ for the equivalent conditions $c \in (1, 2+3\theta - 2\sqrt{2\theta(\theta+1)})$ and $\theta \in (0,1)$. For brevity, we define $\gamma(\theta) = 2+3\theta-2\sqrt{2\theta(\theta+1)}$.

Plugging (2.7) into (A.21b) and rearranging, we aim to show that:

$$\left(2\frac{(1-c/2)^2+\theta(1-c)}{2+2\theta-c}+\theta\right)^3 - c\theta^2 > 0, \tag{A.36}$$

for $c \in (1, \gamma(\theta))$ and $\theta \in (0,1)$.

We rewrite the left-hand side of (A.36) as:

$$\begin{aligned}\left(2\frac{(1-c/2)^2+\theta(1-c)}{2+2\theta-c}+\theta\right)^3 - c\theta^2 = \frac{1}{8(2-c+2\theta)^3}\Big[&c^6 - 6c^5(2+3\theta) + 4c^4(15+42\theta+32\theta^2) \\ &- 8c^3(20+78\theta+108\theta^2+51\theta^3) \\ &+ 48c^2(1+\theta)^2(5+14\theta+12\theta^2)\end{aligned}$$

$$- 32c(1+\theta)^3(6+15\theta+11\theta^2) + 64(1+\theta)^6\Big]. \tag{A.37}$$

Since the denominator is positive, we focus on the numerator, which we call $g(c)$. We can show that $g(c) > 0$ for $c \in [1, \gamma(\theta))$ and $\theta \in (0,1)$ by analyzing the higher-order derivatives of $g(c)$ up to $g^{(5)}(c)$:

- Clearly, $g^{(5)}(c) = 720(c-2-3\theta) < 0$.
- Therefore, $g^{(4)}(c)$ is monotonically decreasing. Then, since $g^{(4)}(\gamma(\theta)) > 0$, we have $g^{(4)}(c) > 0$ for $c \in [1, \gamma(\theta))$.
- Therefore, $g^{(3)}(c)$ is monotonically increasing. Then, since $g^{(3)}(\gamma(\theta)) < 0$, we have $g^{(3)}(c) < 0$ for $c \in [1, \gamma(\theta))$.

The pattern alternates until finally we have that $g(c)$ is monotonically decreasing on $c \in [1, \gamma(\theta))$ with $g(\gamma(\theta)) > 0$ for $\theta \in (0,1)$. Therefore, $g(c) > 0$ for $c \in [1, \gamma(\theta))$ and $\theta \in (0,1)$. □

Proof of Lemma 5. By (A.21b), $\Pi''(\lambda)$ is monotonically decreasing in λ. Then, using Lemma 4 and the fact that $\lambda^*(0) < \bar{\lambda}^* < \lambda^*(1)$, it follows that $\Pi''(\bar{\lambda}^*) < 0$ when $c \in [1,2)$ and $\theta \in (0, \tilde{\theta}(c))$. □

Proof of Proposition 3. Since static pricing charges the same price to all requests, its payment per converted request is simply $p^* = 1 - \lambda^*$. Similarly, the payment per converted request under iterative pricing is $p^I = 1 - \lambda^I$. For match-based pricing, using Theorem 3, the payment per converted request is:

$$p^M = \frac{(c-2)^2(c+2) + 2\theta(8+2c-c^2+4\theta+4c\theta)}{4\left((c-2)^2 + 2\theta(2\theta-c+4)\right)}. \tag{A.38}$$

We already showed in Theorem 2 that $p^* \le p^I$. It remains to show that $p^M \le p^*$, or,

$$\lambda^* \le 1 - \frac{(c-2)^2(c+2) + 2\theta(8+2c-c^2+4\theta+4c\theta)}{4\left((c-2)^2 + 2\theta(2\theta-c+4)\right)}. \tag{A.39}$$

To prove this, we upper bound λ^* as follows. Using the first-order condition of static pricing and the fact that $\lambda^* \le 1$, we iteratively derive a sequence of upper bounds for λ^*, which we call $\lambda^U_{(k)}$ for $k \in \mathbb{Z}_+$. In particular, from (A.21a), we have:

$$\lambda^* = \frac{1}{2}\left[-\frac{c\theta^2}{2(2\lambda^*+\theta)^2} + \left(1-\frac{c}{2}\right)\right] \le \frac{1}{2}\left[-\frac{c\theta^2}{2(2\lambda^U_{(k)}+\theta)^2} + \left(1-\frac{c}{2}\right)\right] = \lambda^U_{(k+1)}, \forall k \in \mathbb{Z}_+.$$

Starting with $\lambda^U_{(0)} = 1$, we then have:

$$\lambda^U_{(1)} = \frac{1}{2}\left[-\frac{c\theta^2}{2(2+\theta)^2} + \left(1 - \frac{c}{2}\right)\right], \quad \lambda^U_{(2)} = \frac{1}{2}\left[-\frac{c\theta^2}{2(2\lambda^U_{(1)}+\theta)^2} + \left(1 - \frac{c}{2}\right)\right].$$

Now, we aim to show that the right-hand side of (A.39) exceeds $\lambda^U_{(2)}$, which reduces to:

$$\frac{c^2\theta^3[-32 + 8(c-10)\theta + 8(c-10)\theta^2 + 3(c-12)\theta^3 - 6\theta^4]}{4\left((c-2)^2 + 2\theta(4-c+2\theta)\right)\left(2(c-2) + 2(c-4)\theta + (c-5)\theta^2 - \theta^3\right)^2} \leq 0, \tag{A.40}$$

which is true for $c \in (0,2)$ and $\theta > 0$ by inspection.

The cost per request under match-based pricing replaces λ in eq. (2.1) with $\lambda(1)$, and is decreasing in $\lambda(1)$. In Theorem 2 and Proposition 2, we showed that $\lambda^I \leq \lambda^* \leq \lambda^*(1)$. Therefore, $\bar{c}(\lambda^I) \geq \bar{c}(\lambda^*) \geq \bar{c}(\lambda^*(1))$. The cost per request is decreasing in the match rate, so $\rho(\lambda^I) \leq \rho(\lambda^*) \leq \rho(\lambda^*(1))$. □

Proof of Lemma 1. Under any pricing policy $\boldsymbol{\lambda}$, let $J^k(\mathbf{s})$ be the *minimum* expected cost to dispatch all currently waiting requests in state $\mathbf{s}$ as well as up to k new arrivals of rider requests. The expectation is taken over all future randomness including new rider arrivals and existing rider departures (solo dispatches). We have the following claim.

Claim 1. *Consider any state* $\mathbf{s}$ *with* $s_i \geq 3$ *(or* $s_i \geq 2$ *if rider type* i *satisfies eq.* (A.43) *for all* $j, m \in [N]$*) for some* $i \in [N]$*, and define another state* $\mathbf{s}'$ *with* $s'_i = s_i - 2$ *and* $s'_j = s_j$*,* $\forall j \neq i$*. The following holds for all* $k \in \mathbb{N}$*,*

$$c_{(i,i)} + J^k(\mathbf{s}') \leq J^k(\mathbf{s}). \tag{A.41}$$

Moreover, for any $j \neq i$ *such that* $s_j \geq 1$*, consider a state* $\mathbf{s}''$ *with* $s''_i = s_i - 1, s''_j = s_j - 1$ *and* $s''_m = s_m$*,* $\forall m \neq j, i$*. The following holds for all* $k \in \mathbb{N}$*,*

$$c_{(i,i)} + J^k(\mathbf{s}') \leq c_{(i,j)} + J^k(\mathbf{s}''). \tag{A.42}$$

We prove the claim by induction. Both equations (A.41) and (A.42) hold when $k = 0$ because

$$\begin{aligned}
c_{(i,i)} + J^0(\mathbf{s}') &= c_{(i,i)} + \sum_{m\in[N]} s'_m c_{(m)} \\
&\leq \sum_{m\in[N]} s_m c_{(m)} \\
&= J^0(\mathbf{s}), \\
c_{(i,i)} + J^0(\mathbf{s}') &= c_{(i,i)} + \sum_{m\in[N]} s'_m c_{(m)} \\
&= c_{(i,i)} + \sum_{m\in[N]} s_m c_{(m)} - 2c_{(i)}
\end{aligned}$$

$$\begin{aligned}
&\leq c_{(i,j)} + \sum_{m\in[N]} s_m c_{(m)} - c_{(i)} - c_{(j)} && (c_{(i,j)} - c_{(j)} \geq 0)\\
&= c_{(i,j)} + \sum_{m\in[N]} s''_m c_{(m)}\\
&= c_{(i,j)} + J^0(\mathbf{s}'').
\end{aligned}$$

Supposing that equations (A.41) and (A.42) hold at k, we now show that they also hold at $k+1$.

We first show (A.41). We consider a stochastic system starting from $\mathbf{s}$ (coined S) and another starting from $\mathbf{s}'$ (coined S') and couple their randomness. If the first event is a solo dispatch for some type i rider in S but not in S', then $J^{k+1}(\mathbf{s}) = c_{(i)} + J^{k+1}(\mathbf{s} - \mathbf{e}_i) \geq c_{(i,i)} + J^{k+1}(\mathbf{s} - 2\cdot\mathbf{e}_i) = c_{(i,i)} + J^{k+1}(\mathbf{s}')$. If the first event is a solo dispatch for some rider in both S and S', then we can simply redefine states $\mathbf{s}$ and $\mathbf{s}'$ after this solo dispatch and perform the analysis thereafter. On the other hand, if the first event is a new arrival of rider type m, we consider the following dispatching outcomes.

- m is kept waiting first in both S and S'. In such a case,

$$\begin{aligned}
c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m)\\
&\leq J^k(\mathbf{s} + \mathbf{e}_m) && \text{(by induction hypothesis (A.41) at } k)\\
&= J^{k+1}(\mathbf{s}).
\end{aligned}$$

- m is matched to a rider in S but kept waiting first in S'. First, consider that the rider to whom m is matched in S also exists in S' and is of type n. In such a case,

$$\begin{aligned}
c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m)\\
&\leq c_{(i,i)} + c_{(m,n)} + J^k(\mathbf{s}' - \mathbf{e}_n)\\
&&& \text{(by optimality of the matching policy)}\\
&\leq c_{(m,n)} + J^k(\mathbf{s} - \mathbf{e}_n) && \text{(by induction hypothesis (A.41) at } k)\\
&= J^{k+1}(\mathbf{s}).
\end{aligned}$$

  Second, consider that the rider to whom m is matched in S does *not* exist in S'; since $\mathbf{s}' = \mathbf{s} - 2\mathbf{e}_i$, the rider has to be of type i. In such a case,

$$\begin{aligned}
c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m)\\
&\leq c_{(m,i)} + J^k(\mathbf{s} - \mathbf{e}_i) && \text{(by induction hypothesis (A.42) at } k)\\
&= J^{k+1}(\mathbf{s}).
\end{aligned}$$

- m is kept waiting first in S but matched to a rider in S'. Since $\mathbf{s}' = \mathbf{s} - 2\mathbf{e}_i$, the rider

to whom m is matched in S' must exist in S (call its type n). Then,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + c_{(m,n)} + J^k(\mathbf{s}' - \mathbf{e}_n) \\ &\leq c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m) \qquad \text{(by optimality of the matching policy)} \\ &\leq J^k(\mathbf{s} + \mathbf{e}_m) \qquad \text{(by induction hypothesis (A.41) at } k) \\ &= J^{k+1}(\mathbf{s}). \end{aligned}$$

- m is matched to a rider in both S and S'. If the rider to whom m is matched is the same in S and S', we can simply redefine the states $\mathbf{s}$ and $\mathbf{s}'$ after this match. Let us consider the case where m is matched to a rider of type n in S, and is matched to a rider of type $n' \neq n$ in S'. If n is in S' as well,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + c_{(m,n')} + J^k(\mathbf{s}' - \mathbf{e}_{n'}) \\ &\leq c_{(i,i)} + J^k(\mathbf{s}' - \mathbf{e}_n) + c_{(m,n)} \\ &\qquad \text{(by optimality of the matching policy)} \\ &\leq J^k(\mathbf{s} - \mathbf{e}_n) + c_{(m,n)} \qquad \text{(by induction hypothesis (A.41) at } k) \\ &= J^{k+1}(\mathbf{s}). \end{aligned}$$

  If n is not in S', then n must be i, since $\mathbf{s}' = \mathbf{s} - 2\mathbf{e}_i$. In such a case,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + c_{(m,n')} + J^k(\mathbf{s}' - \mathbf{e}_{n'}) \\ &\leq c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m) \qquad \text{(by optimality of the matching policy)} \\ &\leq c_{(m,i)} + J^k(\mathbf{s} - \mathbf{e}_i) \qquad \text{(by induction hypothesis (A.42) at } k) \\ &= c_{(m,n)} + J^k(\mathbf{s} - \mathbf{e}_n) \\ &= J^{k+1}(\mathbf{s}). \end{aligned}$$

This completes the proof by induction for (A.41). We now proceed to prove (A.42) by induction as well. The case of $k = 0$ is proven above; we thus prove the case of $k + 1$ assuming it holds at k. Similarly, we consider a stochastic system starting at $\mathbf{s}'$ (coined S') and another starting at $\mathbf{s}''$ (coined S'') and couple their randomness. If the first event is a solo dispatch for some rider in both S' and S'', we can simply redefine states $\mathbf{s}'$ and $\mathbf{s}''$ after this solo dispatch and perform the analysis thereafter. If the first event is two solo dispatches, one for each distinct rider in S' and S'' (we couple them together as they share the same rate for their sojourn times), then after this event, the two states become the same, and (A.42) trivially holds as $c_{(i,i)} \leq c_{(i,j)}$. Now suppose that the first event is a new arrival of rider type m and consider the following dispatching outcomes.

- m is kept waiting first in both S' and S''. In such a case,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^k(\mathbf{s}' + \mathbf{e}_m) \\ &= c_{(i,j)} + J^k(\mathbf{s}'' + \mathbf{e}_m) \qquad \text{(by induction hypothesis (A.42) at } k) \end{aligned}$$

$$= c_{(i,j)} + J^{k+1}(\mathbf{s}'').$$

- m is matched to a rider in S' but kept waiting first in S''. Suppose that the rider to whom m is matched in S' is of type n. Then,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + c_{(m,n)} + J^{k}(\mathbf{s}' - \mathbf{e}_n) \\ &\leq c_{(i,i)} + J^{k}(\mathbf{s}' + \mathbf{e}_m) \qquad \text{(by optimality of the matching policy)} \\ &\leq c_{(i,j)} + J^{k}(\mathbf{s}'' + \mathbf{e}_m) \qquad \text{(by induction hypothesis (A.42) at } k) \\ &= c_{(i,j)} + J^{k+1}(\mathbf{s}''). \end{aligned}$$

- m is matched to a rider in S'' but kept waiting first in S'. First, consider that the rider to whom m is matched in S'' also exists in S' and is of type n. In such a case,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^{k}(\mathbf{s}' + \mathbf{e}_m) \\ &\leq c_{(i,i)} + c_{(m,n)} + J^{k}(\mathbf{s}' - \mathbf{e}_n) \\ &\qquad \text{(by optimality of the matching policy)} \\ &\leq c_{(i,j)} + c_{(m,n)} + J^{k}(\mathbf{s}'' - \mathbf{e}_n) \\ &\qquad \text{(by induction hypothesis (A.42) at } k) \\ &= c_{(i,j)} + J^{k+1}(\mathbf{s}''). \end{aligned}$$

  Second, consider that the rider to whom m is matched in S'' does *not* exist in S' and therefore has to be of type i. In the case when $s_i \geq 3$, this is a contradiction as $\mathbf{s}' = \mathbf{s} - 2 \cdot \mathbf{e}_i$ implies that S' contains a type i rider. In the case when $s_i \geq 2$ and rider type i satisfies (A.43) for all $j, m \in [N]$, we know S' must contain a type j rider. Then,

$$\begin{aligned} c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + J^{k}(\mathbf{s}' + \mathbf{e}_m) \\ &\leq c_{(i,i)} + c_{(m,j)} + J^{k}(\mathbf{s}' - \mathbf{e}_j) \\ &\qquad \text{(by optimality of the matching policy)} \\ &\leq c_{(i,j)} + c_{(m,i)} + J^{k}(\mathbf{s}'' - \mathbf{e}_i) \qquad \text{(by eq. (A.43), } \mathbf{s}'' - \mathbf{e}_i = \mathbf{s}' - \mathbf{e}_j) \\ &= c_{(i,j)} + J^{k+1}(\mathbf{s}''). \end{aligned}$$

- m is matched to a rider in both S' and S''. If the rider to whom m is matched is the same in S and S', we can simply redefine the states $\mathbf{s}$ and $\mathbf{s}'$ after this match. Let us consider the case where m is matched to a rider of type n in S', and is matched to a rider of type $n' \neq n$ in S''. If n' is in S' as well,

$$c_{(i,i)} + J^{k+1}(\mathbf{s}') = c_{(i,i)} + c_{(m,n)} + J^{k}(\mathbf{s}' - \mathbf{e}_n)$$

$$\begin{aligned}
&\leq c_{(i,i)} + c_{(m,n')} + J^k(\mathbf{s}' - \mathbf{e}_{n'}) \\
&\qquad\qquad \text{(by optimality of the matching policy)} \\
&\leq c_{(i,j)} + c_{(m,n')} + J^k(\mathbf{s}'' - \mathbf{e}_{n'}) \\
&\qquad\qquad \text{(by induction hypothesis (A.42) at } k) \\
&= c_{(i,j)} + J^{k+1}(\mathbf{s}'').
\end{aligned}$$

If n' is not in S', then n' has to be i. In the case when $s_i \geq 3$, this is a contradiction as $\mathbf{s}' = \mathbf{s} - 2 \cdot \mathbf{e}_i$ implies that S' contains a type i rider. In the case when $s_i \geq 2$ and rider type i satisfies (A.43) for all $j, m \in [N]$, we know S' must contain a type j rider. Then,

$$\begin{aligned}
c_{(i,i)} + J^{k+1}(\mathbf{s}') &= c_{(i,i)} + c_{(m,n)} + J^k(\mathbf{s}' - \mathbf{e}_n) \\
&\leq c_{(i,i)} + c_{(m,j)} + J^k(\mathbf{s}' - \mathbf{e}_j) \\
&\qquad\qquad \text{(by optimality of the matching policy)} \\
&\leq c_{(i,j)} + c_{(m,i)} + J^k(\mathbf{s}'' - \mathbf{e}_i) \qquad \text{(by eq. (A.43), } \mathbf{s}'' - \mathbf{e}_i = \mathbf{s}' - \mathbf{e}_j) \\
&= c_{(i,j)} + J^{k+1}(\mathbf{s}'').
\end{aligned}$$

This completes the induction proof for (A.42).

Note that (A.41) immediately leads to the statement of Lemma 1 as it is weakly dominated to keep more than two riders of the same type in the system. □

Claim 2. *For any three requests i, j, and m in one dimension,*

$$c_{(i,i)} + c_{(m,j)} \leq c_{(m,i)} + c_{(i,j)}. \tag{A.43}$$

Proof of Claim 2. First, suppose $c_{(m,i)} = c_{(m)} + c_{(i)}$, i.e., matching m and i has no cost saving. Then,

$$\begin{aligned}
c_{(m,i)} + c_{(i,j)} &= c_{(m)} + c_{(i)} + c_{(i,j)} \\
&\geq c_{(m)} + c_{(i)} + c_{(j)} && (c_{(i,j)} \geq c_{(j)}) \\
&\geq c_{(i,i)} + c_{(m,j)}. && (c_{(m,j)} \geq c_{(m)} + c_{(j)},\ c_{(i,i)} = c_{(i)})
\end{aligned}$$

The case when $c_{(i,j)} = c_{(i)} + c_{(j)}$ is identical to the case above, switching the ms with the js. Therefore, we can restrict our attention to $c_{(m,i)} < c_{(m)} + c_{(i)}$ and $c_{(i,j)} < c_{(i)} + c_{(j)}$ — namely, m overlaps with i, and i overlaps with j.

Now, suppose $c_{(m,j)} = c_{(m)} + c_{(j)}$. We want to show that:

$$c_{(i)} + c_{(m)} + c_{(j)} \leq c_{(m,i)} + c_{(i,j)}.$$

In this case, m overlaps with i and i overlaps with j, but m and j do not overlap with each other. Since m and j do not overlap with each other, it must be that they overlap with different parts of i, which also implies that neither of them can fully contain i.

Without loss of generality, we can assume that i fully contains m and j. This is because if m gets picked up before i (hence m is dropped off before i), this merely adds the distance from the origin of m to the origin of i to both sides of the inequality. Recalling that j cannot overlap with m, the reasoning is similar if j gets dropped off after i.

Since i fully contains m and j, and m and j do not overlap, then $c_{(m)} + c_{(j)} \leq c_{(i)}$. Then:

$$c_{(m,i)} + c_{(i,j)} = c_{(i)} + c_{(i)} \geq c_{(i)} + c_{(m)} + c_{(j)}.$$

To sum up the previous three cases, we can restrict our attention to $c_{(m,i)} < c_{(m)} + c_{(i)}$ and $c_{(i,j)} < c_{(i)} + c_{(j)}$ *and* $c_{(m,j)} < c_{(m)} + c_{(j)}$. Namely, all three pairs of riders can share. In one dimension, this means that *all* three riders are traveling in the *same* direction on a line, and that there must be a portion of the line where all three riders overlap. We fix a transfer point P in this portion of shared overlap and divide any trip into two *legs*: in the first leg, riders are picked up from their origins and then dropped off at P, and in the second leg, riders are re-picked up at P and dropped off at their destinations. Assuming that the transfer at P is costless, this division into legs does not change the analysis, since the total cost of the trips remains the same. But adding this transfer point P conveniently allows us to analyze the costs assuming that all riders have the same destination in the first leg (the case where all riders have the same origin in the second leg is the same by symmetry).

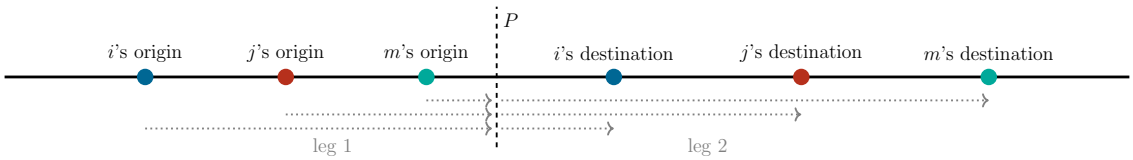


Figure A.5: Illustration of the proof of Claim 2.

Let ℓ_i, ℓ_j, and ℓ_m be the distances of the origins of i, j, and m to the transfer point P. Then, (A.43) becomes $\ell_i + \max\{\ell_m, \ell_j\} \leq \max\{\ell_m, \ell_i\} + \max\{\ell_i, \ell_j\}$. This is equivalent to $\max\{\ell_i + \ell_m, \ell_i + \ell_j\} \leq \max\{\ell_m + \ell_i, \ell_i + \ell_i, \ell_m + \ell_j, \ell_i + \ell_j\}$, which is always true. □

Proof of Theorem 4. Since in an optimal solution, the platform either offers shared rides only or offers solo rides only, we can separately optimize the two cases.

Solo Rides Only. Maximizing the solo rides profit $\lambda_X(1 - \lambda_X - c)$, we obtain the profit-maximizing solo conversion rate $\lambda_X^* = (1 - c)/2$.

Shared Rides Only. Using (A.2), (A.3), and (A.4), the profit maximization problem is given by:

$$\max_{\lambda(0),\lambda(1)} \quad \Pi(\lambda(0), \lambda(1)) = \frac{\lambda(0)}{\lambda(0) + \lambda(1) + \theta} \Bigg(\lambda(1) \Big(1 - \beta \frac{\lambda(1)}{\lambda(1) + \theta} - \lambda(0) + 1 - \beta - \lambda(1) - c \Big) + \theta \Big(1 - \beta \frac{\lambda(1)}{\lambda(1) + \theta} - \lambda(0) - c \Big) \Bigg)$$

$$= \frac{\lambda(0)}{\lambda(0)+\lambda(1)+\theta}\Big(-\lambda(1)^2 + (2-2\beta-c-\lambda(0))\lambda(1) + (1-c-\lambda(0))\theta\Big).$$

The first-order derivatives are:

$$\frac{\partial \Pi(\lambda(0),\lambda(1))}{\partial \lambda(0)} = -\frac{(\lambda(1)+\theta)\Big(\lambda(0)^2 + 2(\lambda(1)+\theta)\lambda(0) + \lambda(1)^2 + (c+2\beta-2)\lambda(1) + (c-1)\theta\Big)}{(\lambda(0)+\lambda(1)+\theta)^2}, \tag{A.44}$$

$$\frac{\partial \Pi(\lambda(0),\lambda(1))}{\partial \lambda(1)} = -\frac{\lambda(0)\Big(\lambda(0)^2 + (2\beta+c+2\lambda(1)-2)\lambda(0) + \lambda(1)^2 + (2\lambda(1)+2\beta-1)\theta\Big)}{(\lambda(0)+\lambda(1)+\theta)^2}. \tag{A.45}$$

Setting (A.44) equal to zero, there are two cases. First, if $\lambda(0) = 0$, then $\lambda(1)$ can take any value in $[0,1]$ with no change in profit (which is always zero). Otherwise, if $\lambda(0) > 0$, we get $\lambda(0)$ as a function of $\lambda(1)$:

$$\lambda(0) = -\lambda(1) - \theta + \Delta, \tag{A.46}$$

where $\Delta := \sqrt{(2-2\beta-c+2\theta)\lambda(1) + \theta(1-c+\theta)}$. Note that the other root $-\lambda(1)-\theta-\Delta$ is negative and can be eliminated.

Next, we plug (A.46) into (A.45) and set (A.45) equal to zero, which gives:

$$2\Delta^2 - (2-2\beta-c+2\theta)\Delta = 0. \tag{A.47}$$

Solving equation (A.47), we get $\Delta = 0$ or $\Delta = 1-\beta-c/2+\theta$. The case $\Delta = 0$ is discarded because it implies $\lambda(0) = -\lambda(1) - \theta < 0$. The case $\Delta = 1-\beta-c/2+\theta$ simplifies to:

$$\lambda^*(1) = \frac{(1-\beta-c/2)^2 + (1-2\beta)\theta}{2-2\beta-c+2\theta}. \tag{A.48}$$

Plugging (A.48) into (A.46), it follows that:

$$\lambda^*(0) = \frac{(1-\beta-c/2)^2 + (1-c)\theta}{2-2\beta-c+2\theta}.$$

This is the only possible critical point inside the interior of the feasible region $(0,1)^2$. Suppose that the critical point $(\lambda^*(0),\lambda^*(1)) \in (0,1)^2$ for some parameters β, θ, c, we first verify that it is a local maximum by applying the second derivative test.

$$\left.\frac{\partial^2 \Pi(\lambda(0),\lambda(1))}{\partial \lambda(0)^2}\right|_{(\lambda^*(0),\lambda^*(1))} = -\frac{2(\lambda^*(1)+\theta)(\lambda^*(0)+\lambda^*(1)+\theta)^2}{(\lambda^*(0)+\lambda^*(1)+\theta)^3} < 0, \tag{A.49a}$$

$$\left.\frac{\partial^2 \Pi(\lambda(0),\lambda(1))}{\partial \lambda(1)^2}\right|_{(\lambda^*(0),\lambda^*(1))} = -\frac{2\lambda^*(0)\Big(\lambda^*(0)(2-2\beta-c+2\theta) + \theta(1-2\beta+\theta)\Big)}{(\lambda^*(0)+\lambda^*(1)+\theta)^3}$$

$$= -\frac{2\lambda^*(0)}{\lambda^*(0)+\lambda^*(1)+\theta} < 0, \tag{A.49b}$$

$$\left.\frac{\partial^2\Pi(\lambda(0),\lambda(1))}{\partial\lambda(0)\partial\lambda(1)}\right|_{(\lambda^*(0),\lambda^*(1))} = -\Big(\lambda^*(0)^3 + 3\lambda^*(0)^2(\lambda^*(1)+\theta) + \lambda^*(0)(3\lambda^*(1)^2 + (2\beta+2c-3)\theta + 2\lambda^*(1)(2\beta+c+\theta-2)) + (\lambda^*(1)+\theta)(\lambda^*(1)^2 + (2\beta-1)\theta + 2\lambda^*(1)\theta)\Big)/(\lambda^*(0)+\lambda^*(1)+\theta)^3$$

$$= 0. \tag{A.49c}$$

In addition, we show that $\Pi(\lambda^*(0),\lambda^*(1))$ achieves *global maximum* when $(\lambda^*(0),\lambda^*(1)) \in (0,1)^2$. We do this by showing $\Pi(\lambda^*(0),\lambda^*(1))$ exceeds the profit at all of the boundary points $(0,\cdot)$, $(1,\cdot)$, $(\cdot,0)$, and $(\cdot,1)$.

- First, $(\lambda^*(0),\lambda^*(1))$ dominates $(0,\cdot)$, since

$$\Pi(\lambda^*(0),\lambda^*(1)) = \left(\frac{(1-\beta-c/2)^2+(1-c)\theta}{2-2\beta-c+2\theta}\right)^2 = \lambda^*(0)^2 \geq 0 = \Pi(0,\lambda),\ \forall\lambda\in[0,1].$$

- Next, we consider point $(\lambda,1)$ for some $\lambda\in[0,1]$, where the profit is

$$\Pi(\lambda,1) = \frac{\lambda((1-c-\lambda)(\theta+1)-2\beta)}{1+\lambda+\theta}.$$

  There are two cases. First, if $c\in[1,2)$, then,

$$\Pi(\lambda,1) < 0 \leq \Pi(\lambda^*(0),\lambda^*(1)).$$

  Second, if $c\in(0,1)$, look at its first-order and second-order derivatives:

$$\frac{d}{d\lambda}\Pi(\lambda,1) = -\frac{(\theta+1)(\lambda^2+2(\theta+1)\lambda-(\theta+1)(1-c)+2\beta)}{(1+\lambda+\theta)^2},$$
$$\frac{d^2}{d\lambda^2}\Pi(\lambda,1) = -\frac{2(\theta+1)\Omega}{(1+\lambda+\theta)^3}.$$

  where $\Omega := (2-c+\theta)(\theta+1)-2\beta$. Now we discuss two sub-cases:

  (i) When $\Omega\leq 0$, it follows that $\Pi(\lambda,1)$ is convex over $\lambda\in[0,1]$ and maximized at $\lambda=0$ or 1, so $\Pi(\lambda,1)\leq\max\{\Pi(1,1),\Pi(0,1)\} = \max\{-(2\beta+c+c\theta)/(2+\theta),0\} = 0\leq\Pi(\lambda^*(0),\lambda^*(1))$, $\forall\lambda\in[0,1]$.

  (ii) When $\Omega>0$, it follows that $\Pi(\lambda,1)$ is concave over $\lambda\in[0,1]$. With the first-order condition that lets $d\Pi(\lambda,1)/d\lambda = 0$, we get one potentially valid root $\tilde{\lambda} = \sqrt{\Omega}-(\theta+1)$ (as the other root is negative and thus eliminated). We can show that $\tilde{\lambda}\leq\lambda^*(0)$ for all β, c, θ since the inequality can be equivalently written as $(((2\beta+c)^2-4(1+\theta+2\beta\theta))/(4(2-2\beta-c+2\theta)))^2\geq 0$ which holds for all

β, c, θ. However, $\tilde{\lambda}$ may not always be positive, so we discuss the following two cases:

(a) If $\theta > [(2\beta + c - 1)/(1 - c)]^+$, combined with $c \in (0, 1)$, $\beta > 0$, then the stationary point $\tilde{\lambda} > 0$. Therefore for all $\lambda \in [0, 1]$,

$$\begin{aligned}\Pi(\lambda, 1) &\le \tilde{\lambda}^2 && \text{(since } \Pi(\lambda, 1) \text{ is maximized at } \lambda = \tilde{\lambda}) \\ &\le \lambda^*(0)^2 && \text{(since } 0 < \tilde{\lambda} \le \lambda^*(0)) \\ &= \Pi(\lambda^*(0), \lambda^*(1)).\end{aligned}$$

(b) For any other β, c, θ, when the stationary point $\tilde{\lambda} \le 0$, $\Pi(\lambda, 1)$ is maximized at $\lambda = 0$. This yields $\Pi(\lambda, 1) \le 0 \le \Pi(\lambda^*(0), \lambda^*(1))$, $\forall \lambda \in [0, 1]$.

- Next, we consider point $(1, \lambda)$ for some $\lambda \in [0, 1]$, where the profit is

$$\Pi(1, \lambda) = \frac{\lambda(1 - c - \lambda - 2\beta) - c\theta}{1 + \lambda + \theta}.$$

There are two cases. First, if $1 - c - \lambda - 2\beta \le 0$, then $\Pi(1, \lambda) \le 0 \le \Pi(\lambda^*(0), \lambda^*(1))$. Second, if $1 - c - \lambda - 2\beta > 0$,

$$\begin{aligned}\Pi(1, \lambda) &= \frac{\lambda(1 - c - \lambda - 2\beta) - c\theta}{1 + \lambda + \theta} \\ &< \frac{\lambda(1 - c - \lambda - 2\beta) + \lambda\theta(1 - c - \lambda)}{1 + \lambda + \theta} && \text{(since } 1 - c - \lambda > 2\beta \ge 0) \\ &= \frac{\lambda((1 - c - \lambda)(\theta + 1) - 2\beta)}{1 + \lambda + \theta} \\ &= \Pi(\lambda, 1) \\ &\le \Pi(\lambda^*(0), \lambda^*(1)).\end{aligned}$$

- Finally, consider point $(\lambda, 0)$ for some $\lambda \in [0, 1]$, where the profit is

$$\Pi(\lambda, 0) = \frac{\lambda(1 - c - \lambda)\theta}{\lambda + \theta}.$$

As before, there are two cases. First, if $1 - c - \lambda \le 0$, then $\Pi(\lambda, 0) \le 0 \le \Pi(\lambda^*(0), \lambda^*(1))$. Second, if $1 - c - \lambda > 0$ (which implies $c < 1 - \lambda \le 1$),

$$\begin{aligned}\frac{d}{d\lambda}\Pi(\lambda, 0) &= -\frac{\theta(\lambda^2 + 2\theta\lambda - (1 - c)\theta)}{(\lambda + \theta)^2}, \\ \frac{d^2}{d\lambda^2}\Pi(\lambda, 0) &= -\frac{2\theta^2(1 - c + \theta)}{(\lambda + \theta)^3} < 0,\end{aligned}$$

from which it follows that $\Pi(\lambda, 0)$ is maximized at $\tilde{\lambda} = \sqrt{\theta(1 - c + \theta)} - \theta$ (as the other root is negative and thus eliminated). Meanwhile, we can prove $\tilde{\lambda} \in (0, \lambda^*(0))$

for any $\beta \geq 0$, $c \in (0,1)$, $\theta > 0$: first, $\lambda^* = \sqrt{\theta(1-c+\theta)} - \theta > \sqrt{\theta^2} - \theta = 0$ since $c \in (0,1), \theta > 0$; second, the inequality $\tilde{\lambda} \leq \lambda^*(0)$ can be equivalently written as $(((2-2\beta-c)^2 + 4\theta(1-2\beta))/(4(2-2\beta-c+2\theta)))^2 \geq 0$ which holds for all β, c, θ. Therefore for all $\lambda \in [0,1]$,

$$\begin{aligned}
\Pi(\lambda, 0) &\leq \tilde{\lambda}^2 && \text{(since } \Pi(\lambda,0) \text{ is maximized at } \lambda = \tilde{\lambda}) \\
&\leq \lambda^*(0)^2 && \text{(since } 0 < \tilde{\lambda} \leq \lambda^*(0)) \\
&= \Pi(\lambda^*(0), \lambda^*(1)).
\end{aligned}$$

Therefore, $(\lambda^*(0), \lambda^*(1))$ is the global maximizer of $\Pi(\lambda(0), \lambda(1))$ given that $(\lambda^*(0), \lambda^*(1)) \in (0,1)^2$. When $(\lambda^*(0), \lambda^*(1))$ is not in the interior, from the discussions above we know that the global maximum either has zero revenue or lies on the boundaries of $(\cdot, 1)$ or $(\cdot, 0)$. In the latter case, however, $\Pi(\lambda, 0), \Pi(\lambda, 1) \leq \lambda(1-\lambda-c)$, $\forall \lambda \in [0,1]$. This suggests that when $(\lambda^*(0), \lambda^*(1))$ is not in the interior, offering shared rides only is weakly dominated by offering solo rides only. We can then conclude the analysis by comparing these two scenarios — offering shared rides with $(\lambda^*(0), \lambda^*(1))$ versus offering solo rides with $\lambda_X^* = (1-c)/2$.

- If $c \in (0,1)$, then both shared and solo rides can be operated profitably. To operate shared rides instead of solo rides, they must be the more profitable option, i.e.,

$$\left(\frac{(1-c/2-\beta)^2 + \theta(1-c)}{2-c-2\beta+2\theta}\right)^2 \geq \left(\frac{1-c}{2}\right)^2. \tag{A.50}$$

  Operating shared rides implies that (i) $\lambda^*(0) + \lambda^*(1) = 1 - c/2 - \beta > 0$ and (ii) $\lambda^*(0) > 0$. Then, $2-c-2\beta+2\theta = 2(\lambda^*(0) + \lambda^*(1) + \theta) > 0$, so (A.50) is equivalent to

$$\frac{(1-c/2-\beta+\theta(1-c)}{2-c-2\beta+2\theta} \geq \frac{1-c}{2},$$

  or $(c-2\beta)(1-c/2-\beta) \geq 0$. Since $1-c/2-\beta > 0$, we must have $\beta \leq c/2$. On the other hand, if $\beta > c/2$, then shared rides are less profitable, so the platform operates solo rides.
  Interestingly, the $\beta = c/2$ threshold also corresponds to the threshold at which the total welfare of a shared ride $(v-\beta-c/2)$ equals the total welfare of a solo ride $(v-c)$, where the welfare is summed over the rider and the platform.

- If $c \in [1,2)$, then solo rides are not profitable. In order to operate shared rides, we must again have (i) $\lambda^*(0) + \lambda^*(1) = 1 - c/2 - \beta > 0$ and (ii) $\lambda^*(0) > 0$. From the first condition, it follows that $\beta < 1 - c/2$. The first condition also implies that $2-c-2\beta+2\theta = 2(\lambda^*(0) + \lambda^*(1) + \theta) > 0$. Then, the second condition becomes

$$\left(1 - \frac{c}{2} - \beta\right)^2 + \theta(1-c) > 0,$$

  or $\theta < (1-c/2-\beta)^2/(c-1)$.

This completes the proof.

□

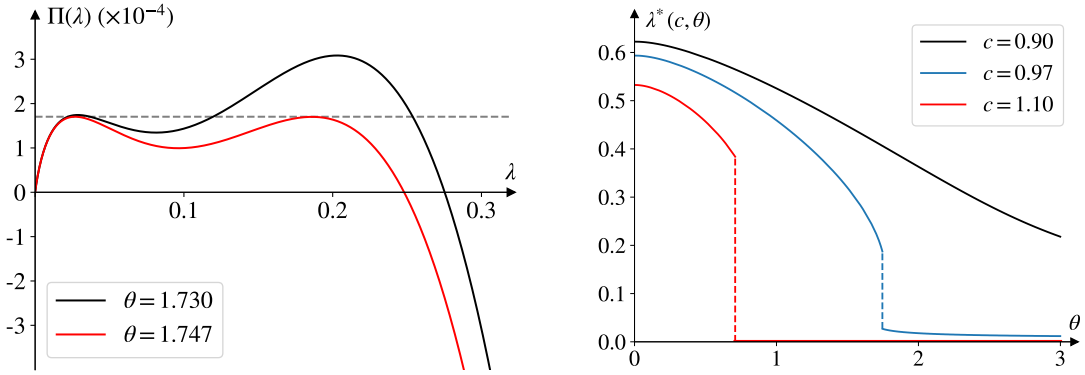
A.4 Robustness Tests

A.4.1 Willingness-To-Pay Distribution

In this section, we test whether the analytical results derived in Section 2.3.1 under a uniform willingness-to-pay distribution hold under other distributions as well. Specifically, we focus on a willingness-to-pay distribution $\mathsf{Beta}(a, b)$ with parameters a and b and a support of $[0, 1]$, which reduces to a uniform distribution when $a = b = 1$. We consider left-skewed distributions with mean $3/4$ ($a = 3$, $b = 1$ and $a = 6$, $b = 2$), right-skewed distributions with mean $1/4$ ($a = 1/3$, $b = 1$ and $a = 2/3$, $b = 2$), and a symmetric unimodal distribution with mean $1/2$ ($a = 2$, $b = 2$). We call riders with relatively high willingness-to-pay *high-value* riders, and riders with relatively low willingness-to-pay *low-value* riders.

Discontinuity of Static Prices. We first look at the expected profit $\Pi(\lambda)$ as a function of the static conversion rate λ. In most cases, the expected profit functions under different Beta distributions display characteristics similar to those described in Section 2.3.1 and illustrated in Figure 2.2 — namely, $\Pi(\lambda)$ has either only one global maximum, or it has two local maxima. It is interesting to note a case with $c = 0.97$, $a = 6$, $b = 2$, shown in Figure A.6a. As θ increases from zero, there exists a region where $\Pi(\lambda)$ contains two local maxima with positive profits. These two local maxima correspond to two operating regimes: the one with a lower conversion rate (higher price) operates at a higher dispatching cost from a lower match rate, and the other with a higher conversion rate (lower price) operates at a lower dispatching cost from a higher match rate. As θ keeps increasing, the profit of the local maximum with the higher conversion rate gradually decreases, and the optimal conversion rate drops to the other local maximum *abruptly*. Discontinuity still exists in the static conversion (and price) in this case. What is different is that the conversion can drop to a strictly positive value, as shown in Figure A.6b (blue curve). Intuitively, as the conversion rate increases from zero, the platform initially receives a positive profit. However, when the conversion rate continues to increase (price continues to decrease), the profit starts to drop due to the lack of high-value riders (as compared to the uniform willingness-to-pay distribution) and the high dispatching cost. It is not until the conversion rate increases to a sufficiently high level (namely, the price decreases enough) to attract low-value riders and thereby lower the dispatching cost that the service becomes profitable again.

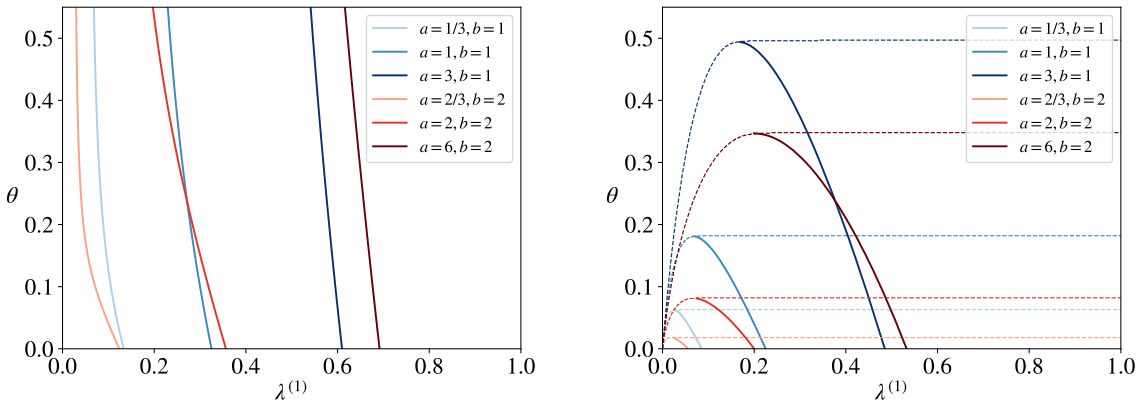
Iterative Pricing Overprices. For general willingness-to-pay distributions, it is difficult to provide closed-form expressions for the conversion rate to which iterative pricing converges (the *converged conversion rate*); we thus numerically verify the convergence of the procedure. Figure A.7 illustrates the results, which can be read similarly to Figure 2.3 except that we do not plot the ratio of the converged conversion rate to the optimal conversion rate λ^*. We do verify that for all instances tested, the iterative pricing procedure overprices. As Figure A.7 illustrates, the convergence pattern is very similar to that in the case of uniform willingness-to-pay, which depends on the values of c, θ, and the initial conversion rate $\lambda^{(1)}$ selected.



(a) Static profit function $\Pi(\lambda)$ when $c = 0.97$

(b) Optimal request conversion rate $\lambda^*(c, \theta)$

Figure A.6: Static prices under the willingness-to-pay distribution $\mathsf{Beta}(6, 2)$.

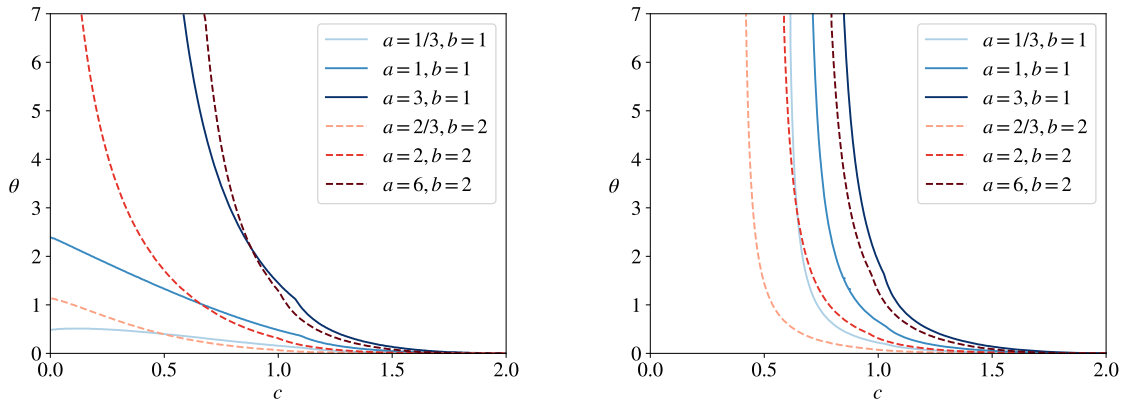


(a) $c = 0.7$

(b) $c = 1.1$

Figure A.7: Convergence of iterative pricing under different $\mathsf{Beta}(a, b)$ willingness-to-pay distributions. The solid curves are the converged conversion rates. If $c \leq 1$ (left figure), iterative pricing converges monotonically (the direction depends on which side it starts from) to the solid curve. If $c > 1$ (right figure), it converges monotonically to the solid curve if θ and the initial conversion rate $\lambda^{(1)}$ is within the area outlined by the corresponding dashed curve; otherwise, it converges to a conversion rate of zero.

Comparison between Match-based and Static Prices. For match-based pricing under general willingness-to-pay distributions, the observation that $\lambda^*(0) + \lambda^*(1) = 1 - c/2$ in Theorem 3 does not hold anymore, so it is difficult to have closed forms of the optimal match-based prices. The continuity of $(\lambda^*(0), \lambda^*(1))$ in (c, θ) has been verified numerically. The relationship between static and match-based pricing is similar to Proposition 2. Figure A.8 illustrates the thresholds of θ above which match-based prices are lower than static prices; these thresholds are shown under six different Beta distributions, where the solid blue curves are for $b = 1$ and the red dashed curves are for $b = 2$. For both values of b, we choose the values of a such that the corresponding means of the Beta distributions are $1/4, 1/2$ and $3/4$, differentiated by color hues. A general observation from Figure A.8a is that all else being equal, the average match-based price is lower when the market is relatively thin and riders are relatively harder to match. This can be seen when comparing curves with the same value of b. For the same value of b, a higher value of a indicates more high-value riders; therefore, more riders convert to requests, which leads to a higher threshold of θ above which the average match-based price is lower. For similar reasons, when comparing willingness-to-pay distributions with the same mean (for example, $\mathsf{Beta}(1,1)$ and $\mathsf{Beta}(2,2)$), the θ thresholds under $\mathsf{Beta}(2,2)$ are higher when the cost c is low and lower when the cost c is high. This is because there are fewer high-value and low-value riders in $\mathsf{Beta}(2,2)$ compared to those under a uniform willingness-to-pay distribution, and consequently, it is relatively easier to match when c is low and harder to match when c is high. For the comparison between $\lambda^*(0)$ and λ^*, we see from Figure A.8b that the region where *both* match-based prices are lower than the optimal static price shrinks when the percentage of high-value riders increases. Finally, we also numerically verified that average payment and cost per request are lower under match-based pricing compared to the optimal static pricing, across all willingness-to-pay distributions tested.



(a) Thresholds $\bar{\lambda}^* = \lambda^*$ above which $\bar{\lambda}^* \geq \lambda^*$

(b) Thresholds $\lambda^*(0) = \lambda^*$ above which $\lambda^*(0) \geq \lambda^*$

Figure A.8: Comparison between match-based and static prices under different $\mathsf{Beta}(a, b)$ willingness-to-pay distributions.

A.4.2 Risk Aversion

In this section, we discuss a model of match-based pricing where riders are risk-averse rather than risk-neutral. In particular, the rider who arrives when no other request is waiting on the platform is quoted two possible prices, and converts based on the maximum possible payment (see, e.g., [43]).

As in Section 2.3.1, a rider who arrives when there is no other request currently on the platform is quoted an initial price $p'(0) = 1 - \lambda'(0)$ as well as the possibility of a discount to $p(1) = 1 - \lambda(1)$ if she ends up getting matched to a later incoming request. A rider arriving with a request waiting is directly quoted $p(1) = 1 - \lambda(1)$. The rider who arrives when there is no other request currently on the platform converts based on the maximum price $\max\{p'(0), p(1)\}$. Meanwhile, the rider arriving with a request waiting converts based on the price $p(1)$, since she is only quoted one price.

The expected profit is then given by

$$\Pi(\lambda'(0), \lambda(1)) = \frac{\min\{\lambda'(0), \lambda(1)\}}{\min\{\lambda'(0), \lambda(1)\} + \lambda(1) + \theta} \left(\lambda(1)(2 - 2\lambda(1) - c) + \theta(1 - \lambda'(0) - c)\right).$$

Intuitively, risk aversion makes it less likely that riders will convert in state 0, degrading the platform's profits. However, many of the insights are similar to the risk-neutral model. The first-order conditions for optimizing the profit do not have a closed form, so the results that follow are numerical.

Figure A.9 reproduces Figure 2.5, now for risk-averse riders, and compares the prices under static and match-based pricing. As in Figure 2.5, the match-based price when there is one rider waiting is always the lowest. In addition, when c and θ are relatively high (above the solid black curve), the average match-based price is lower than the optimal static price. However, in contrast with Figure 2.5, the region of lower prices is not as generous. Moreover, for c and θ sufficiently low (crosshatched area), match-based pricing and static pricing are equivalent.

Similarly Figure A.10 reproduces Figure 2.6 for risk-averse riders. Corroborating our earlier results, Figure A.10a shows that match-based pricing has a larger profitable region than static pricing, and Figure A.10b shows that the improvement in payment and cost per request is most pronounced for high c and θ, when shared rides are most challenging to operate.

A.4.3 Batch Window

In this section, we test whether the analytical results derived in Section 2.3.1 hold when the platform uses a fixed batching window to dispatch. That is, when a rider arrives and makes a request, the platform sets a batching window of L time periods. During this window, if another rider arrives and makes a request, the platform dispatches the two requests together. If the first request's patience elapses before a second request is made, the first request abandons the platform, and the platform loses the associated revenue. Otherwise, the request is dispatched alone at the end of the batching window.

Concretely, a rider who arrives when there is no request waiting on the platform is quoted price $1 - \mu(0)$, and converts to a request at rate $\mu(0)$. Upon conversion, a batching

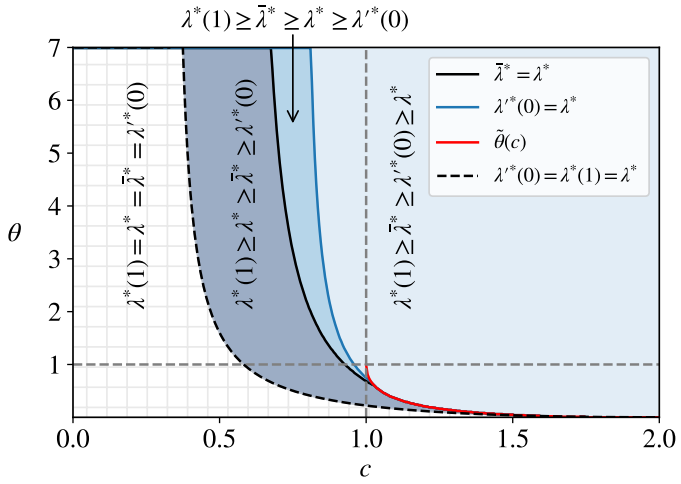
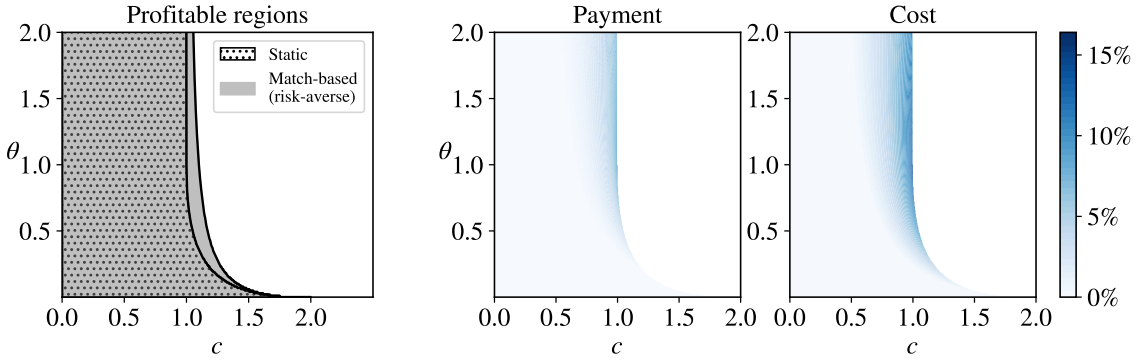


Figure A.9: Comparisons between optimal match-based prices and static prices under the risk-averse model.



(a) Profitable regions of policies

(b) Percentage decrement of payment and cost per request

Figure A.10: Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing). All results are under the risk-averse model.

window of L time periods begins. We divide the L time periods into $T = \lfloor L(1+\theta) \rfloor$ discrete sub-periods, each of length $1/(1+\theta)$ where θ is the reneging rate and 1 is the arrival rate of the rider. Each sub-period t is associated with a quoted price $1 - \mu(t)$. During sub-period t, the probability that another rider will arrive and make a request is then $\Lambda\mu(t)$, where $\Lambda = 1/(1+\theta)$. The probability that the waiting request's patience will elapse is $\zeta = \theta/(1+\theta)$. The system can be modeled as a discrete-time Markov chain with $T+1$ states: state $t = 0$ has no requests waiting in the system, and states $t = 1, \ldots, T$ each have one request waiting.

The expected profit is then

$$\Pi(\boldsymbol{\mu}) = \sum_{t=1}^{T} \underbrace{\pi_t \Lambda\mu(t)(2 - \mu(0) - \mu(t) - c)}_{\substack{\text{profit of two matched riders} \\ \text{dispatched in period } t}} + \underbrace{\pi_T(1 - \Lambda\mu(T) - \zeta)(1 - \mu(0) - c)}_{\substack{\text{profit of one unmatched rider} \\ \text{dispatched in period } T}},$$

where $(\pi_t)_{t=0}^T$ is the steady-state distribution.

We compute the optimal conversion rates $\boldsymbol{\mu} = (\mu^*(t))_{t=0}^T$ by solving the following set of optimality equations where g is the optimal average profit per period:

$$V(t) + g = \max_{\boldsymbol{\mu} \in [0,1]^{T+1}} \left(\underbrace{\Lambda\mu(t)\,(1 - \mu(t) - c + V(0))}_{\substack{\text{second rider makes a request,} \\ \text{two riders dispatched}}} + \underbrace{\zeta(V(0) - (1 - \mu(0)))}_{\substack{\text{first rider's patience elapses,} \\ \text{revenue is lost}}} \right.$$
$$\left. + \underbrace{(1 - \Lambda\mu(0) - \zeta)\,V(t+1)}_{\text{no event occurs}} \right), \quad \forall t = 1, \ldots, T-1, \tag{A.51a}$$

$$V(0) + g = \max_{\boldsymbol{\mu} \in [0,1]^{T+1}} \left(\underbrace{\Lambda\mu(0)\,(1 - \mu(0) + V(1))}_{\text{first rider makes a request}} + \underbrace{(1 - \Lambda\mu(0))\,V(0)}_{\text{no event occurs}} \right), \tag{A.51b}$$

$$V(T) + g = \max_{\boldsymbol{\mu} \in [0,1]^{T+1}} \left(\underbrace{\Lambda\mu(T)\,(1 - \mu(T) - c + V(0))}_{\substack{\text{second rider makes a request,} \\ \text{two riders dispatched}}} + \underbrace{\zeta(V(0) - (1 - \mu(0)))}_{\substack{\text{first rider's patience elapses,} \\ \text{revenue is lost}}} \right.$$
$$\left. + \underbrace{(1 - \Lambda\mu(T) - \zeta)\,(V(0) - c)}_{\substack{\text{no event occurs,} \\ \text{one rider dispatched}}} \right). \tag{A.51c}$$

Given optimal value functions, we can compute the optimal conversions $\mu^*(t)$ for $t = 1, \ldots, T$ as

$$\mu^*(t) = \frac{1 - c + V(0) - V(t+1)}{2} \quad \forall t = 1, \ldots, T,$$
$$\mu^*(T) = \frac{1}{2},$$

using the first-order conditions for (A.51a) and (A.51c). Meanwhile, to compute the optimal

$\mu^*(0)$, we simply use a line search. We solve the optimal value functions using value iteration.

Figure A.11 and Figure A.12 reproduce Figure 2.5 and Figure 2.6, now for the batch window model with $L = 3$. (Note that the step pattern in Figure A.11 and Figure A.12 arises from the discrete number of sub-periods $T = \lfloor L(1+\theta) \rfloor$; $\bar{\mu}^*(1)$ in Figure A.11 is the average conversion rate of riders arriving when there is one request waiting, as an analogy to $\lambda^*(1)$ in Figure 2.5.) As in Figure 2.6a, Figure A.12a shows that match-based pricing has a larger profitable region. However, the shape of the profitable regions in Figure A.12a highlight a key distinction: for high c, there is now a threshold of θ *above* which the platform is profitable, and below which the platform is unprofitable. This is because when cost is high, the platform in fact *benefits* from high impatience: the first request is more likely to abandon the platform before the end of the batch window, saving the platform a costly solo dispatch.

To compare the prices under each policy, Figure A.11 and Figure A.12b show that match-based pricing quotes lower average prices and induces lower payment and cost per request under the high-c, low-θ regime when static pricing cannot be profitable. This is similar to the message from Section 2.3.1: match-based pricing can benefit both the platform *and* riders under settings when shared rides are most challenging to operate. One can think of increasing the patience parameter θ of our original model in Section 2.3.1 as a combined effect of changing the patience of the rider θ and the length of the batching window of the platform L in this model so that requests are more likely to be dispatched solo.

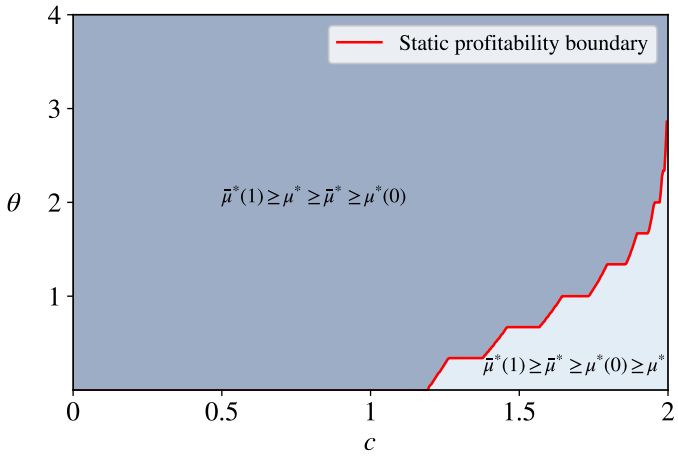
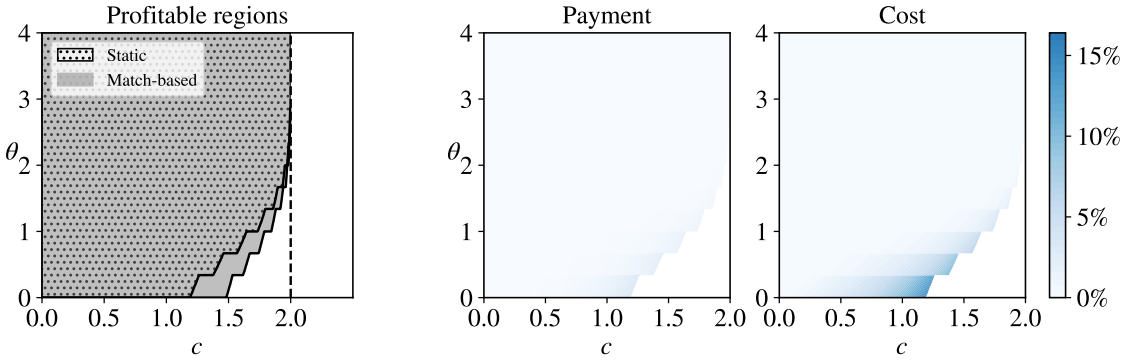


Figure A.11: Comparisons between optimal match-based prices and static prices under the batch window model.



(a) Profitable regions of policies

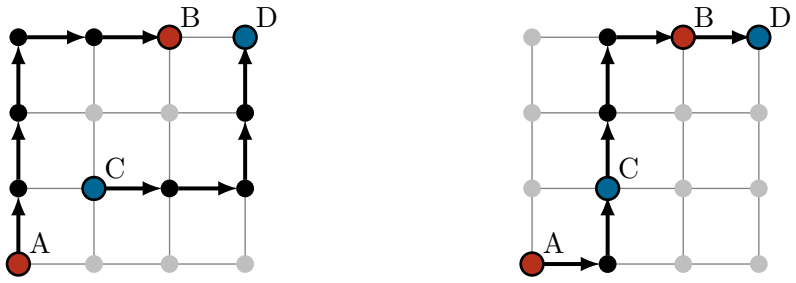
(b) Percentage decrement of payment and cost per request

Figure A.12: Profitable regions of different pricing policies; percentage decrement of payment and cost per request (match-based pricing over static pricing). All results are under the batch window model.

A.5 Additional Results

A.5.1 Allocating Dispatching Cost to Requests

Consider the following dispatch (i, j) where rider type i goes from A to B while rider type j goes from C to D. Figures A.13a and A.13b illustrate the minimum-cost routings when i and j are dispatched solo and matched respectively. In the matched case (Figure A.13b), the total dispatching cost is allocated to rider type i using the cost of $A \to C$, plus *half* of the cost of $C \to B$ as it is shared by both riders. Similarly, the cost is allocated to rider type j using *half* of $C \to B$, plus $B \to D$. In the iterative pricing approach, the average cost $\bar{c}_i$ of a particular rider type i is then estimated by averaging over all such allocated costs through past dispatching data.



(a) Not matched

(b) Matched

Figure A.13: Cost allocation example.

A.5.2 Recovering Equivalent Match-Based Prices with Multiple O-D Pairs

We show how equivalent match-based prices, discussed at the beginning of Section 2.3.1, can be (easily) recovered from the optimal pricing and matching policies $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$ presented in Section 2.3.2. Following the description at the end of Section 2.3.2, let $[p'_i]_{i\in[N]}$ be the initial prices quoted to each type of rider when they arrive but have not finalized a match yet. Let $[\lambda'_i]_{i\in[N]}$ be their corresponding request conversion rates. Moreover, let $[p_{i,j}]_{i,j\in[N]}$ be the discounted price rider type i pays once matched with rider type j, and $p_{i,j} \neq p_{j,i}$ in general. Similarly, let $[\lambda_{i,j}]_{i,j\in[N]}$ be the corresponding request conversion rates. From the matching and pricing policies (2.14) and (2.15), we know when an incoming request type i is immediately matched with type j, type i is charged at a price with conversion rate $\lambda^*_{i,j}$ that solves: $\lambda^*_{i,j} \in \arg\max_{\lambda\in[0,\Lambda_i]:p_i^1(\lambda)\geq 0}\lambda(p_i^1(\lambda) - (c_{(i,j)} - \alpha_j^*))$; and when incoming request type i is let to wait first in the system, it is charged at a price with conversion rate λ_i^* that solves: $\lambda_i^* \in \arg\max_{\lambda\in[0,\Lambda_i]:p_i^0(\lambda)\geq 0}\lambda(p_i^0(\lambda) - \alpha_i^*)$.

We simply let $\lambda_{i,j} = \lambda^*_{i,j}$, $\forall i, j \in [N]$ for the discounted match-based prices with successful matches. To recover $[\lambda'_i]_{i\in[N]}$, let $[\pi^*_{\mathbf{s}}]_{\mathbf{s}\in\mathcal{S}}$ be the steady-state distribution, $P^*_{i,j}$ be the probability a rider type i is matched with a rider type j conditioning on rider type i is left waiting in the system first upon arrival, and P_i^* be the probability a rider type i is dispatched solo, under the optimal pricing and matching policies $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$.

$$P^*_{i,j} = \frac{\sum_{\mathbf{s}\in\mathcal{S}} \pi^*_{\mathbf{s}} \cdot \lambda^*_j(\mathbf{s}) \cdot \mathbf{1}\{\phi^*_j(\mathbf{s}) = i\}}{\sum_{\mathbf{s}\in\mathcal{S}} \pi^*_{\mathbf{s}} \cdot \lambda^*_i(\mathbf{s}) \cdot \mathbf{1}\{\phi^*_i(\mathbf{s}) = 0\}}, \qquad \forall i, j \in [N],$$

$$P_i^* = 1 - \sum_{j\in[N]} P^*_{i,j}, \qquad \forall i \in [N].$$

Alternatively, $P^*_{i,j}$ and P_i^* can simply be estimated with Monte Carlo. The remaining match-based prices $[p'_i]_{i\in[N]}$ solves the following N separate linear equations.

$$p_i^0(\lambda_i^*) = p'_i \cdot P_i^* + \sum_{j\in[N]} p_i^1(\lambda^*_{i,j}) \cdot P^*_{i,j}, \qquad \forall i \in [N].$$

Note that in the case where P_i^* is zero, there is no need to define p'_i as it will be offered to riders with probability zero.

A.5.3 Other Results Based on Chicago Ridesharing Data

This subsection includes additional computational results based on Chicago ridesharing data. Table A.2 is the corresponding training results for the peak hours on Mondays discussed in Section 2.4. In addition to metrics presented in the test results, Table A.2 also presents "Static UB", the optimal value of the fluid formulation described in Appendix A.2, which is an upper bound of the profit under static pricing. We also report the relative difference between "Static UB" and the profit under match-based pricing which is still significantly higher.

Cost ($/mile)		$c = 0.7$				$c = 0.9$				$c = 1.1$			
Sojourn Time $1/\theta$ (min)		0.5	1	3	5	0.5	1	3	5	0.5	1	3	5
Profit ($/min)	Iterative	3.121	3.417	4.347	4.981	0.390	0.462	0.889	1.324	0.000	0.000	0.000	0.000
	Static	3.173	3.528	4.481	5.039	0.414	0.549	1.181	1.609	0.000	0.000	0.000	0.000
	Static UB	3.340	3.753	4.812	5.454	0.449	0.631	1.312	1.807	0.000	0.000	0.000	0.000
	Match	3.379	3.807	4.807	5.437	0.594	0.844	1.560	2.056	0.000	0.000	0.165	0.326
	Match vs. Iterative	8.3%	11.4%	10.6%	9.2%	52.3%	82.9%	75.6%	55.4%	-	-	∞	∞
	Match vs. Static	6.5%	7.9%	7.3%	7.9%	43.3%	53.8%	32.1%	27.8%	-	-	∞	∞
	Match vs. Static UB	1.2%	1.4%	1.2%	-0.3%	32.2%	33.7%	19.0%	13.8%	-	-	∞	∞
Quoted Price ($/mile · rider)	Iterative	0.816	0.800	0.768	0.754	0.931	0.918	0.884	0.861	1.000	1.000	1.000	1.000
	Static	0.809	0.789	0.749	0.731	0.920	0.895	0.844	0.819	1.000	1.000	1.000	1.000
	Match	0.801	0.781	0.742	0.723	0.904	0.880	0.833	0.813	1.000	1.000	0.935	0.899
	Match vs. Iterative	-1.8%	-2.4%	-3.4%	-4.1%	-2.9%	-4.1%	-5.8%	-5.5%	0.0%	0.0%	-6.5%	-10.1%
	Match vs. Static	-1.0%	-1.0%	-1.0%	-1.1%	-1.8%	-1.7%	-1.3%	-0.7%	0.0%	0.0%	-6.5%	-10.1%
Quoted Disutility ($/mile · rider)	Iterative	0.837	0.829	0.806	0.793	0.942	0.937	0.915	0.897	1.000	1.000	1.000	1.000
	Static	0.830	0.816	0.786	0.772	0.932	0.916	0.876	0.856	1.000	1.000	1.000	1.000
	Match	0.822	0.807	0.776	0.760	0.920	0.902	0.865	0.847	1.000	1.000	0.959	0.927
	Match vs. Iterative	-1.8%	-2.7%	-3.7%	-4.1%	-2.3%	-3.7%	-5.4%	-5.6%	0.0%	0.0%	-4.1%	-7.3%
	Match vs. Static	-1.0%	-1.1%	-1.2%	-1.5%	-1.2%	-1.5%	-1.2%	-1.0%	0.0%	0.0%	-4.1%	-7.3%
Payment ($/mile · request)	Iterative	0.815	0.797	0.762	0.747	0.929	0.911	0.862	0.833	-	-	-	-
	Static	0.807	0.783	0.741	0.723	0.914	0.881	0.817	0.793	-	-	-	-
	Match	0.785	0.761	0.721	0.704	0.867	0.834	0.786	0.769	-	-	0.830	0.812
	Match vs. Iterative	-3.6%	-4.5%	-5.4%	-5.8%	-6.6%	-8.5%	-8.9%	-7.7%	-	-	-	-
	Match vs. Static	-2.7%	-2.8%	-2.7%	-2.7%	-5.1%	-5.3%	-3.8%	-2.9%	-	-	-	-
Realized Disutility ($/mile · request)	Iterative	0.834	0.824	0.799	0.785	0.938	0.927	0.891	0.867	-	-	-	-
	Static	0.828	0.811	0.778	0.763	0.927	0.905	0.852	0.831	-	-	-	-
	Match	0.809	0.791	0.759	0.743	0.890	0.864	0.824	0.807	-	-	0.867	0.848
	Match vs. Iterative	-3.0%	-4.1%	-5.0%	-5.3%	-5.1%	-6.9%	-7.5%	-6.9%	-	-	-	-
	Match vs. Static	-2.2%	-2.5%	-2.5%	-2.6%	-3.9%	-4.5%	-3.3%	-2.9%	-	-	-	-
Throughput (requests/min)	Iterative	4.633	4.834	5.472	5.851	1.644	1.781	2.408	2.916	0.000	0.000	0.000	0.000
	Static	4.823	5.199	6.041	6.456	1.930	2.355	3.509	4.080	0.000	0.000	0.000	0.000
	Match	5.060	5.459	6.312	6.776	2.258	2.749	3.798	4.337	0.000	0.000	1.182	2.082
	Match vs. Iterative	9.2%	12.9%	15.4%	15.8%	37.4%	54.4%	57.7%	48.7%	-	-	∞	∞
	Match vs. Static	4.9%	5.0%	4.5%	5.0%	17.0%	16.7%	8.2%	6.3%	-	-	∞	∞
Match Rate	Iterative	0.227	0.338	0.502	0.569	0.113	0.205	0.413	0.524	-	-	-	-
	Static	0.244	0.354	0.522	0.605	0.149	0.287	0.498	0.582	-	-	-	-
	Match	0.306	0.408	0.570	0.638	0.307	0.422	0.585	0.633	-	-	0.641	0.655
Cost Efficiency	Iterative	0.062	0.098	0.180	0.221	0.030	0.057	0.143	0.203	-	-	-	-
	Static	0.067	0.108	0.190	0.229	0.039	0.081	0.179	0.221	-	-	-	-
	Match	0.098	0.142	0.222	0.260	0.101	0.150	0.232	0.268	-	-	0.272	0.293
Detour Rate (over matched requests)	Iterative	0.067	0.062	0.048	0.038	0.066	0.060	0.042	0.034	-	-	-	-
	Static	0.066	0.058	0.046	0.039	0.065	0.064	0.043	0.037	-	-	-	-
	Match	0.054	0.047	0.037	0.030	0.050	0.043	0.034	0.025	-	-	0.020	0.016

Table A.2: Training results on Chicago data ($\beta_0 = 0.05, \beta_1 = 0.675$).

Appendix B

SUPPLEMENTARY MATERIALS FOR CHAPTER 3

B.1 Modeling Combined Pre-Trip and On-Trip Matching

In this section, we extend our on-trip matching model to a *combined matching model* that permits both pre-trip and on-trip matching. When a new rider converts and is not immediately matched with another rider, the platform first initiates a waiting window of at most $T \in \mathbb{Z}_{\geq 0}$ time periods at their origin, during which it attempts to perform *pre-trip matching*. The value of T is the maximum allowable time before the rider must begin their trip. If T time periods have elapsed and the rider is still solo, the platform dispatches a vehicle for a solo trip. Any successful matching that occurs after a rider has departed their origin is referred to as *on-trip matching*.

Consistent with the pure on-trip matching model, we represent the state of a solo rider using the tuple (i, u), where $i \in [N]$ denotes the rider type and integer u tracks the temporal state. The range of u is extended to $-T + 1 \leq u \leq \ell_i - 1$. When $u \leq 0$, the rider is still at the origin with $|u|$ waiting periods remaining. When $u > 0$, the rider has already departed the origin, and u denotes the number of time periods elapsed since departure. The set $\mathcal{C}$ is therefore defined as $\{(i, u)\}_i \in [N], -T + 1 \leq u \leq \ell_i - 1, u \in \mathbb{Z}$.

Since the platform cannot match two on-trip riders (because they are already on separate vehicles), any valid matching must involve at least one rider who is either new or in the pre-trip phase. We can therefore still let $\ell^u_{i,j}$ denote the shared trip length, when a new or pre-trip rider of type $i \in [N]$ is matched with a solo rider $(j, u) \in \mathcal{C}$. Because a solo rider (j, u) with $u \leq 0$ remains at their origin O_j, the shared trip length is invariant during the pre-trip phase; thus, $\ell^u_{i,j} = \ell^v_{i,j}$ for all $u, v \leq 0$. Let $\mathcal{N}^+_{j,u}$ be the set of rider types compatible with a solo rider $(j, u) \in \mathcal{C}$. Similarly to the pure on-trip matching case, we set two conditions for a solo rider (j, u) to be compatible with a new or pre-trip rider of type i: (i) $\ell^u_{i,j} < \ell_i + \ell_j - \max\{0, u\}$; and (ii) (j, u) is not on any shortest path from O_i (exclusive) to D_j. One can easily verify that a pre-trip rider ($u \leq 0$) is always compatible with a new or pre-trip rider of the same type, that is, $i \in \mathcal{N}^+_{i,u}$ holds for all $i \in [N]$ and $u \leq 0$.

Finally, note that pure pre-trip matching and pure on-trip matching can be seen as special cases of the combined matching model. To disable pre-trip matching, the platform can let $T = 0$, such that the rider departs immediately and no pre-trip matching is possible. To disable on-trip matching, the platform can set $\mathcal{N}^+_{i,u} = \emptyset$ for all $u > 0$.

B.1.1 Fluid Model

Under the combined model, the optimal pricing and matching problem is formulated as an average-reward, discrete-time MDP. Assuming a match always occurs between a new rider and an existing solo rider, the MDP and fluid relaxation extend directly from Sections 3.3.1

and 3.3.2. Specifically, in the fluid relaxation, $C(\boldsymbol{\lambda})$ is now given by

$$
\begin{aligned}
C(\boldsymbol{\lambda}) = \quad & \min_{\boldsymbol{x},\boldsymbol{y}} \quad \sum_{i\in[N]} \sum_{u=0}^{\ell_i-1} c y_i^u + \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} c\ell_{i,j}^u x_{i,j}^u \\
& \text{s.t.} \quad \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} x_{i,j}^u + y_i^{-T} = \Lambda_i \lambda_i, \qquad \forall i \in [N], && \text{(B.1a)} \\
& \qquad \sum_{i\in\mathcal{N}_{j,u}^+} x_{i,j}^u + y_j^u = y_j^{u-1}, \qquad \forall (j,u) \in \mathcal{C}, && \text{(B.1b)} \\
& \qquad \Lambda_i \lambda_i y_j^u \geq \left(1 - \sum_{k\in\mathcal{N}_{j,u}^+} \Lambda_k \lambda_k\right) x_{i,j}^u, \qquad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+, \\
& \qquad x_{i,j}^u \geq 0, \qquad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+, \\
& \qquad y_i^u \geq 0, \qquad \forall i \in [N], -T \leq u \leq \ell_i - 1, u \in \mathbb{Z},
\end{aligned}
$$

where we now use y_i^{-T} to denote the average rate of new riders of type i that are not immediately matched and become solo riders, for all $i \in [N]$. The conclusion of Proposition 4 still holds in this context: the optimal objective value of the fluid relaxation provides an upper bound on the optimal long-run average profit of the MDP model.

B.1.2 Pricing and Matching

To determine the pricing policy $\hat{\boldsymbol{\lambda}}$ for the combined matching model, we continue to use the MM algorithm outlined in Section 3.3.3.

To implement the matching policy, we extend the dual-based policy from Section 3.3.3 to a generalized setting that accommodates matching beyond just new riders.

Specifically, let $\gamma_i(\hat{\boldsymbol{\lambda}})$ and $\xi_i^u(\hat{\boldsymbol{\lambda}})$ denote the optimal dual variables associated with constraints (B.1a) and (B.1b), respectively. At time t, let $\mathcal{C}_t$ represent the set of all riders currently in the system (including both waiting and on-trip riders). Any newly arrived rider of type i at time t is initialized with state $(i, -T)$ and also included in $\mathcal{C}_t$, and for notational convenience, let $\xi_i^{-T}(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}})$.

We determine the matching decisions at time t by solving the following non-bipartite matching problem:

$$
\begin{aligned}
& \min_{\boldsymbol{z}} \quad \sum_{\substack{(i,u),(j,v)\in\mathcal{C}_t:\\ u<v}} z_{i,j}^{u,v} \left(c\ell_{i,j}^v - \xi_i^u(\hat{\boldsymbol{\lambda}}) - \xi_j^u(\hat{\boldsymbol{\lambda}})\right) && \text{(B.2a)} \\
& \text{s.t.} \quad \sum_{\substack{(j,v)\in\mathcal{C}_t:\\ u<v}} z_{i,j}^{u,v} + \sum_{\substack{(j,v)\in\mathcal{C}_t:\\ u>v}} z_{j,i}^{v,u} \leq 1, \qquad \forall (i,u) \in \mathcal{C}_t, && \text{(B.2b)} \\
& \qquad z_{i,j}^{u,v} = 0, \qquad \forall (i,u),(j,v) \in \mathcal{C}_t : u < v < 0, && \text{(B.2c)}
\end{aligned}
$$

$$z_{i,j}^{u,v} = 0, \qquad \forall (i,u),(j,v) \in \mathcal{C}_t : 0 < u < v, \tag{B.2d}$$

$$z_{i,j}^{u,v} \in \{0,1\}, \qquad \forall (i,u),(j,v) \in \mathcal{C}_t : u < v. \tag{B.2e}$$

In (B.2), the binary decision variable $z_{i,j}^{u,v}$ equals 1 if rider (i,u) is matched with rider (j,v) at time t (where $u < v$ avoids double counting), and 0 otherwise. The objective function (B.2a) minimizes the total generalized cost of the matching. Since we never match two on-trip riders together (enforced by constraint (B.2d)), the shared trip length for a match between (i,u) and (j,v) is given by $\ell_{i,j}^{v}$, and thus the associated generalized cost is $c\ell_{i,j}^{v} - \xi_i^u(\hat{\boldsymbol{\lambda}}) - \xi_j^v(\hat{\boldsymbol{\lambda}})$.

Constraint (B.2b) ensures that each rider is matched to at most one other rider. Constraint (B.2c) restricts pre-trip matching to cases where at least one rider is ready to depart (i.e., $v = 0$). This constraint is based on the logic that if both riders still have waiting time remaining ($u < v < 0$), it is always better to defer the match to later time periods to preserve flexibility. Finally, Constraint (B.2d) prohibits matching two riders who have both already departed the origin ($0 < u < v$), because they are already on separate vehicles.

B.2 Proofs

B.2.1 Proof of Proposition 4

Denote $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$ as the optimal pricing and matching policies under the MDP. For all $(j,u) \in \mathcal{C}$ and $i \in \mathcal{N}_{j,u}^+$, define the counting process $\{M_t^*(i,(j,u))\}_{t\geq 0}$ as the accumulated number of matches between i and (j,u), up to time t under the optimal policies $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$. Define the counting process $\{A_t^*(i,u)\}_{t\geq 0}$ as the accumulated number of events that solo riders $(i,u) \in \mathcal{C}$ continue solo and move to the next state. For notational convenience, we also define $\{A_t^*(i)\}_{t\geq 0}$ as the accumlated number of the requests of type-$i \in [N]$ riders, and define $\{A_t^*(i,0)\}_{t\geq 0}$ as the accumulated number of events that new type-$i \in [N]$ riders do not get matched immediately and become solo riders, under the optimal policies $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$.

Then we define the steady flow rates under $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$ as

$$\begin{aligned}
\hat{x}_{i,j}^u &:= \lim_{t\to\infty} 1/t \cdot \mathbb{E}[M_t^*(i,(j,u))], && \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+, \\
\hat{y}_i^u &:= \lim_{t\to\infty} 1/t \cdot \mathbb{E}[A_t^*(i,u)], && \forall (i,u) \in \mathcal{C}, \\
\hat{y}_i^0 &:= \lim_{t\to\infty} 1/t \cdot \mathbb{E}[A_t^*(i,0)], && \forall i \in [N].
\end{aligned}$$

We first prove that $\hat{x}_{i,j}^u$, $\hat{y}_i^u$ and $\hat{y}_i^0$ gives a feasible solution to (CB) when $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$. According to the definitions, the following equations hold:

$$\sum_{\substack{(j,u)\in\mathcal{C}: \\ i\in\mathcal{N}_{j,u}^+}} M_t^*(i,(j,u)) + A_t^*(i,0) = A_t^*(i), \qquad \forall i \in [N], t \in \mathbb{N}, \tag{B.3}$$

$$\sum_{i\in\mathcal{N}_{j,u}^+} M_t^*(i,(j,u)) + A_t^*(j,u) = A_{t-1}^*(j,u-1), \qquad \forall (j,u) \in \mathcal{C}, t \in \mathbb{N}^+, \tag{B.4}$$

where (B.3) is because the new requests are either immediately matched with existing solo riders, or become solo riders; and (B.4) is because all riders in state (j,u) come from state $(j,u-1)$, and are either matched with new riders or continuing solo. Taking the long-run time-average expected value for (B.3) and (B.4), we get

$$\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} \hat{x}_{i,j}^u + \hat{y}_i^0 = \lim_{t\to\infty}\frac{1}{t}\mathbb{E}[A_t^*(i)] = \Lambda_i\lambda_i^*, \qquad \forall i\in[N], \tag{B.5}$$

$$\sum_{i\in\mathcal{N}_{j,u}^+} \hat{x}_{i,j}^u + \hat{y}_j^u = \hat{y}_j^{u-1}, \qquad \forall (j,u)\in\mathcal{C}, \tag{B.6}$$

thus (3.2b) and (3.2c) are satisfied.

To show that (3.2d) holds, for each $(j,u)\in\mathcal{C}$, $i\in\mathcal{N}_{j,u}^+$, and $t\in\mathbb{N}$, the expected increment of the number of matches between successive time steps t and $t+1$ is:

$$\begin{aligned}
&\mathbb{E}[M_{t+1}^*(i,(j,u)) - M_t^*(i,(j,u))]\\
=&\sum_{\mathbf{s}\in\mathcal{S}} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\mathbb{E}[M_{t+1}^*(i,(j,u)) - M_t^*(i,(j,u))|\mathbf{s}_{t+1}=\mathbf{s}]\\
\overset{(a)}{=}&\sum_{\mathbf{s}\in\mathcal{S}} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\Pr[M_{t+1}^*(i,(j,u)) - M_t^*(i,(j,u)) = 1|\mathbf{s}_{t+1}=\mathbf{s}]\\
\overset{(b)}{\leq}&\sum_{\mathbf{s}\in\mathcal{S}:\mathrm{s}_{j,u}=1} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\Pr[A_{t+1}^*(i) - A_t^*(i) = 1|\mathbf{s}_{t+1}=\mathbf{s}]\\
=&\sum_{\mathbf{s}\in\mathcal{S}:\mathrm{s}_{j,u}=1} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\Lambda_i\lambda_i^*,
\end{aligned} \tag{B.7}$$

where (a) is because $M_{t+1}^*(i,(j,u)) - M_t^*(i,(j,u)) \in \{0,1\}$, and (b) is because $M_{t+1}^*(i,(j,u)) - M_t^*(i,(j,u)) = 1$ means that there must be (i) an existing solo rider (j,u) at time $t+1$ and (ii) a new type-i rider converts at time $t+1$.

Meanwhile, for every $(j,u)\in\mathcal{C}$ and $t\in\mathbb{N}$, the expected increment of counts of riders continuing solo and moving to the next state is:

$$\begin{aligned}
&\mathbb{E}[A_{t+1}^*(j,u) - A_t^*(j,u)]\\
=&\sum_{\mathbf{s}\in\mathcal{S}} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\mathbb{E}[A_{t+1}^*(j,u) - A_t^*(j,u)|\mathbf{s}_{t+1}=\mathbf{s}]\\
=&\sum_{\mathbf{s}\in\mathcal{S}} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\Pr[A_{t+1}^*(j,u) - A_t^*(j,u) = 1|\mathbf{s}_{t+1}=\mathbf{s}]\\
=&\sum_{\mathbf{s}\in\mathcal{S}:\mathrm{s}_{j,u}=1} \Pr(\mathbf{s}_{t+1}=\mathbf{s})\cdot\Pr\left[\sum_{k\in\mathcal{N}_{j,u}^+} M_{t+1}^*(k,(j,u)) - \sum_{k\in\mathcal{N}_{j,u}^+} M_t^*(k,(j,u)) = 0\middle|\mathbf{s}_{t+1}=\mathbf{s}\right]
\end{aligned}$$

$$\overset{(a)}{\geq} \sum_{\mathbf{s}\in\mathcal{S}:\mathrm{s}_{j,u}=1} \Pr(\mathbf{s}_{t+1}=\mathbf{s}) \cdot \mathbb{E}\left[1 - \sum_{k\in\mathcal{N}^+_{j,u}} \Lambda_k \lambda^*_k \middle| \mathbf{s}_{t+1}=\mathbf{s}\right]$$

$$= \sum_{\mathbf{s}\in\mathcal{S}:\mathrm{s}_{j,u}=1} \Pr(\mathbf{s}_{t+1}=\mathbf{s}) \cdot \left(1 - \sum_{k\in\mathcal{N}^+_{j,u}} \Lambda_k \lambda^*_k\right), \tag{B.8}$$

where (a) is because when there are no compatible riders $k \in \mathcal{N}^+_{j,u}$ who convert at time $t+1$, the existing solo rider (j,u) cannot get matched and thus must continue solo.

Combining (B.7) and (B.8), we get

$$\mathbb{E}[A^*_{t+1}(j,u) - A^*_t(j,u)] \geq \frac{1 - \sum_{k\in\mathcal{N}^+_{j,u}} \Lambda_k \lambda^*_k}{\Lambda_i \lambda^*_i} \mathbb{E}[M^*_{t+1}(i,(j,u)) - M^*_t(i,(j,u))],$$

which holds for all $(j,u) \in \mathcal{C}, i \in \mathcal{N}^+_{j,u}$ and $t \in \mathbb{N}$.

Summing over $t \in \mathbb{N}$, and taking the long-run time-average value, we get (3.2d). Therefore, $\hat{x}^u_{i,j}$, $\hat{y}^u_i$ and $\hat{y}^0_i$ gives a feasible solution to (CB) when $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$.

In the second part of the proof, we show that the fluid objective value under prices $\boldsymbol{\lambda}^*$ and fluid variables $\hat{x}^u_{i,j}$, $\hat{y}^u_i$ and $\hat{y}^0_i$ is equivalent to the optimal long-run average profit of the MDP model. Under $\boldsymbol{\lambda}^*$ and $\boldsymbol{\phi}^*$, the total profit up to time t is given as follows

$$\Pi_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*} = \sum_{n\in[A_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}]} \left[p_{i_n}(\lambda^*_{i_n}) - c\ell_{i_n}\right] + \sum_{n\in[M_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}]} \left[p_{i_n}(\lambda^*_{i_n}) + p_{j_n}(\lambda^*_{j_n}) - cu_n - c\ell^{u_n}_{i_n,j_n}\right].$$

Since $A_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}$ is the total number of riders who end up solo and $M_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}$ is the total number of matches, we have

$$A_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*} = \sum_{i\in[N]} A^*_{t-1}(i,\ell_i - 1),$$
$$M_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*} = \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} M^*_t(i,(j,u)),$$

thus we can reformulate $\Pi_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}$ into

$$\Pi_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*} = \sum_{i\in[N]} A^*_{t-1}(i,\ell_i-1)\left[p_i(\lambda^*_i) - c\ell_i\right] + \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} M^*_t(i,(j,u))\left[p_i(\lambda^*_i) + p_j(\lambda_j) - cu - c\ell^u_{i,j}\right].$$

Taking the long-run time-average value, we get

$$\Pi = \lim_{T\to+\infty} \frac{1}{t}\mathbb{E}[\Pi_t^{\boldsymbol{\lambda}^*,\boldsymbol{\phi}^*}] = \sum_{i\in[N]} \hat{y}_i^{\ell_i - 1}\left[p_i(\lambda^*_i) - c\ell_i\right] + \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}^u_{i,j}\left[p_i(\lambda^*_i) + p_j(\lambda^*_j) - cu - c\ell^u_{i,j}\right],$$

and we further reformulate the first term by iteratively applying (B.6) as follows. (For ease of exposition, we present the derivation assuming $\ell_i \geq 3$.)

$$
\begin{aligned}
&\sum_{i\in[N]} \hat{y}_i^{\ell_i-1}\left[p_i(\lambda_i^*) - c\ell_i\right] \\
&= \sum_{i\in[N]} \hat{y}_i^{\ell_i-1}\left[p_i(\lambda_i^*) - c(\ell_i-1)\right] - c\sum_{i\in[N]} \hat{y}_i^{\ell_i-1} \\
&= \sum_{i\in[N]} \left(\hat{y}_i^{\ell_i-2} - \sum_{j\in\mathcal{N}^+_{i,\ell_i-1}} \hat{x}_{j,i}^{\ell_i-1}\right)\left[p_i(\lambda_i^*) - c(\ell_i-1)\right] - c\sum_{i\in[N]} \hat{y}_i^{\ell_i-1} \\
&\qquad\qquad \text{(apply (B.6) for } u = \ell_i - 1) \\
&= \sum_{i\in[N]} \hat{y}_i^{\ell_i-2}\left[p_i(\lambda_i^*) - c(\ell_i-1)\right] - \sum_{i\in[N]}\sum_{j\in\mathcal{N}^+_{i,\ell_i-1}} \hat{x}_{j,i}^{\ell_i-1}\left[p_i(\lambda_i^*) - c(\ell_i-1)\right] - c\sum_{i\in[N]} \hat{y}_i^{\ell_i-1} \\
&= \sum_{i\in[N]} \hat{y}_i^{\ell_i-3}\left[p_i(\lambda_i^*) - c(\ell_i-2)\right] - \sum_{i\in[N]}\sum_{u=\ell_i-2}^{\ell_i-1}\sum_{j\in\mathcal{N}^+_{i,u}} \hat{x}_{j,i}^{u}\left[p_i(\lambda_i^*) - cu\right] - c\sum_{i\in[N]}\sum_{u=\ell_i-2}^{\ell_i-1} \hat{y}_i^{u} \\
&\qquad\qquad \text{(apply (B.6) for } u = \ell_i - 2) \\
&= \cdots \\
&= \sum_{i\in[N]} \hat{y}_i^{0} p_i(\lambda_i^*) - \sum_{i\in[N]}\sum_{u=1}^{\ell_i-1}\sum_{j\in\mathcal{N}^+_{i,u}} \hat{x}_{j,i}^{u}\left[p_i(\lambda_i^*) - cu\right] - c\sum_{i\in[N]}\sum_{u=0}^{\ell_i-1} \hat{y}_i^{u} \\
&= \sum_{i\in[N]} \hat{y}_i^{0} p_i(\lambda_i^*) - \sum_{i\in[N]}\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}_{i,j}^{u}\left[p_j(\lambda_j^*) - cu\right] - c\sum_{i\in[N]}\sum_{u=0}^{\ell_i-1} \hat{y}_i^{u}. \qquad \text{(B.9)}
\end{aligned}
$$

Hence, by substituting $\sum_{i\in[N]} \hat{y}_i^{\ell_i-1}\left[p_i(\lambda_i^*) - c\ell_i\right]$ using (B.9), Π can be expressed as

$$
\begin{aligned}
\Pi &= \sum_{i\in[N]} \hat{y}_i^{0} p_i(\lambda_i^*) - \sum_{i\in[N]}\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}_{i,j}^{u}\left[p_j(\lambda_j^*) - cu\right] - c\sum_{i\in[N]}\sum_{u=0}^{\ell_i-1} \hat{y}_i^{u} \\
&\quad + \sum_{i\in[N]}\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}_{i,j}^{u}\left[p_i(\lambda_i^*) + p_j(\lambda_j^*) - cu - c\ell_{i,j}^{u}\right] \\
&= \sum_{i\in[N]} \hat{y}_i^{0} p_i(\lambda_i^*) + \sum_{i\in[N]}\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}_{i,j}^{u} p_i(\lambda_i^*) - c\sum_{i\in[N]}\sum_{u=0}^{\ell_i-1} \hat{y}_i^{u} - c\sum_{i\in[N]}\sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}^+_{j,u}}} \hat{x}_{i,j}^{u}\ell_{i,j}^{u}
\end{aligned}
$$

$$\overset{(a)}{=} \sum_{i\in[N]} \Lambda_i \lambda_i^* p_i(\lambda_i^*) - c \sum_{i\in[N]} \sum_{u=0}^{\ell_i-1} \hat{y}_i^u - c \sum_{i\in[N]} \sum_{\substack{(j,u)\in\mathcal{C}:\\ i\in\mathcal{N}_{j,u}^+}} \hat{x}_{i,j}^u \ell_{i,j}^u,$$

where (a) is due to (B.5). Therefore, Π is equivalent to the fluid objective value under the feasible solution $\boldsymbol{\lambda}^*$, $\hat{x}_{i,j}^u$, $\hat{y}_i^u$ and $\hat{y}_i^0$, and thus we have $g \geq \Pi$. □

B.2.2 Proof of Proposition 5

The proof of Proposition 5 is built on the following two lemmas.

Lemma 6. *$\forall i, j \in [N]$, $\ell_{i,j}^1 + 1 = \min_{u\in[\ell_j-1]}\{\ell_{i,j}^u + u\}$.*

Proof. Proof of Lemma 6 Suppose, for the sake of contradiction, that there exists an integer $2 \leq v \leq \ell_j - 1$ such that $\ell_{i,j}^v + v < \ell_{i,j}^1 + 1$. Note that the total travel distance $\ell_{i,j}^v + (v-1)$ represents a feasible path starting from node $(j,1)$ that first travels to (j,v) and then begins the shared ride with i. By the definition of the shortest shared-ride distance, we must have $\ell_{i,j}^1 \leq \ell_{i,j}^v + (v-1)$. Adding 1 to both sides yields $\ell_{i,j}^1 + 1 \leq \ell_{i,j}^v + v$, which directly contradicts our assumption. □ □

Lemma 7. *$\forall i \in [N], (j,u) \in \mathcal{C}$, if $i \in \mathcal{N}_{j,u}^+$, then $i \in \mathcal{N}_{j,1}^+$.*

Proof. Proof of Lemma 7 If $u = 1$, the lemma clearly holds. Suppose $i \in \mathcal{N}_{j,u}^+$ for some $u \geq 2$. Based on the trip length condition, we have $\ell_{i,j}^u < \ell_i + \ell_j - u$. Combining this with Lemma 6 (i.e., $\ell_{i,j}^1 + 1 \leq \ell_{i,j}^u + u$), we obtain:

$$\ell_{i,j}^1 \leq \ell_{i,j}^u + u - 1 < (\ell_i + \ell_j - u) + u - 1 = \ell_i + \ell_j - 1.$$

This implies that i and $(j,1)$ satisfy the trip length condition.

Now consider the backtracking condition. If (j,u) is not on any shortest path from O_i (exclusive) to D_j, then $(j,1)$ must also not be on any such shortest path. This follows because (j,u) lies on the shortest path from $(j,1)$ to D_j; if $(j,1)$ were on a shortest path from O_i to D_j, then by the optimality of sub-paths, (j,u) itself would also be on that shortest path, leading to a contradiction. Therefore, i and $(j,1)$ satisfy the backtracking condition.

Since i and $(j,1)$ satisfy both the trip length condition and the backtracking condition, $i \in \mathcal{N}_{j,1}^+$. □ □

Now we go back to the proof of Proposition 5. Note that after relaxing the ratio constraints (3.2d), $C(\boldsymbol{\lambda})$ can be obtained via the following dual problem:

$$\begin{aligned}
C(\boldsymbol{\lambda}) = \quad & \max_{\boldsymbol{\gamma},\boldsymbol{\xi}} \quad \sum_{i\in[N]} \Lambda_i \lambda_i \gamma_i \\
& \text{s.t.} \quad \gamma_i - \xi_i^1 \leq c, && \forall i \in [N], && \text{(B.10a)} \\
& \qquad \xi_i^u - \xi_i^{u+1} \leq c, && \forall i \in [N], 1 \leq u \leq \ell_i - 2, u \in \mathbb{Z}, && \text{(B.10b)}
\end{aligned}$$

$$\xi_i^{\ell_i-1} \le c, \qquad \forall i \in [N], \tag{B.10c}$$
$$\gamma_i + \xi_j^u \le c\ell_{i,j}^u, \qquad \forall (j,u) \in \mathcal{C}, i \in \mathcal{N}_{j,u}^+. \tag{B.10d}$$

Let $\gamma_i(\hat{\boldsymbol{\lambda}})$ be an optimal value of γ_i under $\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}}$. Fixing $\gamma_i(\hat{\boldsymbol{\lambda}})$, let

$$\xi_i^u(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) - cu, \quad \forall (i,u) \in \mathcal{C}.$$

The following lemma shows that $(\gamma_i(\hat{\boldsymbol{\lambda}}), \xi_i^u(\hat{\boldsymbol{\lambda}}))$ gives an optimal dual solution.

Lemma 8. *Assume $\gamma_i(\hat{\boldsymbol{\lambda}})$ is an optimal value of γ_i when $\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}}$. Then, $\xi_i^u(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) - cu$ gives an optimal value of ξ_i^u.*

Proof. Proof of Lemma 8. Note that if all constraints (B.10a)-(B.10b) are binding, we have $\xi_i^u(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) - cu$, $\forall (i,u) \in \mathcal{C}$. Therefore, assume by contradiction that there exists an optimal solution $\gamma_i(\hat{\boldsymbol{\lambda}})$ and $\xi_i^u(\hat{\boldsymbol{\lambda}})$, such that at least one constraint in (B.10a)-(B.10b) is not binding. Now consider the following two cases.

For constraint (B.10a), if there exists $i \in [N]$, such that $\gamma_i(\hat{\boldsymbol{\lambda}}) - \xi_i^1(\hat{\boldsymbol{\lambda}}) < c$, then for all $0 \le u \le \ell_i$, we can decrease the value of $\xi_i^u(\hat{\boldsymbol{\lambda}})$ by a sufficiently small value $\varepsilon > 0$. It is easy to verify that all other constraints (B.10b)-(B.10d) are still satisfied since the right-hand sides are unchanged and the left-hand sides are non-increasing with respect to the decrease in $\xi_i^u(\hat{\boldsymbol{\lambda}})$, and the objective value is unchanged, thus we can always let $\gamma_i(\hat{\boldsymbol{\lambda}}) - \xi_i^1(\hat{\boldsymbol{\lambda}}) = c$ hold for all $i \in [N]$.

Similarily, for constraint (B.10b), suppose there exists $i \in [N]$ and $1 \le u \le \ell_i - 2$ ($u \in \mathbb{Z}$), such that $\xi_i^u(\hat{\boldsymbol{\lambda}}) - \xi_i^{u+1}(\hat{\boldsymbol{\lambda}}) < c$. Then for all $v \ge u+1$, we decrease the value of $\xi_i^v(\hat{\boldsymbol{\lambda}})$ by a sufficiently small value $\varepsilon > 0$. All other constraints (B.10a), (B.10c) and (B.10d) are still satisfied. Therefore, we can always let $\xi_i^u(\hat{\boldsymbol{\lambda}}) - \xi_i^{u+1}(\hat{\boldsymbol{\lambda}}) = c$ hold for all $i \in [N]$ and $1 \le u \le \ell_i - 2$ ($u \in \mathbb{Z}$).

Consequently, we can always find an optimal value of ξ_i^u such that all constraints (B.10a)-(B.10b) are binding, which leads to $\xi_i^u(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) - cu$, $\forall (i,u) \in \mathcal{C}$. □ □

Now, consider the case where the platform implements the matching policy (3.5) under an optimal dual solution $(\gamma_i(\hat{\boldsymbol{\lambda}}), \xi_i^u(\hat{\boldsymbol{\lambda}}))$. Suppose a solo rider (j,u) is matched with a new rider $i \in \mathcal{N}_{j,u}^+$. By the definition of the matching policy and the lower bound provided by Lemma 6, the following must hold:

$$\gamma_i(\hat{\boldsymbol{\lambda}}) \ge c\ell_{i,j}^u - \xi_j^u(\hat{\boldsymbol{\lambda}}) = c(\ell_{i,j}^u + u) - \gamma_j(\hat{\boldsymbol{\lambda}}) \ge c(\ell_{i,j}^1 + 1) - \gamma_j(\hat{\boldsymbol{\lambda}}). \tag{B.11}$$

Rearranging this yields the lower bound: $\gamma_i(\hat{\boldsymbol{\lambda}}) + \gamma_j(\hat{\boldsymbol{\lambda}}) \ge c(\ell_{i,j}^1 + 1)$.

On the other hand, since $i \in \mathcal{N}_{j,u}^+$, Lemma 7 implies that $i \in \mathcal{N}_{j,1}^+$. Substituting the optimal dual form from Lemma 8 into the dual constraint (B.10d) for the pair i and $(j,1)$, we obtain:

$$\gamma_i(\hat{\boldsymbol{\lambda}}) + \xi_j^1(\hat{\boldsymbol{\lambda}}) = \gamma_i(\hat{\boldsymbol{\lambda}}) + (\gamma_j(\hat{\boldsymbol{\lambda}}) - c) \le c\ell_{i,j}^1.$$

Rearranging this yields the upper bound: $\gamma_i(\hat{\boldsymbol{\lambda}}) + \gamma_j(\hat{\boldsymbol{\lambda}}) \le c(\ell_{i,j}^1 + 1)$.

Combining the lower and upper bounds, we must have $\gamma_i(\hat{\boldsymbol{\lambda}}) + \gamma_j(\hat{\boldsymbol{\lambda}}) = c(\ell^1_{i,j} + 1)$. Consequently, all inequalities in (B.11) must hold as equalities, implying that $\ell^u_{i,j} + u = \ell^1_{i,j} + 1$. □

B.3 Other Results Based on Chicago Ridesharing Data

B.3.1 Training Data Results

This subsection presents additional computational results based on Chicago ridesharing data. Table B.1 is the corresponding training results for time window defined in Section 3.4.2, using a synthetic training dataset generated over a sufficiently long horizon with mean arrival rates estimated from the true training data.

Cost ($/mile)		$c = 0.7$					$c = 0.9$					$c = 1.1$				
Waiting Window (min)		0	0.5	1	1.5	2	0	0.5	1	1.5	2	0	0.5	1	1.5	2
Profit ($/min)	Pre-trip	2.631	4.001	4.824	5.384	5.796	0.292	0.763	1.322	1.767	2.123	0.000	0.000	0.000	0.000	0.037
	Combined	4.769	5.282	5.700	6.075	6.399	1.449	1.779	2.090	2.385	2.651	0.000	0.084	0.214	0.338	0.510
	Difference	81.2%	32.0%	18.1%	12.8%	10.4%	396.9%	133.3%	58.1%	35.0%	24.9%	-	∞	∞	∞	1287.6%
Quoted Price ($/mile)	Pre-trip	0.850	0.795	0.774	0.762	0.752	0.950	0.900	0.869	0.852	0.841	1.000	1.000	1.000	1.000	0.985
	Combined	0.798	0.778	0.764	0.755	0.747	0.890	0.870	0.853	0.840	0.831	1.000	0.982	0.959	0.944	0.928
	Difference	-6.1%	-2.1%	-1.2%	-0.9%	-0.7%	-6.3%	-3.4%	-1.9%	-1.4%	-1.1%	0.0%	-1.8%	-4.1%	-5.6%	-5.8%
Payment ($/mile)	Pre-trip	0.850	0.788	0.764	0.751	0.741	0.950	0.884	0.841	0.819	0.807	-	-	-	-	0.773
	Combined	0.790	0.770	0.755	0.744	0.736	0.870	0.847	0.826	0.811	0.801	-	0.880	0.877	0.849	0.830
	Difference	-7.0%	-2.3%	-1.2%	-0.8%	-0.7%	-8.5%	-4.2%	-1.7%	-1.0%	-0.8%	-	-	-	-	7.4%
Throughput (requests/min)	Pre-trip	4.238	5.797	6.397	6.739	7.006	1.410	2.830	3.696	4.175	4.491	0.000	0.000	0.000	0.000	0.412
	Combined	5.713	6.277	6.656	6.936	7.158	3.095	3.686	4.167	4.511	4.762	0.000	0.498	1.154	1.577	2.029
	Difference	34.8%	8.3%	4.1%	2.9%	2.2%	119.4%	30.2%	12.7%	8.0%	6.0%	-	∞	∞	∞	392.5%
Match Rate	Pre-trip	0.000	0.409	0.541	0.607	0.646	0.000	0.355	0.523	0.612	0.662	-	-	-	-	0.720
	Combined	0.601	0.675	0.726	0.760	0.786	0.612	0.689	0.743	0.780	0.805	-	0.776	0.808	0.867	0.899
On-trip Match Portion	Combined	1.000	0.674	0.454	0.328	0.249	1.000	0.729	0.498	0.355	0.268	-	0.798	0.587	0.401	0.276
Cost Efficiency	Pre-trip	0.000	0.109	0.165	0.198	0.222	0.000	0.086	0.154	0.194	0.220	-	-	-	-	0.360
	Combined	0.160	0.190	0.216	0.235	0.253	0.160	0.188	0.213	0.234	0.252	-	0.247	0.244	0.267	0.290
Average Detour Rate (matched requests)	Pre-trip	-	0.095	0.079	0.071	0.064	-	0.103	0.083	0.072	0.066	-	-	-	-	0.000
	Combined	0.099	0.094	0.089	0.083	0.078	0.098	0.094	0.089	0.084	0.079	-	0.054	0.076	0.077	0.075

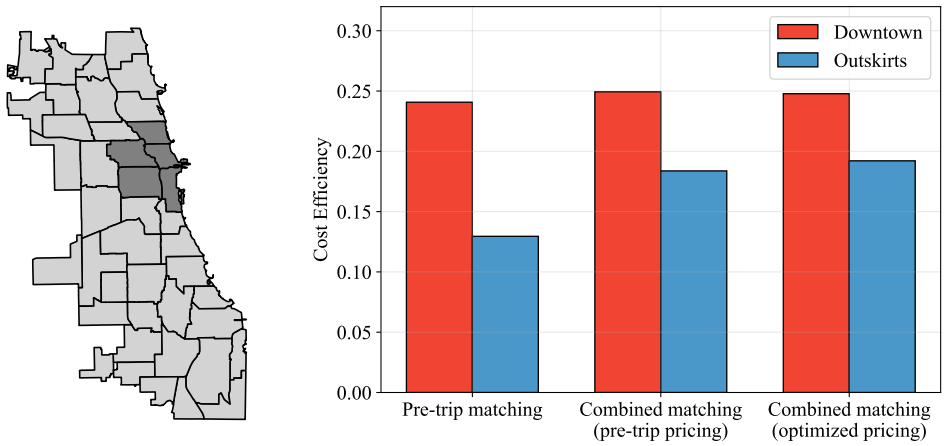
Table B.1: Training results on Chicago data.

B.3.2 Regional Pricing Effects

In this section, to understand the reason behind the small price increase around the densest downtown zones in Figure 3.9c, we compare three policies: (1) pre-trip matching and the associated pricing policy, (2) combined matching under the pre-trip pricing policy, and (3) combined matching and the associated pricing policy. We separate the small downtown region where prices increase versus the outskirts as shown in Figure B.1a. Figure B.1b presents the cost efficiency for each region achieved by each of the three policies. We also define five types of matching outcomes for any request: (1) matched with a downtown rider pre-trip, (2) matched with an outskirts rider pre-trip, (3) matched with a downtown rider on-trip, (4) matched with an outskirts rider on-trip, or (5) unmatched and dispatched solo. Figure B.2 presents the matching outcome distributions over downtown and outskirts requests for each of the three policies.

Under the pre-trip matching and pricing policies (first column in Figures B.1b, B.2b and C.1), downtown requests achieve high cost efficiency (0.24) and are mainly matched pre-trip with other downtown requests (48.5% of all downtown requests). Meanwhile, outskirts requests achieve a lower cost efficiency (0.13) despite a high pre-trip match rate (35.3% with other outskirts riders). In other words, pre-trip matching alone is highly efficient downtown, but inefficient in the outskirts.

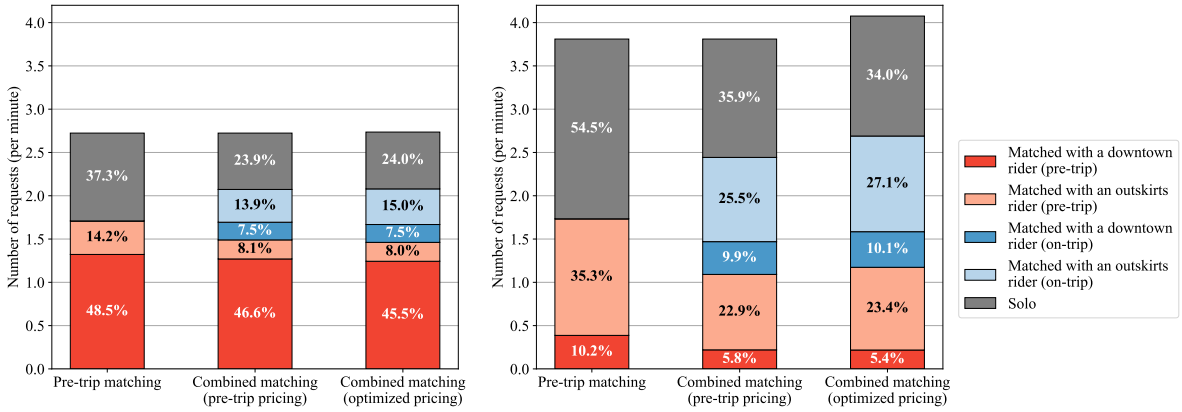
Fixing the pricing policy and adding on-trip matching (second column in Figures B.1b, B.2b and C.1), outskirts riders benefit significantly (cost efficiency increases from 0.13 to 0.18), whereas downtown riders only see modest improvement (cost efficiency increases only from 0.24 to 0.25). Figure B.2 illustrates that rather than simply adding on new matching opportunities, the addition of on-trip matching in fact *suppresses* pre-trip matching: a downtown (or outskirts) request is less likely to be matched pre-trip with another downtown (or outskirts) request, and more likely to be matched on-trip (Figure B.2). Then, re-optimizing the pricing policy (third column in Figures B.1b, B.2b and C.1) reinforces this shift. The (slight) increase in downtown prices further reduces pre-trip matching between downtown riders, and shifts the downtown riders towards more on-trip matches with outskirts riders. Meanwhile, in the outskirts, the price decrease is driven by the dominant effect of more efficient on-trip matches.



(a) Region Split

(b) Cost Efficiency Distribution

Figure B.1: Downtown/outskirts region split and the cost efficiency distribution in downtown versus outskirts across the three policies ($c = 0.7$, waiting period is 1 min).



(a) Downtown

(b) Outskirts

Figure B.2: Matching outcome distribution over downtown/outskirts requests ($c = 0.7$, waiting window = 1 min).

Appendix C

SUPPLEMENTARY MATERIALS FOR CHAPTER 4

C.1 Case

C.1.1 Introduction

Eliot George is the CEO of Calyber, a ridesharing company based in Chicago, Illinois. When he founded Calyber several years ago, Eliot was driven by a firm belief: to truly succeed, Calyber must offer shared rides—where each driver transports multiple compatible passengers in a carpool-like service. Having grown up in Los Angeles, Eliot was all too familiar with the frustrations of traffic congestion. This experience deeply shaped his vision for Calyber. He was fond of saying that if the company failed to achieve a high level of ride sharing, it would be no different from a traditional taxi service—just one with a slick mobile app.

But offering a profitable shared rides service had turned out to be more difficult than he had once imagined. Downtown, Calyber kept prices low to attract customers. But prices were so low that even with most riders matching with compatible co-riders, Calyber was unable to recoup its dispatching costs. Outside of downtown, prices were higher, but this caused separate issues. Most riders who booked a shared ride were never matched with compatible co-riders; meanwhile, those who did share were often sent on long, costly routes picking up co-riders all over the city. This had turned Calyber into the butt of jokes on social media, as vocal customers complained about irritating co-riders and long detours. (Customers with positive experiences rarely posted on social media, Eliot noted with some chagrin.)

Faced with these challenges, Eliot decided to discuss a way forward with Calyber's Chief Scientist, Thea Bruckner. He hoped that together, they could design a shared rides product that could achieve high ridership *and* profitability at the same time.

C.1.2 Background

A ridesharing platform is a transportation service provider that connects customers (*riders*) with drivers who can take them from their origins to their desired destinations, usually via a mobile app.

Lyft was founded in 2007 by Logan Green and John Zimmer as an intercity carpool service between college campuses, before shifting in 2013 to short-distance rides [8]. Uber was founded in 2009 by Travis Kalanick and Garrett Camp after they had difficulty hailing a taxi in Paris and sought to make the experience as simple as pushing a button on a smartphone [111].

Initially, both companies focused on solo rides, similar to a taxi service. This changed in 2014 when both companies (within a day of each other!) launched *shared* rides in San

Francisco [7]. The products, dubbed UberPool and Lyft Line, offered riders lower prices in exchange for potentially sharing the ride with other riders in a carpool-like service. The idea was that multiple riders with compatible origins and destinations could be served concurrently (*matched*) with the same driver, reducing vehicle miles and improving efficiency. As Kalanick put it, "We saw a lot of people pushing the same button at the same time going essentially to the same place. And so we started thinking about, well, how do we make those two trips and turn them into one?" [106] The platform would benefit by serving more riders at lower cost; riders would benefit if lower costs translated into prices low enough to make them willing to share; and Zimmer even claimed that vehicle miles would be reduced so much as to enable the "end of traffic" [61].

Within a few years, the new shared rides products became core offerings, comprising 20-30% of all rides [70, 72]. In 2018, Uber doubled down on shared rides and launched Uber Express Pool, a low-cost variant of UberPool that charged riders low fares in exchange for walking toward meeting points with other riders [108]. At the same time, Lyft announced its ambition for 50% of its rides to be shared by the end of 2020 [72].

However, operational challenges have prevented shared rides from living up to their lofty aspirations. After the launch in 2014, both companies sought to "win" riders in San Francisco, chiefly by keeping prices low—Uber even offered a $5 flat fare.[1] But at the same time, Uber's algorithms struggled to match riders: the *match rate*, or the fraction of riders who get matched with another rider, was reportedly only 8%. The combined effect was that Uber was essentially offering ultra-cheap solo rides, a practice that lost $1 million a week and that was subsequently described as "bleeding cash subsidizing rides [without] a plan for tomorrow". Over the next few months, Uber improved its matching algorithms and achieved a match rate of almost 25%.[2] But this improvement was still not enough to break even. Attempting to cut its losses, Uber raised prices, which then tanked ridership by 22% [7].

Compounding these difficulties, the COVID-19 pandemic compelled both companies to shut down shared rides entirely to aid physical distancing. Recognizing the difficulty of bringing customers back to shared rides, Lyft's and Uber's relaunches in 2021 and 2022 came with a strong emphasis on affordability for inflation-conscious customers. Lyft included an option to reserve shared rides early for an extra discount, promising "the most affordable pricing available" [73]. Meanwhile, Uber launched a pricing policy that would offer further discounts to riders if they were matched with co-riders. As Uber CEO Dara Khosrowshahi explained, "you get a deeper discount if you get matched up with somebody, so it's in your interest to get matched up... taking that sharing to the next level and the discounts to the next level" [110, 80]. However, these launches struggled to find traction [33], and in the face of financial difficulties, Lyft stopped offering shared rides in 2023 [29]. Announcing the decision, Lyft CEO David Risher explained that "The problem with shared trips is that they take people out of their way. At some point you have to pay attention to what your customers want."

---

[1] See Figure 1.1 for a timeline of the pricing policies that Uber and Lyft experimented with over the first year launching shared rides.

[2] See Figure 1.2 for the match rates of Uber and Lyft's shared rides in Chicago over recent years.

The up-and-down history of shared rides highlights the following operational challenges. First, matching requires multiple riders with compatible origins, destinations, and request times, which can be rare unless demand density is sufficiently high. Second, low prices are needed to attract riders and achieve enough density for efficient matching, but can create losses for the platform if riders cannot be matched effectively. In trying to strike a balance, platforms may set prices that are both too high to attract riders *and* too low to comfortably sustain. These operational challenges fuel skepticism that shared rides can be operated viably.

At the same time, these challenges present an opportunity for any ridesharing company that can solve the puzzle of a viable shared rides product. In 2025, Uber promoted new features called Route Share and Price Lock, aimed at making regular trips (e.g., to and from work or major events) more affordable to combat continued inflation and improve accessibility [91, 103, 104]. Lyft also started experimenting with shared rides again, reintroducing them at airports with high demand density [71]. Moreover, beyond the high-density corridors formed by commutes, major events, and airport trips, low-density neighborhoods are an underserved market primed for innovation.

C.1.3 Calyber's Operations

Calyber is based in Chicago, Illinois and focuses entirely on offering shared rides. Its platform is used by about 35,000 customers per day. The service is *on-demand*, which means that riders can expect to be served quickly. Calyber might let riders wait on the platform to give time for compatible co-riders to arrive, but this waiting time is kept short (e.g., at most a few minutes) to ensure customer satisfaction. Calyber also guarantees that each rider will be matched with at most one other co-rider.

To request a ride, a rider can open Calyber's mobile app, enter her origin and destination, and receive a price quote (this is called a rider *session*). If the rider is willing to pay the quoted price (she decides chiefly based on the price per unit distance, in $/mile), then she *requests* a ride. Otherwise, she leaves the platform. The rider's willingness to pay is unknown to Calyber.

After a rider makes a request, Calyber makes a dispatch decision. If there is another compatible rider currently waiting on the platform, Calyber can match the two riders and dispatch them immediately in a shared ride. Otherwise, Calyber can let the rider wait on the platform for a short period. The rider has limited patience, so she is willing to wait only for a limited sojourn time, which is also unknown to Calyber. If her patience expires, then as she is about to exit the platform, Calyber can intervene and dispatch her immediately in a solo ride. Whether the dispatch is shared or solo, Calyber pays the driver $0.70/mile to perform the dispatch. As Calyber is known to treat drivers well, there are sufficient drivers to ensure that a driver is available for immediate dispatch.

A critical feature of shared rides is that the average cost per request depends on demand density. When demand density is high (i.e., more riders request similar trips at similar times), it is easier to find matches with low detours. Then, a single driver can serve multiple requests with only slightly longer routes, reducing the average cost per request. By contrast, when demand is sparse, it is more difficult to find matches. Consequently, the

platform dispatches more requests along routes with longer detours or even solo, increasing the average cost per request.

C.1.4 A New Engine for Pricing and Matching

Calyber's pricing and matching engine had been designed by one of its founding engineers, Nick Bailey. Unfortunately, Nick was no longer at Calyber, and the engine was a black box to the rest of the engineering team (hastily-written code from the early years of a startup is not known for tidiness).

Eliot and Thea agreed that Calyber could not address its operating issues without overhauling its pricing and matching engine, and decided to discuss the best way to move forward.

Eliot: The shared rides product isn't working for us—we're bleeding cash.[3] I think we need to raise prices. And it seems that the engine is pretty complicated and has caused your team a lot of trouble. What if we simplify and charge everyone a flat price, like $0.80/mile?

Thea: I agree that we should raise prices, although we should be careful not to drive away riders. As for the flat price, I see the appeal of simplicity, but I don't think charging different riders different prices is the problem. Actually, charging different riders different prices can be a *good* thing. Downtown riders are matched more than the riders in the outer neighborhoods, so they cost us less per mile. Shouldn't they get lower prices?[4]

Eliot: What do you think is the problem with the engine, then?

Thea: To me, the problem with the engine is that its pricing decisions are completely black box. It's impossible to troubleshoot. Remember a couple weeks ago when a bug made it into production and we tanked the price of airport rides for an entire weekend? It took my team days to track down the issue.

Eliot: Yeah, we need to make sure that this kind of mistake never happens again. That's one reason why I proposed a flat price for everyone. But you made a good point just now—different riders cost us different amounts per mile. So, what about pricing at cost? It seems like we could at least differentiate downtown and outer riders.

---

[3] See Table C.1 ("Nick's engine") for Calyber's Key Performance Indicators (KPIs) under the current engine.

[4] See Figure C.3 for demand, cost, and price (as priced by Nick's engine) across different neighborhoods of Chicago.

Thea: We can model costs at an even more granular level than downtown versus outer. We record every route that our drivers take, including all rider pickups and dropoffs. We can easily use the routes to calculate the average cost by origin and destination.[5]

Eliot: Great. What do you think riders will think of our pricing strategy?

Thea: That's another piece we should figure out. Every time a rider uses the mobile app, we record the origin, destination, price, and whether the rider makes a request.[6] We could use this historical data to learn how riders respond to our prices and build a demand model. Combining the demand model with the cost model, we can set prices that maximize profit.

Eliot: This sounds good to me. So, pricing is resolved then. What will we do about matching, then?

Thea: There's so much to do on the pricing side that I haven't thought about matching yet. I'm hoping that we'll have enough customers that we won't have to worry so much about matching. Then we could do something simple. Say a rider has just made a request on the platform. If there are no compatible co-riders waiting for dispatch, then let the arriving rider wait. But if there are compatible co-riders waiting on the platform, then match the arriving rider with the waiting rider that generates the highest level of cost saving in their shared route.

Eliot: That sounds reasonable to me. Why would we ever match any other way?

Thea: Well, it's possible that if we let the arriving rider wait even longer before we dispatch, then other compatible riders might arrive and we might be able to find better matches. But by waiting, we also risk riders trying to cancel their rides if we make them wait too long.

Eliot: Waiting can be valuable, then. You talked about using historical data to understand how riders respond to prices. Can we also use it to understand how they feel about waiting?

Thea: Maybe... We do record riders' waiting time from arrival to dispatch in our database.[7] But we're missing some information—for riders who are matched with other riders, we don't know how long they would have waited otherwise. I think this would be difficult for us to do right—we already have a lot to do on the pricing side, and my team is stretched pretty thin. Why don't we focus on getting a simpler prototype up and running, and work on this waiting idea later?

---

[5]See Table C.2 for a snapshot of Calyber's routing database.

[6]See Table C.3 for a snapshot of Calyber's rider database.

[7]See Table C.3 for a snapshot of Calyber's rider database, "Waiting Time" column.

Eliot: I see. I do like the idea of giving customers fast service, so let's go with your first idea.

Thea: Sounds great to me. We'll get started right away on implementing the new engine, and we'll kick off an experiment to track profit, match rate, and throughput.

Eliot: Sounds like a plan.

Thea's team implemented the new engine and piloted it in an experiment.[8] Thanks to an increase in prices, profit increased—instead of losing $3.76 per minute, Calyber was now making $2.47 per minute in profit. Thea was happy that the new engine was profitable, but suspected that there was still room for improvement. Throughput had dropped by 76%, from 16.82 to 4.02 riders per minute. As a result, the match rate dropped from 80% to 59%.

Thea wondered if they had raised prices too quickly. Should they back off the higher prices? If so, were the demand and cost models accurate enough to quantify the impact of such a pricing change? And could they do anything else to increase demand beyond simply adjusting prices up and down? Finally, if they did have to keep prices at the higher levels, could they improve match rates with the existing throughput? She called a team meeting to discuss these questions and chart a path forward.

Exhibits

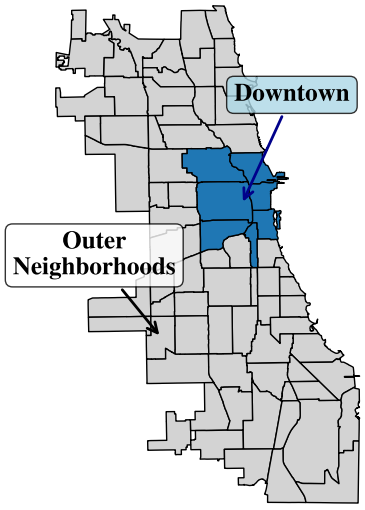


Figure C.1: Downtown and outer neighborhoods of Chicago.
Note: The downtown neighborhoods cover 20.36 square miles and have demand of 9.74 riders/min. The outer neighborhoods cover 197.08 square miles and have demand of 18.73 riders/min.

[8]See Table C.1 ("Thea's engine") for Calyber's Key Performance Indicators (KPIs) under the new engine.

	Nick's Engine			Thea's Engine		
Key Performance Indicator	Downtown	Outer	Overall	Downtown	Outer	Overall
Profit ($/min)	-1.96	-1.79	-3.76	0.76	1.72	2.47
Throughput (#/min)	6.92	9.90	16.82	1.71	2.31	4.02
Match rate	0.86	0.76	0.80	0.67	0.53	0.59
Detour rate	0.14	0.14	0.14	0.14	0.14	0.14
Average quoted price ($/mile)	0.48	0.60	0.56	0.80	0.84	0.83
Average cost per rider ($/mile)	0.59	0.63	0.62	0.61	0.64	0.63

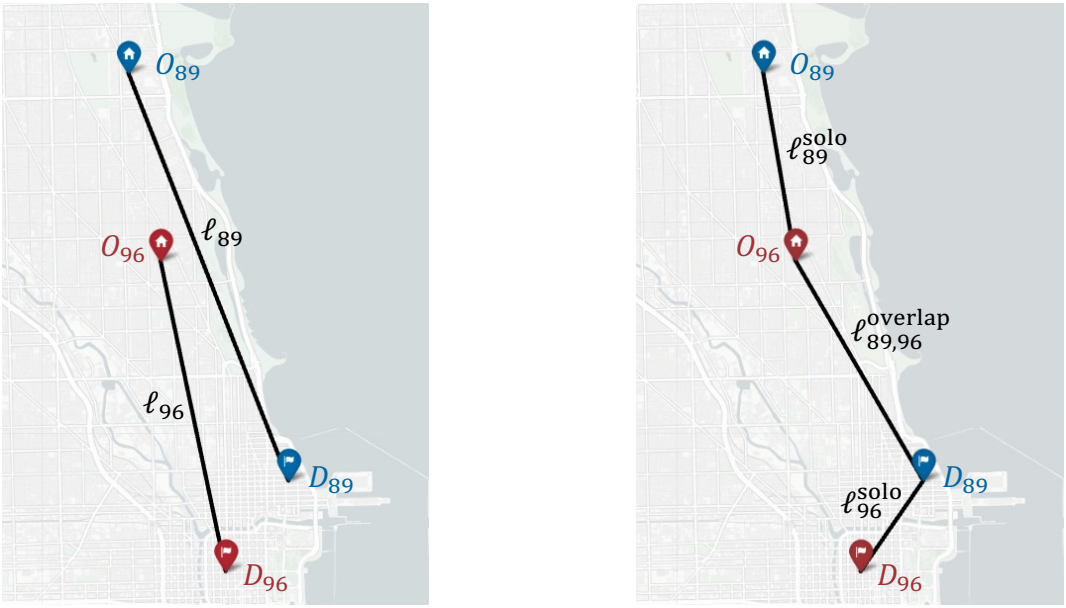
Table C.1: KPIs for Nick's and Thea's pricing and matching engines, by neighborhood type.
Note: "Throughput" is the number of requests served per minute. "Match rate" is the fraction of requests that are matched with another request. For the calculation of "Detour rate" and "Average cost per rider", see the caption of Table C.2.

Route ID	**Rider ID**	**OD Length**	**Component Length**			**Detour Rate**	**Cost**
		(miles)	Overlap	Solo	Total		($/mile)
1	76	5.28	0.00	5.28	5.28	0.00	0.70
2	89	4.69	2.72	2.04	5.94	0.01	0.51
2	96	3.39	2.72	1.18	5.94	0.15	0.52

Table C.2: Sample of Calyber's data on routes taken to fulfill rider requests.
Note: Details of Route 2 are given in Figure C.2.
"OD Length" is the direct distance from the rider's origin to destination.
"Overlap Length" is the distance of the route component when both riders are in the vehicle.
"Solo Length" is the distance of the route component when only the given rider is in the vehicle.
"Total Length" is the length of the entire route, including shared and solo components for all riders (e.g., $5.94 = 2.72 + 2.04 + 1.18$).
"Detour Rate" is the fraction of the OD length that is added for sharing (e.g., $0.01 = (2.72 + 2.04 - 4.69)/4.69$).
"Cost" is the cost per mile of fulfilling the given rider, allocating the cost along the shared component equally between the two riders (e.g., $0.51 = 0.7(2.72/2 + 2.04)/4.69$).

Rider ID	**Route ID**	**Arrival Date**	**Arrival Time**	**Pickup Lat**	**Pickup Lon**	**Dropoff Lat**	**Dropoff Lon**	**Pickup Area**	**Dropoff Area**	**Solo Length** (miles)	**Quoted Price** ($/mile)	**Converted**	**Waiting Time** (seconds)	**Matched With**
61	NaN	09/09	7:31:55	41.892	-87.665	41.881	-87.633	24	32	1.83	0.43	0	NaN	NaN
76	1	09/09	7:32:20	41.961	-87.752	42.009	-87.673	15	1	5.28	0.74	1	473.59	NaN
89	2	09/09	7:32:40	41.958	-87.653	41.895	-87.620	3	8	4.69	0.61	1	15.00	96
96	2	09/09	7:32:55	41.929	-87.646	41.881	-87.633	7	32	3.39	0.57	1	0.00	89

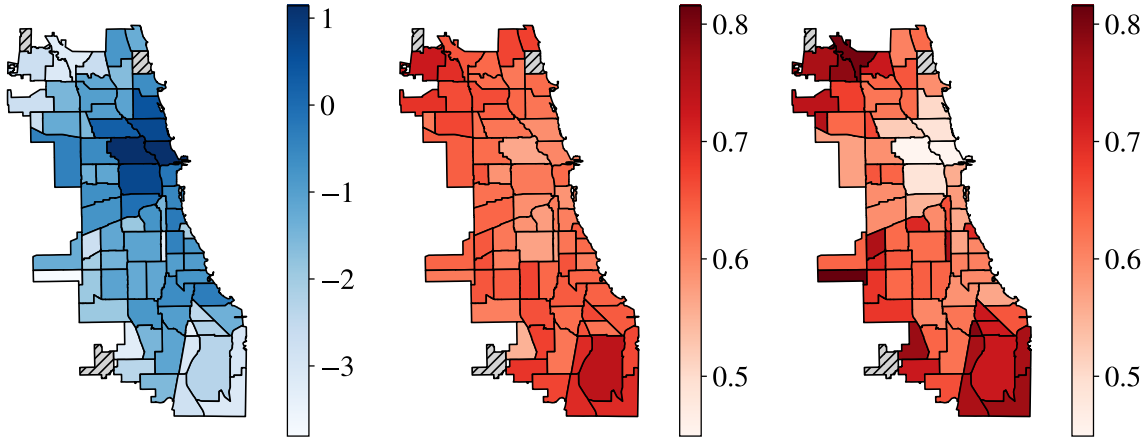
Table C.3: Sample of Calyber's data on rider sessions.
Note: Riders who were quoted a price but did not request a ride have "Converted" = 0 and "Waiting Time" = NaN. Riders who requested a ride but were not matched with another rider have "Converted" = 1 and "Matched With" = NaN.



(a) Unmatched

(b) Matched

Figure C.2: Illustration of the components of routes serving riders 89 and 96. Calyber currently cannot afford either a commercial mapping service or the development of an in-house alternative, so it relies on Haversine distance for routing.
Note: The labels O_{89}, O_{96}, D_{89}, and D_{96} denote the origins and destinations of riders 89 and 96, respectively. Figure C.2a shows the routes followed if both riders are dispatched unmatched. In this case, their dispatching costs are based on the OD lengths ℓ_{89} and ℓ_{96}.
Figure C.2b shows the route followed when the riders are matched. In this case, the route is divided into solo components of length ℓ_{89}^{solo} and ℓ_{96}^{solo}, and an overlap component of length $\ell_{89,96}^{\text{overlap}}$. The dispatching cost attributed to riders 89 and 96 are $\ell_{89}^{\text{solo}} + \frac{1}{2}\ell_{89,96}^{\text{overlap}}$ and $\frac{1}{2}\ell_{89,96}^{\text{overlap}} + \ell_{96}^{\text{solo}}$, respectively.



(a) Demand (log scale) (b) Cost per rider (c) Quoted price

Figure C.3: Demand (riders per minute), cost per rider ($/mile, under Nick's engine), and quoted price ($/mile, under Nick's engine) across different neighborhoods of Chicago, for a typical weekday 7:30-8:30am.
Note: Cost per rider can be above $0.70/mile because of detours. Neighborhoods with limited data due to low demand are shaded gray and excluded.