Université catholique de Louvain

Louvain School of Management

Center for Operations Research and Econometrics

Stochastic Models for On-Demand Service Platforms Under Uncertainty

Thomas De Munck

Thesis submitted in partial fulfillment of the requirements for the degree of *Docteur en sciences de l'économie et gestion.*

Supervisors:

Philippe Chevalier (UCLouvain, Belgium)

Jean-Sébastien Tancrez (UCLouvain, Belgium)

Chair:

Stefan Creemers (UCLouvain, Belgium)

Jury:

Aadhaar Chaturvedi (Clemson University, USA)

Stefan Creemers (UCLouvain, Belgium)

Pieter Vansteenwegen (KULeuven, Belgium)

Jorge Vera (Pontificia Universidad Católica de Chile, Chile)

Abstract

On-demand service platforms such as Uber, Lyft, or Deliveroo, are implemented in many cities worldwide, and in various industries, including food delivery, grocery delivery, and ride-hailing. By providing timely and economical services, these platforms improve consumer satisfaction and are expected to play important roles in urban logistics. However, they also represent critical challenges as they evolve in dynamic environments characterized by large-scale operations and strategic user behaviors. In this Ph.D. thesis, we investigate various challenges emerging in the operations of on-demand service platforms and formulate stochastic models to support related decisions. Various numerical experiments are performed to collect managerial insights into the relevance of several decision levers, the benefits of recent innovations, the effects of multiple parameters, and the trade-off between different objectives.

The thesis is structured in five chapters. Chapter 1 motivates the research and provides background materials to describe on-demand service platforms, and understand methodologies supporting the rest of the thesis. The chapter concludes by summarizing the thesis contributions.

Chapter 2 studies the problem of an on-demand service platform that differentiates the service of two customer classes that arrive with distinct willingness to wait and to pay. For each customer arrival, the platform decides whether to admit the request and, if so, whether to allocate an available service provider. The problem is formulated as a Markov decision process, and an optimal policy is found using the value iteration algorithm. The properties of the value function are exploited to show that the optimal policy is described by two admission thresholds and a strict allocation rule. In a numerical study, the performance of the optimal policy is compared with simpler policies. Results show that the platform, customers, and service providers can benefit from a policy controlling customer admission and allocation. Finally, some evidence of the robustness of the policy is provided in two extensions.

Chapter 3 considers a ride-hailing platform that makes pricing, driver repositioning, and customer admission decisions to connect supply with demand over multiple locations. Customers are impatient and have distinct willingness to pay. Drivers can be repositioned by the platform or can relocate on their own. The problem is formulated as a Markov decision process, and a reinforcement learning approach is developed to find an efficient policy. The approach

first pretrains two neural networks to reproduce the policy prescribed by a deterministic rolling horizon strategy. Then, the policy is further improved with the Proximal Policy Optimization (PPO) algorithm. Using data from New York City, results show that the approach improves upon the PPO algorithm without pretraining and rolling horizon strategies, while reducing computation time and stabilizing learning. The interplay between pricing, driver repositioning, and customer admission is also explored, providing insights into their respective roles.

In Chapter 4, the benefits of ride-hailing dispatching with public transit transfers are evaluated. This is done by considering an online dispatching problem where customers arrive dynamically over time, and can receive door-to-door, first-mile, or last-mile service by the platform. To address the problem, a lookahead approach re-optimizes a two-stage stochastic program at each decision epoch. The approach is enhanced by a procedure to remove irrelevant decision variables and an adaptation of the L-shaped method. In numerical experiments, an extensive set of synthetic and real-world instances is designed to assess the benefits of the integrated system. Results suggest that the integration of public transit can increase the percentage of fulfilled requests, while improving profit and customer travel time. The trade-off between maximizing profit and minimizing customer travel time is also analyzed. A slight reduction of one objective allows a significant increase in the other objective.

In Chapter 5, some general concluding remarks are drawn by first reviewing the limits of the methodologies used, and then outlining several opportunities for future research.

Acknowledgments

While doing a PhD may seem solitary from the outside, it is actually a rich human experience. Above all, I sincerely thank my supervisors, Philippe Chevalier and Jean-Sébastien Tancrez, for giving me the opportunity to start this doctoral journey. I will always remember their constant support, encouragement, and sound advice. With his natural curiosity, Philippe played a key role in the early stages, leading reading groups and courses that laid the foundation of this thesis. Jean-Sébastien later guided me with an impressive eye for detail, offering numerous ideas that shaped the whole thesis. It has been a pleasure and honor to work with them, both as researchers and as friends.

I would also like to extend my gratitude to Professors Aadhaar Chaturvedi, Stefan Creemers, Pieter Vansteenwegen, and Jorge Vera for being part of my thesis jury. Thanks to all of them for the valuable comments, questions, and discussions raised during the defense. I am especially grateful to Professor Jorge Vera for hosting me during six months at Pontificia Universidad Catolica de Chile. My gratitude also goes to Professor Paul Belleflamme, who has been a member of my thesis committee, and has provided me with helpful comments and advice throughout my doctoral studies.

I have been fortunate to complete my doctoral studies surrounded by many great colleagues. As such, I would like to thank the research assistant team at LSM, in particular Henri, Brieuc, Hakimeh, Emma, François, and Flore. I also want to thank Quentin, Martial, Antoine, Nicolas, Jacques, Cefane, Thomas, Gianmarco, Jehum, Daniel, Francesco, Giorgio, and many other PhD fellows at CORE for the fun and stimulating discussions. My warmest appreciation also goes to Abey, Isha, David, John, Diego, and my football mates of Oldham, who made my exchange in Chile such an unforgettable experience.

I now wish to turn to my relatives. Thanks to my amazing parents for contributing so much to my education. Since the beginning, they have been a true source of inspiration. Thanks also to my dear sisters, Anaïs and Elsa. I feel privileged to have grown up with them in such a supportive environment. I also thank my lifelong friends for the laughter and companionship over the years. These moments in their company were essential after busy days of work.

Lastly, I want to thank my partner in life, Antonia, for moving with me to Belgium five years ago. I truly cherish her commitment to our journey together, and now look forward to the next adventures that lie ahead of us.

To Antonia

"Doubt is an uncomfortable condition, but certainty is a ridiculous one."

— Voltaire (1764)

Contents

1 Introduction

1.1 Context and Motivations

With the rise of the sharing economy, the transportation sector has witnessed the rapid growth of on-demand service platforms. These platforms can be used for restaurant food delivery (e.g., TakeAway, Deliveroo), grocery delivery (e.g., DoorDash, Gorillas), or ride-hailing transportation (e.g., Lyft, Uber). They usually operate in an urban context and are becoming important components of the transportation industry. For instance, Uber provided services to more than 164 million monthly customers worldwide as of 2024 (Uber, 2024b). Between 2017 and 2021, the restaurant food delivery market has more than tripled, becoming a global market worth more than $150 billion (Ahuja et al., 2021).

If we follow the terminology of transaction cost theory (Williamson, 1981), three factors can be highlighted to explain the success of on-demand service platforms. First, they reduce search costs by centralizing supply and demand that vary in time and space (Rogers, 2015). Second, they decrease negotiation costs by imposing prices between supply and demand (L. Sun et al., 2019). Third, they lower enforcement costs by developing rating and review systems (Thierer et al., 2015). As an illustration of these factors, let us consider the ride-hailing industry. Traditional taxi services usually cruise in the city to find riders, incurring high search costs. In contrast, ride-hailing platforms match drivers and riders over a single, central interface. While taxi services can generate friction due to possible price negotiations, ride-hailing platforms set service prices beforehand. Finally, should the rider or the driver be dissatisfied with the experienced service, ride-hailing platforms allow them to provide a rating, informing future users.

Although these platforms can be valuable, their operations can pose significant challenges for at least three reasons. First, on-demand service platforms operate in uncertain environments, with complex stochastic processes coexisting (H. Wang & Yang, 2019). For example, customers arrive dynamically, and service times are uncertain. These processes are also time-varying and can be influenced by external factors, such as weather conditions. To solve realistic

problems, dynamic methods are thus required.

Second, on-demand service platforms operate in large-scale settings as they aggregate supply and demand. Consequently, hundreds, if not thousands, of service providers and customers must be coordinated within short periods, e.g., less than a minute (Lowalekar et al., 2018). For this reason, exact solution methods are of little use. Instead, on-demand service platforms must rather adopt heuristic methods to find efficient solutions.

Third, these platforms involve users, who tend to act strategically (Qin et al., 2022). For example, customers may not be willing to pay the price set by the platform; service providers are not disposable resources, but may choose to relocate by themselves. These unpredictable behaviors introduce additional sources of uncertainty, further complicating the platform's operations. Since these behaviors are endogenous to the platform's decisions, on-demand service platforms must carefully evaluate the implications of their decisions on the various stakeholders.

Motivated by these challenges, the goal of this thesis is to develop mathematical models and solution methods to support the operations of on-demand service platforms, while understanding the implications for various stakeholders. To this aim, we consider three specific problems related to on-demand service platforms, and, more specifically, ride-hailing platforms. Before diving into these problems, we start by providing some context in the rest of this chapter. Section 1.2 gives a brief overview of operations on on-demand service platforms. Section 1.3 and Section 1.4 introduce the two main methodologies used in this thesis: dynamic optimization, and stochastic optimization. Lastly, Section 1.5 summarizes the thesis and its contributions.

1.2 On-Demand Service Platforms

This section introduces several aspects of on-demand service platforms, relevant to subsequent chapters. Section 1.2.1 first defines the timing of operations on on-demand service platforms. Section 1.2.2 then details the associated decisions. Section 1.2.3 finally discusses recent innovations.

1.2.1 Operations

On-demand service platforms typically connect customers with dedicated service providers using mobile apps. To build intuition, Fig. 1.1 illustrates the sequential process when a potential customer enters the application in search of a service. First, the potential customer enters the system and observes the price set by the platform. If the customer is willing to pay the price, a request is submitted. Otherwise, the customer balks and opts for an outside option (e.g., subway). Second, upon receiving the request, the platform decides whether to admit the customer into the system. If the customer is accepted, then the customer waits for dispatching, possibly abandoning in case of excessive waiting.

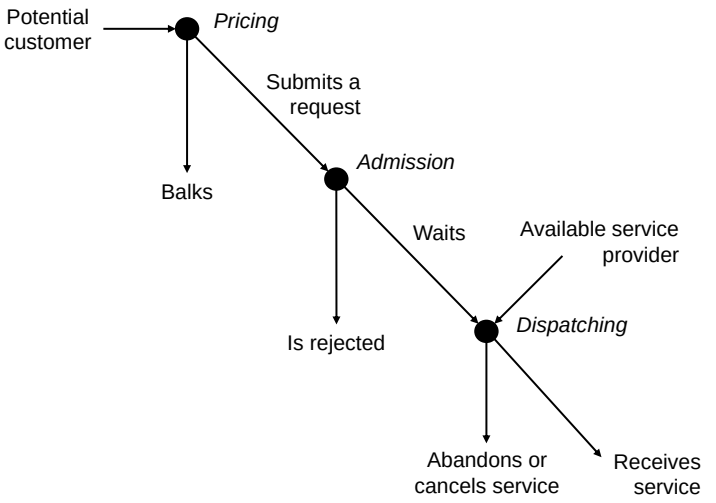


Figure 1.1: The sequential process of a customer passing a request on on-demand service platforms.

Otherwise, the customer is rejected, and opts for an outside option. Third, an available service provider is dispatched to complete the customer service. If the waiting is too long, then the customer may abandon before service.

The process described above suggests three key decision levels on on-demand service platforms: pricing, admission, and dispatching. Another important decision level concerns the repositioning of idle service providers, which can be realized in parallel. Together, these levels contribute to balancing supply and demand on on-demand service platforms. Other decisional aspects may be mentioned, such as the timing of the service (between the admission and dispatching decisions), and the routing of service providers while serving requests. However, we will only discuss decisions studied in the thesis for the sake of conciseness.

1.2.2 Decisions

We next detail the four decisions introduced in the previous section: pricing, admission, dispatching, and repositioning. For each decision, we explain how it is modeled in the thesis.

Pricing

On-demand service platforms can be viewed as two-sided markets (cf. Rochet & Tirole, 2006) where both demand and supply are affected by the price set by the platform. Therefore, the pricing strategy is an important tool to manage supply

and demand imbalances. In general, pricing on on-demand service platforms involves three main components.

The first component is *temporal*, and implies that some part of the price is set in advance, as a function of the time of the day. This component is often observed in practice due to its simplicity (Taylor, 2018), and because customers and providers can consider it fairer than dynamic pricing. The second component is *spatial*, which sets the price beforehand, depending on the origin and/or destination of the request. This scheme is useful to cope with structural imbalances, e.g., customers traveling from residential areas to the business district in the morning (Bimpikis et al., 2019), or to cover the operational costs of service providers (e.g., fuel costs for drivers, Gómez-Lobo et al., 2022). The third component is *state-dependent*. In that case, prices are adjusted dynamically as a function of the observed demand and supply. In practice, it can be implemented using simple threshold policies (e.g., Banerjee et al., 2015) or machine learning methods (e.g., Lei & Ukkusuri, 2023).

In this thesis, prices are set using these components in isolation, or jointly. In Chapter 3, prices are endogenous to the model, and adjusted as a function of the time of the day, the request origin, and the current system state. In Chapter 2, prices are exogenous, and determined in advance for specific periods. In Chapter 4 prices are also exogenous, and only depend on the distance traveled and the time of the day.

Customer Admission

As a complement to pricing, the admission decision can be used to further ration demand when supply is limited. It can sometimes even replace the pricing decision, e.g., in markets where dynamic pricing is negatively perceived (Kanoria & Qian, 2024).

To make this decision, on-demand service platforms can rely either on *direct* or *indirect* mechanisms. With a direct mechanism, the decision may take the form of an announcement stating that no driver can serve the customer request (e.g., Afèche et al., 2023). With an indirect mechanism, the platform does not explicitly refuse to serve the request but, instead, may inform the customer that long waiting times are to be expected before service (e.g. G. Wang et al., 2024).

The three chapters of this thesis account for the admission decision. In Chapter 3, the platform uses the decision to reject requests traveling to low-demand locations, hence directing drivers to more convenient ones. In Chapter 2, the platform employs the decision to balance the service of express customers (who pay a premium for faster service) and regular customers (who pay a standard price for a potentially delayed service). In Chapter 4, the decision is implicitly modeled, as a consequence of the dispatching process.

Dispatching

Besides pricing, dispatching providers to customers is also an important problem, as on-demand service platforms centralize demand and supply in the same mobile app. To dispatch providers, two categories of methods can be considered, which differ by the timing of the decision.

The first class of methods makes dispatching decisions *as soon as new information becomes available* (e.g., when a new request enters the system). These methods can take the form of simple heuristics (e.g., dispatching an order to the nearest provider, Castillo et al., 2017) or algorithmic strategies used in online matching (Dickerson et al., 2021). The second class of methods, more common in practice, makes dispatching decisions *over repeated time intervals* (e.g., every 5 minutes). With these methods, the problem is often represented as a bipartite graph (e.g. Azagirre et al., 2024), and solved using the Hungarian algorithm (Kuhn, 1955). By aggregating demand and supply, these methods enhance the dispatching process. At their core, they involve a trade-off between customer abandonment and service cancellation: customers waiting for too long may abandon before service, whereas customers matched too early may cancel service due to long pickup times (G. Wang et al., 2024).

Both classes of methods are considered in this thesis. In Chapter 3, provider dispatching occurs as soon as a request enters the system, based on a first-come, first-served rule. In Chapter 2, driver dispatching is determined as soon as a customer enters or abandons the system, or when a service is completed. In Chapter 4, driver dispatching is optimized at fixed time intervals.

Repositioning

The repositioning decision consists of guiding idle service providers to locations with high demand and/or high-value customers. In contrast to other sharing systems (e.g., bike-sharing systems), the repositioning decision can either be centralized, that is, made by the platform, or decentralized, that is, made by the resources themselves (service providers here).

If the decision is *centralized*, then service providers fully comply with the decision. Usually, providers earn a monetary compensation for repositioning. This scheme is relevant in the case of autonomous vehicles (Zhang & Pavone, 2016), and full-time employees (Via, 2025). It is also expected to become more prevalent in the future (Bertsimas et al., 2019). If the decision is *decentralized*, then providers choose to which location to relocate. The providers can make their decisions based on past experience. The platform can also influence the decision by using recommendation systems (Yu et al., 2019), sharing real-time information to drivers (Lu et al., 2018), or providing monetary incentives to relocate (Zhu, Ke, & Wang, 2021).

Only Chapter 3 models the repositioning decision. In this chapter, drivers can be repositioned by the platform, or relocate by themselves. When repositioned by the platform, drivers are assumed to follow the decision, in exchange for a compensation proportional to the distance traveled. When self-relocating

by themselves, drivers make their decision based on the observed price in each location, and the time to reach it.

1.2.3 Innovations

Over the last years, on-demand service platforms have emerged as important poles of innovation. Below, some of the main innovations related to on-demand service platforms are summarized:

- *Waiting time differentiation*: Some on-demand service platforms account for heterogeneous willingness to wait and to pay among their customers (see Lyft, 2021a; Uber, 2021). To this end, they propose a menu of prices for an almost identical service, with the service quality determined by the customer waiting time. The platform thus allocates service providers to the most urgent orders, and creates some value by reducing total waiting costs.

- *Integration with public transit systems*: Some ride-hailing platforms adapt their operations to public transit systems, by giving real-time information on public transit system schedules or by suggesting a multi-modal journey to the customers (Uber, 2022). By doing so, a platform can propose a journey that is more suited to the customer's needs. The public transit systems may also become more accessible, as on-demand service platforms can act as first-mile and last-mile connectors for customers from remote areas (Feng et al., 2022).

- *Ride pooling*: Ride pooling allows customers to share a ride with other customers, in exchange for a discounted price. This option is a way for a ride-hailing platform to differentiate customers, proposing a cheaper option to customers with low willingness to pay (Jacob & Roet-Green, 2021). It is also useful during peak hours when there is an insufficient supply to meet the entire demand. With ride pooling, ride-hailing platforms can satisfy more demand, while using fewer drivers and reducing the cumulative trip lengths (Alonso-Mora et al., 2017).

- *Autonomous vehicles*: Allowed by recent progress in autonomous control and machine learning, autonomous vehicles are expected to bring important changes to on-demand service platforms. In particular, autonomous vehicles have the potential to transform on-demand service platforms by (i) reducing the operating costs, (ii) reducing the number of injuries, and (iii) increasing the control of the platform (see, e.g., Al-Kanj et al., 2020; Mao et al., 2020).

Waiting time differentiation and the integration of public transit are two topics studied in this thesis. Chapter 2 considers the problem of managing priorities in the context of waiting time differentiation. Two classes of customers arrive with distinct willingness to wait and to pay, and the platform

decides whether to admit each incoming request and whether to reserve service providers for high-priority customers (a simplified form of dispatching). Chapter 4 considers the problem that arises when a platform integrates with public transit transfers into its dispatching process. In the latter case, the platform provides first-mile, last-mile, and door-to-door services to customers who arrive in multiple locations at different times.

1.3 Dynamic Optimization

This section presents the main notions of dynamic optimization. Section 1.3.1 introduces Markov decision processes, a modeling framework for sequential decision-making. Section 1.3.2 discusses exact and approximate methods to solve Markov decision processes.

1.3.1 Markov Decision Processes

Originating from the work by Bellman (1957), Markov Decision Processes (MDPs) provide a versatile framework for representing decision-making problems under uncertainty. In brief, they extend the concept of Markov chains by introducing a decision maker who interacts with the system repeatedly. At each epoch, the decision maker selects an action that influences the system transition and the reward received. By doing this, the decision maker can act strategically to achieve a predefined goal (e.g., maximizing profit).

For further intuition, Fig. 1.2 describes the interaction between the decision maker and the system at epoch t. First, the agent receives a representation of the system, defined as the state s_t. Based on that state, the agent then takes an action a_t over a set of available actions $\mathcal{A}(s_t)$. In return, the agent collects a numerical reward R_t that informs the agent of the relevance of the action chosen. At the same time, the system transitions from the current state s_t to

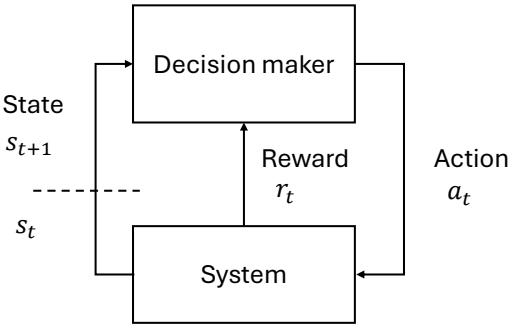


Figure 1.2: The interaction between a decision maker and the system in a Markov decision process.

the next state s_{t+1}, triggering another interaction. All in all, the interactions between the agent and the system generate the following sequence:

$$\{s_0, a_0, R_0, s_1, a_1, R_1, s_2, a_2, R_2, \ldots\}$$

More formally, any MDP can be characterized by four main components:

- The *state* s_t consists of all information relevant to model the system from epoch t onward. It can include elements to describe the system now (e.g., currently available supply), or elements to predict the system in the future (e.g., demand forecasts). The set of all possible states is called the state space.

- The *decision* a_t refers to the decision selected at epoch t. It can be binary, discrete, or continuous, and scalar or vectorial. The set of all possible decisions is called the decision space.

- The *transition function*, denoted by $\mathbb{P}(s_{t+1} \mid s_t, a_t)$, returns the probability of reaching the state s_{t+1} given that the system is in state s_t and the decision a_t is selected. Importantly, the transition function always respects the Markov property, i.e., $\mathbb{P}(s_{t+1} \mid s_0, a_0, \ldots, s_t, a_t) = \mathbb{P}(s_{t+1} \mid s_t, a_t)$. Although the transition function necessarily exists, it may be unknown to the decision maker at the time of the decision.

- The *reward function*, denoted by $R_t(s_t, a_t)$, gives the reward that is obtained from state s_t at epoch t, if the decision a_t is taken. Similar to the transition function, the reward function only depends on the state and decision at epoch t, and is not always known by the decision maker.

Example 1.1. *(inventory problem, inspired from Powell, 2007, exercise 3.1). A company monitors the inventory of a product over time. During day t, demand d_t arrives according to some fixed distribution with discrete support $\{0, \ldots, D\}$. The company then satisfies customers using available stock. Every unsatisfied demand leads to a penalty cost p for the company. At the end of the day, the company incurs a holding cost h for each unit of product remaining in stock. In this setting, the company determines how much to order each day to minimize its costs. This problem can be formulated as an MDP:*

- *The state s_t is the inventory level at (the end of) day t,*

- *The decision a_t is the quantity ordered at day t.*

- *The transition function is $\mathbb{P}(s_{t+1} \mid s_t, a_t) = \mathbb{P}(d_t = s_t + a_t - s_{t+1})$ if $s_{t+1} > 0$, and $\mathbb{P}(0 \mid s_t, a_t) = \sum_{k=s_t+a_t}^{D} \mathbb{P}(d_t = k)$ otherwise.*

- *The reward function is $R_t(s_t, a_t) = h \cdot \max(s_t - d_t, 0) + p \cdot \max(d_t - s_t, 0)$.*

Often, the decision maker selects decisions following a policy π. A policy π is a sequence $\pi = \{\pi_0, \pi_1, \pi_2, \ldots\}$ where π_t is a function mapping the states to decisions made in those states at epoch t. The ultimate objective for the decision maker is then to find an optimal policy π^* that maximizes the expected total reward (Powell, 2007). Accordingly, the optimal policy π^* can be expressed as

$$\pi^* = \arg\max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^t \cdot R_t(s_t, \pi_t(s_t)) \mid s_0, \Big], \tag{1.1}$$

where γ is a discount rate with $0 \leq \gamma \leq 1$, and T can be finite or infinite.

1.3.2 Solution Methods

To find optimal policies (or near-optimal policies), various approaches have been developed. Broadly speaking, solution methods in dynamic optimization can be divided into three categories: value-based methods, policy-based methods, and hybrid methods.

Value-Based Methods

Value-based methods aim to compute optimal policies using the concept of *value function*. The value function $v_\pi(\mathbf{s}_t)$ measures the total future reward that can be expected from a given state $\mathbf{s}_t$ following some policy π, i.e., $v_\pi = \mathbb{E}\Big[\sum_{k=t}^{T-1} \gamma^k \cdot R_k(s_k, \pi_k(s_k)) \mid s_t \Big]$. The optimal value function is then the value function associated with the optimal policy. It can be defined recursively as

$$v^*(\mathbf{s}_t) = \max_{\pi_t(\mathbf{s}_t)} \mathbb{E}_\pi \Big[R_t\big(\mathbf{s}_t,\, \pi_t(\mathbf{s}_t)\big) + v^*(\mathbf{s}_{t+1}) \,\Big|\, \mathbf{s}_t \Big]. \tag{1.2}$$

Finding the optimal value function as formulated in Eq. (1.2) implies solving a set of nonlinear equations, known as the Bellman's optimality equations (Bellman, 1957). Several exact methods exist to compute the optimal policy from Eq.(1.2). A notorious approach is the *value iteration algorithm* (Puterman, 2014). Following this approach, the decision maker does not directly interact with the system to find the optimal policy. Instead, she relies on her perfect knowledge of the system dynamics, i.e., the transition and the reward function. The decision maker starts with some arbitrary estimate $v_0(s_t)$ for each state s_t. Then, at each iteration n, the algorithm updates the value of each state, such as

$$v_n(s_t) = \max_{a_t} \left\{ \mathbb{E}[R_t(s_t, a_t)] + \gamma \cdot \sum_{s'} \mathbb{P}\left(s' \mid s_t, a_t\right) \cdot v_{n-1}\left(s'\right) \right\}. \tag{1.3}$$

Ultimately, with a finite number of states and decisions, the function v_n is guaranteed to converge to v^*, i.e., $\lim_{n\to\infty} v_n(s_t) = v^*(s_t)$. Once it happens, the optimal policy can be deduced:

$$\pi_t^*(s_t) = \arg\max_{a_t} \left\{ \mathbb{E}[R_t(s_t,\, a_t)] + \gamma \cdot \sum_{s'} \mathbb{P}\left(s' \mid s_t,\, a_t\right) \cdot v^*\left(s'\right) \right\}. \tag{1.4}$$

Chapter 2 uses value iteration to find the optimal policy. The algorithm also permits us to derive theoretical results about the structure of the optimal policy by exploiting properties of the so-called "Bellman operator" (i.e., the right-hand side term in Eq. (1.3)). Extensive literature has examined the link between the Bellman operator and the structural properties of optimal policies. We refer to Koole (2007) for a detailed introduction.

It is often the case that $v^*(s_t)$ cannot be computed exactly using value iteration, as the system dynamics are initially unknown to the decision maker (Bertsekas, 2019). Then, simulation methods must be employed. The second approach, named the *Q-learning algorithm*, comes from computer science and the closely related field of reinforcement learning (Sutton & Barto, 2018). According to this approach, the optimal value function is learned from experiences, by interacting with the system. The decision maker stores and updates the estimate $q_t(s_t,\, a_t)$ of the optimal action-value function $q^*(s_t,\, a_t)$, which evaluates at epoch t the value of being in a state s_t, taking decision a_t, and then following the optimal policy π^*. This estimate guides the decision maker's choices, and, at each epoch t, is updated as

$$q_t(s_t,\, a_t) = (1-\alpha_t)\cdot q_{t-1}(s_t,\, a_t) + \alpha_t \cdot \Big[R_t(s_t, a_t) + \gamma \cdot \max_a q_{t-1}(s_{t+1},\, a) \Big], \tag{1.5}$$

where α_t is the learning rate with $0 \leq \alpha_t < 1$. When the number of states and decisions is finite, the estimates $q_t(s_t,\, a_t)$ converge to the optimal values $q^*(s_t,\, a_t)$, provided that (i) states are visited an infinite amount of times (which can be ensured, e.g., by selecting decisions randomly with some positive probability ε), and (ii) that the learning rate is set such that $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$ (Watkins & Dayan, 1992). The optimal value function, and optimal policy are respectively given by $v^*(s_t) = \max_{a_t} q^*(s_t,\, a_t)$, and $\pi_t^*(s_t) = \arg\max_{a_t} q^*(s_t,\, a_t)$.

While the concept of value function is theoretically elegant, it is typically not suited for many real-world problems characterized by an exponential number of states, decisions, and transitions (the well-known "curses of dimensionality"). To address this issue, a common strategy is to replace the (action-)value function by some parametrized approximation $\widehat{v}(\mathbf{s}_t,\, \eta)$ with η denoting the parameter set. For instance, the approximate value function could be a linear function such as

$$\widehat{v}(\mathbf{s}_t,\, \eta) = \eta_0 + \eta_1 \cdot f_1(s_t) + \eta_2 \cdot f_2(s_t) + \ldots \tag{1.6}$$

The two algorithms presented before can then be adapted to estimate the approximate optimal value function $\widehat{v}^*(\mathbf{s}_t, \eta)$. For instance, using stochastic gradient descent and assuming fixed learning rate α, the update in Eq. (1.5) becomes

$$\eta_n = \eta_{n-1} + \alpha \cdot \Big[R_t(s_t, a_t) + \gamma \cdot \max_a q_{n-1}(s_{t+1}, a, \eta) \Big] \nabla_\eta \, q_{n-1}(s_t, a_t, \eta). \quad (1.7)$$

With adequate approximation, near-optimal policies can be derived. This observation has led to many specialized algorithms, which are often referred to as approximate dynamic optimization (Powell, 2007). In parallel, other algorithms exist that derive near-optimal policies without using value functions. The next section presents some of these methods, which learn policies as parametric functions.

Policy-Based Methods

Policy-based methods differ from value-based methods in that they do not find near-optimal policies from the value function. Instead, they aim at deriving a policy $\pi^*(f(\theta))$ that selects decisions according to some parametric function $f(\theta)$ with parameter set θ. Following Powell (2019), policy-based methods thus aim to find the optimal parameters θ^* such that

$$\theta^* = \arg \max_{f(\theta) \in \mathcal{F}} \mathbb{E}\Big[\sum_{t=0}^{T-1} \gamma^t \cdot R_t\big(s_t, \pi_t(s_t, f(\theta))\big) \mid s_0, \Big], \quad (1.8)$$

where $\mathcal{F}$ is some class of functions (e.g., the linear functions). If $\mathcal{F}$ comprises a parametric function specifying optimal policy π^*, then finding θ^* in Eq. (1.8) leads to π^*.

Example 1.2. *(cont'd). The company has heard of the work by Scarf et al. (1960), and knows that the optimal policy to manage its inventory takes the form of a base-stock (s, S)-policy. According to this class of policy, the company orders products as soon as the inventory level becomes inferior to s, in order to reach the inventory level S. At day t, the decision can then be described as*

$$\pi_t(s_t, \theta) = \begin{cases} S - s_t, & \text{if} \quad s_t < s \\ 0, & \text{otherwise.} \end{cases}$$

A base-stock (s, S)-policy only needs the parameter set $\theta = (s, S)$ to be fully specified. Thus, deducing the optimal policy π^* reduces to finding the optimal parameter set θ^*.

Base-stock policies are particular cases where the optimal policy has a simple structure. In practice, however, optimal policies are often much more complex. As a result, the function class $\mathcal{F}$ may not include the parametric function (or the parameter set) describing the optimal policy (Powell, 2019). We can

turn to universal approximators such as neural networks (LeCun et al., 2015) to expand the search space, and use stochastic gradient ascent as in Eq. (1.7). Yet, there is little hope to discover the optimal policy with gradient ascent if the associated parametric function is nonconvex (Goodfellow et al., 2016, Chapter 4).

Despite these limitations, policy-based methods still present several advantages. First, they are not limited to deterministic policies as value-based methods, but allow for stochastic policies (Sutton & Barto, 2018). This feature is especially helpful when the system can unpredictably adapt to the decision maker's decisions (consider, e.g., a decision maker playing poker). Second, they can mitigate tractability issues resulting from a large number of decisions or continuous decisions (see, e.g., Lillicrap et al., 2015). This second feature is especially relevant for us. In Chapter 3, the parameter set θ takes the form of a neural network, and continuous decisions are sampled from normal distributions such as $\pi_t(s_t, \theta) \sim \mathcal{N}\big(\mu(s_t,\, \theta),\, \sigma(s_t,\, \theta)\big)$, where $\mu(s_t,\, \theta)$ and $\sigma(s_t,\, \theta)$ are parametrized functions, with $\sigma(s_t,\, \theta)$ ensuring exploration. Without policy-based methods, the MDP formulated in Chapter 3 would have been intractable.

Hybrid Methods and Proximal Policy Optimization

Often, value-based methods and policy-based methods are combined to leverage their respective strengths. These methods are usually called "actor-critic" methods. On the one hand, the actor refers to the approximation of the policy, $\pi(\theta)$, and determines the decisions to make for every state encountered. On the other hand, the critic refers to the approximation of the value function $\hat{v}(\eta)$ (Sutton & Barto, 2018) and evaluates the relevance of the parametrized policy $\pi(\theta)$.

In Chapter 2, we apply an actor-critic algorithm, named Proximal Policy Optimization (PPO). This algorithm, proposed by Schulman et al. (2017), approximates the value function by minimizing the following loss function:

$$L^{\text{VF}}(\eta) = \sum_{t=0}^{T-1} \sum_{\mathbf{s}_t \in \mathcal{S}} \mathbb{P}(\mathbf{s}_t) \Big[\frac{1}{2} \Big(\hat{v}(s_t,\, \eta) - G_t \Big)^2 \Big], \tag{1.9}$$

where G_t denotes the total realized profit from state $\mathbf{s}_t$ until the end of the horizon under policy $\pi(\theta)$, i.e., $G_t = \sum_{k=t}^{T-1} R_k\big(s_k, \pi_k(s_k, \theta)\big)$. Eq. (1.9) aims to quantify the difference between the approximate value function $\hat{v}_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t,\, \boldsymbol{\eta})$ and G_t, an unbiased estimate of the true value function $v_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t,\, \boldsymbol{\eta})$ for following policy $\pi(\boldsymbol{\theta})$ at epoch t.

At the same time, to learn an approximation of the optimal policy, PPO finds the parameters θ maximizing the clipped loss function:

$$L^{\text{CLIP}}(\theta) = \sum_{t=0}^{T-1} \sum_{\mathbf{s}_t \in \mathcal{S}} \mathbb{P}(\mathbf{s}_t) \left[\min \left(pr(\theta) \cdot A_t,\, \text{clip} \Big(pr(\theta),\, \big[1-\varepsilon, 1+\varepsilon\big] \Big) \cdot A_t \right) \right], \tag{1.10}$$

where $\mathbb{P}(\mathbf{s}_t)$ is the probability of being in system state $\mathbf{s_t}$ at epoch t following policy $\pi(\boldsymbol{\theta})$; $pr(\boldsymbol{\theta}) = \frac{\pi(\mathbf{s}_t, \boldsymbol{\theta})}{\pi(\mathbf{s}_t, \boldsymbol{\theta}_{old})}$ is the ratio between the current policy output and the old policy output; A_t is the advantage estimate such that $A_t = R_t + \hat{v}(s_{t+1}, \eta) - \hat{v}(s_t, \eta)$; and ε is the clipping parameter limiting the policy update. For a detailed interpretation of Eq. (1.10), we refer the reader to Schulman et al. (2017). In brief, the clipped function $L^{\text{CLIP}}(\theta)$ maximizes future rewards as a function of the parameters θ. Yet, it ensures that the ratio $pr(\theta)$ lies within the interval $[1-\varepsilon, 1+\varepsilon]$ where ε is a tunable parameter between 0 and 1. This last feature avoids extreme policy changes, and stabilizes the learning process of PPO.

The PPO algorithm is summarized in Algorithm 1. While the termination condition is not met (e.g., the average reward has not improved by at least 1% over ten consecutive evaluations), PPO first simulates the policy $\pi(\theta)$ for T^{PPO} decision epochs, collecting states and rewards. Then, the value function estimates G_t and the advantage estimates A_t are calculated for the collected states. After that, the collected states, value function estimates $\hat{v}(s_t, \eta)$, and advantage estimates A_t are divided into smaller sets, called mini-batches. For each mini-batch, the gradients of Eq. (1.9), and Eq. (1.10) are computed. Finally, the parameters θ and η are updated using stochastic gradient descent. When an update is done for each mini-batch, one training epoch is said to be completed. New mini-batches are made, and gradient updates are performed for a total of E^{PPO} training epochs.

1.4 Stochastic Optimization

In the following, we introduce several concepts from stochastic optimization that are used throughout the thesis. Section 1.4.1 presents two-stage problems, a modeling framework supporting stochastic optimization. Section 1.4.2 gives a brief overview of solution methods, focusing on the L-shaped method.

Algorithm 1: Proximal Policy Optimization (PPO)

while termination condition is not met **do**
  Run the policy $\pi(\theta)$ for T^{PPO} decision epochs.
  Compute the value function estimates G_t, and advantage estimates A_t.
  for E^{PPO} training epochs **do**
    Distribute the states, value function, and advantage estimates into mini-batches.
    Compute $\nabla_\eta \widehat{L^{\text{VF}}}(\eta)$, and $\nabla_\theta \widehat{L^{\text{CLIP}}}(\theta)$.
    Update $\pi(\theta)$ such that $\theta \leftarrow \theta + \alpha \nabla_\theta \widehat{L^{\text{CLIP}}}(\theta)$.
    Update $\hat{v}(\eta)$ such that $\eta \leftarrow \eta - \alpha \nabla_\eta \widehat{L^{\text{VF}}}(\eta)$.
  end
end

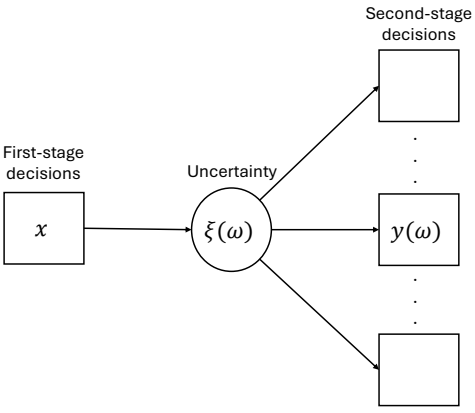


Figure 1.3: Stochastic optimization as a two-stage decision process.

1.4.1 Two-Stage Problems

Following the seminal work by Dantzig (1955), scholars in operations research have extensively relied on stochastic optimization to deal with uncertainty. Broadly speaking, stochastic optimization can be viewed as a generalization of deterministic optimization, where some parameters of the optimization problem are uncertain. The optimization problem then accounts for the (partially) known distributions of these parameters, determining decisions that are optimal in expectation. These decisions are taken sequentially, before random outcomes are revealed.

Fig. 1.3 illustrates the general structure underlying stochastic optimization problems. First, some decisions are made before uncertainty is revealed. These decisions are referred to as *first-stage decisions*, and are denoted by x. The time when they are made is known as the *first stage*. Next, the uncertainty is revealed, modeled by random variable(s) $\xi(\omega)$, where ω represents an outcome in the sample space Ω. Finally, additional decisions are taken, in response to uncertainty. These decisions are named *second-stage decisions* (or recourse decisions), and are denoted by y. They occur during the so-called *second stage*.

Following Birge and Louveaux (2011), the two-stage linear problem can be formulated as

$$\min c^T x + \mathbb{E}_\xi \left[\min q(\omega)^T y(\omega)\right] \tag{1.11a}$$
$$\text{s.t.} \quad Ax = b, \tag{1.11b}$$
$$T(\omega)x + W(\omega)y(\omega) = h(\omega), \tag{1.11c}$$
$$x \geq 0,\ y(\omega) \geq 0. \tag{1.11d}$$

In this formulation, the first-stage decisions are variables $x \in \mathbb{R}^{n_1}$, and the second-stage decisions are variables $y \in \mathbb{R}^{n_2}$. The first-stage parameters

are deterministic, and given by $c \in \mathbb{R}^{n_1}, b \in \mathbb{R}^{m_1}$, and $A \in \mathbb{R}^{m_1 \times n_1}$. The second-stage parameters are possibly stochastic, and given by $q(\omega) \in \mathbb{R}^{n_2}$, $h(\omega) \in \mathbb{R}^{m_2}, T(\omega) \in \mathbb{R}^{m_2 \times n_1}$, and $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$.

Example 1.3. *(the newsvendor problem, Shapiro et al., 2021, Chapter 1). Consider the problem of a company that manufactures a product by some quantity x in the current period, and sells it by a quantity y during the subsequent period. Due to limited capacity, the company cannot produce more than u units of the product in the current period. At the subsequent period, the company cannot sell more units than the observed demand $d(\omega)$, which is uncertain and characterized by the random outcome $\omega \in \Omega$. The company aims to maximize profit, which depends on the marginal production c and the marginal selling price p. Then, we can express the problem as follows:*

$$\min c \cdot x + \mathbb{E}_{d(\omega)} \left[\min -p \cdot y(\omega)\right] \tag{1.12a}$$

$$s.t. \quad x \leq u, \tag{1.12b}$$

$$y(\omega) \leq x, \tag{1.12c}$$

$$y(\omega) \leq d(\omega), \tag{1.12d}$$

$$x \geq 0,\ y(\omega) \geq 0. \tag{1.12e}$$

In reality, only a limited number of two-stage problems exist, like the one just presented. Multi-stage problems are more common, but also harder to solve. For this reason, the stochastic optimization community typically employs two-stage problems to approximate multi-stage problems. The idea is to embed the two-stage problem in a rolling horizon strategy, so as to support decisions made repeatedly over time. The first-stage variables x then correspond to decisions made at the current epoch t, while the second-stage variables $y(\omega)$ represent future decisions, possibly spanning over several epochs. This strategy has proven to be relevant in practice (cf. Birge, 1997), and is the one followed in Chapter 4. Given that multi-stage problems fall outside the scope of this thesis, we only present solution methods for two-stage problems.

1.4.2 Solution Methods

We start by presenting deterministic formulations for two-stage problems, using scenarios. Then, we detail the L-shaped method, an algorithm to solve large two-stage problems.

The Deterministic Formulation

In general, it is impossible to exactly solve two-stage problems, as they represent a large (or even infinite) number of connected deterministic problems, one per outcome $\omega \in \Omega$. For this reason, the stochastic optimization community often solves approximations of two-stage problems, which only account for some smaller, discrete sample space $\mathcal{S} \subset \Omega$. This reduced sample space $\mathcal{S}$ is called

the scenario set. Each outcome $s \in \mathcal{S}$ refers to a scenario, and is associated with a probability $\mathbb{P}_s$ such that $\sum_{s\in\mathcal{S}} \mathbb{P}_s = 1$.

With the scenario set $\mathcal{S}$, the problem (1.11) can be approximated as

$$\min c^T x + \sum_{s=1}^{S} \mathbb{P}_s \cdot q_s^T y_s \tag{1.13a}$$

$$\text{s.t.} \quad Ax = b, \tag{1.13b}$$

$$T_s x + W_s y_s = h_s, \qquad \forall s = 1, \ldots, S, \tag{1.13c}$$

$$x \geq 0,\ y_s \geq 0, \qquad \forall\, s = 1, \ldots, S. \tag{1.13d}$$

The above formulation, often named the *deterministic equivalent*, is nothing but a linear optimization problem. When the number of scenarios is limited, the formulation can be solved efficiently, using standard algorithms such as the simplex method.

Naturally, the quality of the optimal solution directly depends on how well the scenario set $\mathcal{S}$ approximates the real sample space Ω. It is therefore not surprising that researchers have developed several methods to generate scenarios accurately. These methods include scenario tree reduction (Heitsch & Römisch, 2009), scenario sampling (Kleywegt et al., 2002), and moment matching (Høyland et al., 2003), among others. In Chapter 4, we adopt a traditional approach, by generating scenarios from historical data without any transformation. In our case, this proved sufficient to produce high-quality solutions.

Sometimes, a large number of scenarios is still required to obtain a relevant approximation. As the number of scenarios grows, the problem size becomes prohibitive, and specialized procedures must be developed. To tackle such large problems, various methods have been designed (see, e.g. Rockafellar & Wets, 1991; Pereira & Pinto, 1991) In particular, Van Slyke and Wets (1969) propose a decomposition method, called the L-shaped method. Since this method is applied in Chapter 4, we give a detailed introduction to it in the next section.

The L-Shaped Method

In this section, we present a variant of the L-shaped method, called the *multicut* L-shaped method (Birge & Louveaux, 1988). At its core, this method uses Benders decomposition (1962) to decompose the problem (1.13) as

$$\min z = c^T x + \sum_{s=1}^{S} \mathbb{P}_s \cdot \mathcal{Q}(x, s) \tag{1.14a}$$

$$\text{s.t.} \ Ax = b, \tag{1.14b}$$

$$x \geq 0. \tag{1.14c}$$

where, for all $s = 1, \ldots, S$,

$$\begin{aligned} Q(x,s) \equiv \min\, & q_s^T y_s \\ \text{s.t. } & W_s y_s = h_s - T_s x, \\ & y_s \geq 0. \end{aligned} \tag{1.15}$$

Let $\mathcal{K}^1 \equiv \{x \mid Ax = b, x \geq 0\}$ be the set of feasible first-stage decisions. In addition, let $\mathcal{K}_s^2 \equiv \{x \mid \exists\, y \geq 0,\ \text{s.t. } W_s y = h_s - T_s x\}$ $\left(\text{with } \mathcal{K}_2 = \bigcap_s \mathcal{K}_s^2\right)$ be the set of first-stage decisions that have feasible second-stage decisions for a given scenario s. We also define θ_s as the variable bounding the cost of the second-stage problems for scenario s. Using this notation, the problem (1.11) can be reformulated as follows:

$$\min z = c^T x + \sum_{s=1}^{S} \mathbb{P}_s \cdot \theta_s \tag{1.16a}$$

$$\text{s.t. } x \in \mathcal{K}^1, \tag{1.16b}$$

$$x \in \mathcal{K}^2, \tag{1.16c}$$

$$\theta_s \geq \mathcal{Q}(x,s), \qquad \forall\, s = 1, ..., S. \tag{1.16d}$$

The central idea of the L-shaped method is to drop constraints (1.16c), (1.16d), and to reconstruct them iteratively (Birge & Louveaux, 2011). By doing this, the method only includes relevant constraints in the problem, which becomes more tractable.

The L-shaped method is summarized in Algorithm 2. At each iteration m, the method first solves the master problem to obtain some candidate solution $(x^m,\, \theta_s^m)$:

$$\min c^T x + \sum_{s=1}^{S} \mathbb{P}_s \cdot \theta_s \tag{1.17a}$$

Algorithm 2: The multicut L-shaped method

```
for iteration m = 1,... do
    Solve the master problem (with no cuts initially) to find (x^m, θ_s^m).
    for scenario s = 1, ..., S do
        Solve dual feasibility problem F^D(x^m, s).
        if F^D(x^m, s) > 0, then add a feasibility cut, and go to next iteration.
    end
    for scenario s = 1, ..., S do
        Solve dual subproblem Q^D(x^m, s)
    end
    if θ_s^m ≥ Q(x^m, s) for all s, then stop with optimal solution (x^m, θ_s^m).
    else add an optimality cut for all s and go to next iteration.
end
```

$$\text{s.t. } x \in \mathcal{K}_1 \tag{1.17b}$$
$$f_i(x) \leq 0, \qquad \forall\, i = 1, \ldots, I \tag{1.17c}$$
$$g_j(x, \theta_s) \leq 0, \qquad \forall\, j = 1, \ldots, J_s,\ s = 1, \ldots, S, \tag{1.17d}$$

where $I = J_s = 0$ initially. The constraints $f_i(x) \leq 0$ refer to *feasibility cuts*, whereas the constraints $g_j(x, \phi) \leq 0$ refer to *optimality cuts*.

Second, given the candidate solution $(x^m,\, \theta_s^m)$, a feasibility problem is solved for each scenario s:

$$F\left(x^m, s\right) \equiv \min_{y, v_s^+, v_s^-} e^\top v_s^+ + e^\top v_s^- \tag{1.18a}$$
$$\text{s.t. } W_s y_s + I v_s^+ - I v_s^- = h_s - T_s x^m, \tag{1.18b}$$
$$y, v_s^+, v_s^- \geq 0, \tag{1.18c}$$

where I is the identity matrix, e is a column vector of ones, and v_s^+ and v_s^+ are feasibility variable. By strong duality, this is equivalent to solving the problem

$$F^D\left(x^m, s\right) \equiv \max_{\sigma_s} \sigma_s^\top \left(h_s - T_s x^m\right) \tag{1.19a}$$
$$\text{s.t. } \sigma_s^\top W_s \leq 0, \tag{1.19b}$$
$$\sigma_s^\top I \leq e^\top, \tag{1.19c}$$
$$-\sigma_s^\top I \leq e^\top. \tag{1.19d}$$

with dual variables σ_s. If $\sigma_s^\top \left(h_s - T_s x^m\right) = 0$ for all s, then $x^m \in \mathcal{K}^2$. Otherwise, we have $= x^m \notin \mathcal{K}_2$. In the latter case, the feasibility cut $f_i(x) \equiv \left(\sigma_s^m\right)^\top \left(h_s - T_s x\right) \leq 0$ is added to the master problem, and the algorithm iterates, solving the master problem with updated constraints.

Third, the subproblem $\mathcal{Q}\left(x^m, s\right)$ is solved for each scenario s. Again, by strong duality, solving $\mathcal{Q}\left(x^m, s\right)$ boils down to solving its associated dual problem:

$$\mathcal{Q}^D\left(x^m, s\right) = \max_{\rho_s} \rho_s^\top \left(h_s - T_s x^m\right) \tag{1.20a}$$
$$\text{s.t. } \rho_s^\top W_s \leq q_s^\top. \tag{1.20b}$$

Fourth, if $\theta_s^m \leq \mathcal{Q}(x^m,\, s) = \rho_s^m (h_s - T_s x^m)$ for all s, then we conclude that $(x^m,\, \theta_s^m)$ is optimal. Otherwise, we know that constraints (1.16d) is not satisfied at point (x^m, θ_s^m). To ensure the constraints to be respected $\big($at least around $(x^m, \theta_s^m)\big)$, the optimality cut $g_j(x,\, \theta_s) \equiv \left(\rho_s^m\right)^\top \left(h_s - T_s x\right) - \theta_s \leq 0$ is added to master problem for each scenario s. The method then iterates, solving the master problem with updated constraints.

Since there is a finite number of feasibility and optimality cuts, it can be shown that the L-shaped method converges to optimality in finite time. Before

convergence, we can derive lower and upper bounds on the optimal solution z^* at each iteration such that:

$$c^\top x^m + \sum_{s=1}^{S} \mathbb{P}_s \cdot \theta_s^m \leq z^* \leq c^\top x^m + \sum_{s=1}^{S} \mathbb{P}_s \cdot \mathcal{Q}(x^m, s). \tag{1.21}$$

Alternatively, the algorithm can use the gap between these bounds as a stopping criterion, terminating once the gap is sufficiently small (e.g., less than 1%, cf. Chapter 4).

1.5 Contributions

The thesis is organized into three chapters. Each chapter corresponds to a problem related to on-demand service platforms. The first two chapters use dynamic optimization methods, while the third chapter relies on stochastic optimization methods. At the end of the thesis, some concluding remarks are provided to discuss the limitations of the research, while suggesting opportunities for future work. Below, we summarize the next chapters and their contributions.

Chapter 2. On-Demand Service Platforms with Waiting Time Differentiation

Inspired by recent applications, this chapter considers the problem of an on-demand service platform (e.g., Uber, Deliveroo) that differentiates the service of customers based on their experienced waiting time. Two customer classes arrive with distinct willingness to wait and to pay, and the platform determines how to organize their service. Specifically, the platform must make two main decisions. First, the platform determines upon each arrival whether to admit the customer, considering the customer class. Second, the platform decides how to allocate its service providers to the customers of each class that have been admitted. In this setting, the platform aims to minimize costs for customer rejection and customer abandonment.

We formulate this queueing problem as a continuous-time Markov decision process, which is used to find the optimal policy. By decomposing the Bellman optimality equations into several operators, we prove that the optimal value function satisfies several properties, such as supermodularity and Schur-convexity. Exploiting these theoretical results, we then demonstrate that the optimal policy can be described by state-dependent admission thresholds and a strict priority rule with respect to service provider allocation.

In a numerical study, the performance of the optimal policy is compared with simpler policies that only control customer admission, service provider allocation, or apply a simple priority rule. We find that our optimal policy significantly outperforms the policy applying a simple priority rule, by about 25%. Our results also suggest that customer admission and service provider

allocation are complementary, and can significantly outperform other policies under intermediary parameter values. Customer and service providers can also benefit as customer waiting times are maintained at low values while service providers are ensured sufficient utilization. Lastly, we evaluate the robustness of the policy in two extensions. Our policy is found to perform well in both extensions, leading to less than 2% additional costs on average.

Chapter 3. Pricing, Repositioning and Admission in Ride-Hailing Networks

This chapter considers the problem of a ride-hailing platform such as Uber or Lyft, that balances supply and demand over a network of locations. To this aim, the platform makes pricing, driver repositioning, and customer admission decisions at the tactical level (i.e., every 5 minutes). Customers arrive as a function of the location and time. They are impatient and have distinct willingness to pay. Drivers can be repositioned by the platform, or can choose to relocate by themselves. In this setting, the platform's objective is to maximize revenues for serving customers at specific prices, while minimizing costs incurred by repositioning drivers and leaving some requests unsatisfied.

The problem is formulated as a discrete-time Markov decision process that makes both continuous decisions (e.g., pricing) and discrete decisions (e.g., repositioning). Given the problem size and complexity, we develop a reinforcement learning approach with pretraining to find an efficient policy. Our approach first derives a rolling-horizon strategy by repeatedly solving a deterministic convex program based on the expected values of the problem. Then, two neural networks are pretrained to replicate the strategy and learn the associated value function. Finally, the policy is further improved through Proximal Policy Optimization (PPO), an actor-critic algorithm commonly used in reinforcement learning.

Using data from New York City, our approach is applied to networks of up to 20 locations. The results show that our approach outperforms classical PPO and rolling-horizon strategies by 16.7% and 4.1% on average, while improving the training time by 42% in comparison with classical PPO. By analyzing the policies found by our approach, managerial insights are provided on the respective roles of pricing, driver repositioning, and customer admission. We find that prices play an important role, as they adjust spatially, temporally, and dynamically as a function of customer demand. Driver repositioning is also relevant to rebalance the network, and appears to be more effective than driver self-relocation. Lastly, customer admission is only used when driver supply becomes scarce. In that case, it serves as a way to indirectly reposition drivers, by prioritizing requests traveling to convenient locations.

Chapter 4. Ride-Hailing Dispatching with Public Transit Integration

This chapter considers the online dispatching problem of a ride-hailing platform

that integrates public transit transfers into its dispatching process. Customers are impatient and arrive dynamically over time. For each request, the platform must make three decisions. First, the platform decides whether to serve the request, and if so, whether to provide door-to-door service, first-mile service, or last-mile service. Second, if a request receives first-mile service (last-mile service resp.), the platform determines at which transit station to drop off (pick up resp.) the customer. Third, the platform chooses which driver to dispatch to the request.

To address this problem, we propose a lookahead approach based on stochastic optimization. At each decision epoch, the approach solves a two-stage stochastic program using historical data. It can be shown that, under certain conditions, the offline version of the problem with static demand can be solved efficiently through its linear relaxation. Using this result, two methods are developed to improve the tractability of the two-stage program. First, a preprocessing procedure is used to select decision variables that are likely to be optimal, which significantly reduces the problem size. Second, the L-shaped method is applied to decompose and solve the two-stage program as a master problem and a series of subproblems, hence reducing computational effort.

To conduct our numerical experiments, synthetic instances are designed to represent a stylized routing network under various supply, demand, and public transit configurations. Using these instances, we show the relevance of our approach over myopic and deterministic rolling horizon and lookahead strategies. Our results also demonstrate the value of an integrated system using real-world data from NYC. Specifically, we find the platform's profit by 7.3%, the percentage of fulfilled requests by 17.9%, and it can reduce customers' travel time by 5.5%. The integration benefits are more significant when the instances are characterized by excessive demand, high pickup fares, and high-frequency public transit. In addition, by analyzing the trade-off between profit maximization and travel time minimization, we find that the structure of the solutions can significantly vary depending on the platform's objective.

2 On-Demand Service Platforms with Waiting Time Differentiation

De Munck, T., Chevalier, P., and Tancrez, J. S. (2023). Managing priorities on on-demand service platforms with waiting time differentiation. International Journal of Production Economics, 266, 109053.

2.1 Introduction

Over the last decade, the transportation sector has witnessed the spectacular growth of on-demand service platforms (e.g., Uber, Lyft, Deliveroo). In contrast with other platforms (e.g., product-sharing platforms such as Airbnb or Blablacar, and freelancing platforms such as Upwork), on-demand service platforms provide a service that is immediate and generic, rather than scheduled and agent-specific (Taylor, 2018). Recently, multiple on-demand service platforms have considered waiting time differentiation, i.e., the practice of differentiating the service of the customers according to their willingness to wait and to pay. The central idea of waiting time differentiation is to offer a menu of price/waiting time options to the customers, for an almost identical service. Impatient customers are expected to pay a higher price and select the service option with a low waiting time, and conversely.

Similar to other industries (e.g., mail delivery, see Fedex, 2021), there exist several practical illustrations of waiting time differentiation on on-demand service platforms. For instance, Lyft proposes both a "priority pickup" and a "wait-and-save" option to riders in various cities worldwide. With the priority pickup option, riders can pay higher prices to be served in priority, so as to endure shorter waiting times. Conversely, the "wait-and-save" option offers lower prices to riders in exchange for longer waiting times (Lyft, 2021b). In the context of food delivery, platforms have established analogous programs. On Uber Eats, customers can pay more than the regular price for a faster delivery or can pay less for a longer delivery (Lekach, 2020).

Waiting time differentiation can turn out to be a valuable practice, as it allows a more efficient allocation of limited resources (service providers on on-demand service platforms). On the one hand, impatient customers wait for shorter times. On the other hand, patient customers can benefit from reduced prices. Overall, the platform alleviates the total cost associated with customer

waiting. It can then extract some part of the created value, which is potentially shared with the service providers.

Nevertheless, waiting time differentiation can also be challenging. Given that the resources are limited, the service quality of one customer class is influenced by the service quality of the other customer class. For instance, short waiting times for impatient customers come at the cost of long waiting times for patient customers. Thus, a delicate balance must be found when allocating the resources to the customer classes. In contrast with other industries, this challenge is exacerbated for on-demand service platforms as they are often characterized by highly impatient customers, and self-scheduling service providers who decide when to work.

In this chapter, we consider an on-demand service platform that allocates service providers to two customer classes, called express customers and regular customers. In the remainder of this chapter, whenever there is no confusion, we will often refer to "on-demand service platform" and "service provider" as the "platform" and "provider". Express customers are the customers who pay a higher price to obtain shorter waiting times. Regular customers are the customers who pay the regular price, and do not benefit from reduced waiting times. In this setting, priority management involves two types of decisions. First, the platform determines when to admit or reject customers of each class upon arrival. Behind that decision lies a trade-off between admitting (and serving) a maximum number of customers, or rejecting them to avoid system congestion and costly abandonment. Second, the platform decides how to allocate the service providers to the customers of each class who are waiting for service. Because express customers are more impatient and bring higher profits, the platform may reserve service providers for later arrivals of express customers. As a result, the platform may purposely avoid serving regular customers who have been admitted.

To improve the understanding of the dynamics behind these two decisions, we develop a stylized model formulated as a Markov decision process. The model considers a fixed fleet of service providers and two customer classes that may abandon after exponentially distributed maximum waiting times. The model aims to determine the dynamic policy that minimizes the platform's total cost. The total cost is composed of costs that are associated with customer rejection, waiting time, and abandonment. Each of these costs varies according to the customer class.

The main findings of our research can be summarized as follows.

- First, we consider an unstudied setting with customer impatience, multiple servers (service providers in our application), customer classes, and a policy that integrates control over customer admission and service provider reservation. The policy, named the admission and reservation policy (ARP), prescribes whether to admit express and regular customers upon arrival and, once the customers have been admitted, whether to reserve a service provider for express customers. ARP can be used to gain

insight into how to manage priorities in the context of waiting time differentiation on on-demand service platforms.

- Second, by using properties of the value function, we theoretically characterize the structure of ARP. Specifically, we establish the existence of state-dependent admission thresholds for both customer classes, and the existence of a strict priority rule with respect to service provider allocation (always serve express customers in priority) when customer abandonment is exponentially distributed.

- Third, we show the benefits of jointly controlling customer admission and service provider reservation in a numerical study. Our results underline that admission and reservation control can often be used complementarily by ARP. This leads to positive implications for the platform, the customers, and service providers, which are analyzed in detail.

- Fourth, we provide some evidence of the robustness of ARP in two extensions. Namely, ARP is found to perform well in environments with self-scheduling service providers (first extension), or non-exponentially distributed customer impatience (second extension). ARP generates on average only 1.29% and 1.67% of additional costs when applied to the first and the second extension, respectively.

The remainder of the chapter is organized as follows. In Section 2.2, we review and discuss the related literature. In Section 2.3, we formulate our Markov decision process model for the problem considered and we characterize the structure of the optimal policy. In Section 2.4, we show the value of jointly controlling customer admission and service provider reservation in a numerical study. In Section 2.5, we investigate two extensions to our initial model to show the robustness of ARP. In Section 2.6, we conclude and propose potential research directions.

2.2 Literature Review

This chapter contributes to the growing literature on on-demand service platforms and is related to the literature on waiting time differentiation and controlled queueing systems. In this section, we review these three literature streams.

The recent literature on on-demand service platforms, reviewed by Benjaafar and Hu (2020), can roughly be divided into two streams: the first stream explores the allocation of spatially distributed resources on on-demand service platforms, and the second stream examines their pricing strategy, given that both supply and demand are functions of the price. In the stream on on-demand platforms with spatially distributed resources, several papers, including by Banerjee et al. (2022a), Bimpikis et al. (2019) and Wu et al. (2020), analyze spatial pricing, that is, the practice of setting the service price as a

function of the origin (and possibly destination) location. Other studies, like those by Aouad and Saritaç (2022), Al-Kanj et al. (2020), Özkan and Ward (2020), and M. Hu and Zhou (2022), develop matching models where the platform pairs riders and drivers from different locations. Lastly, a few papers have considered routing and repositioning models where the platform dispatches service providers within the network (e.g. Afèche et al., 2023, Braverman et al., 2019, Shou et al., 2020). The mentioned papers show the importance of the spatial component for on-demand service platforms. In this chapter, we rather focus on differentiating customers according to their willingness to wait and to pay. We thus capture the main characteristics of waiting time differentiation in a model with a single location.

The stream on pricing strategies for on-demand service platforms has devoted special attention to surge pricing, that is, the practice of setting the service price according to the supply and demand observed in real time. Banerjee et al. (2016) design a queueing model to compare surge pricing and time-based pricing. With this model, they find that time-based pricing maximizes the throughput and the platform's revenue. On the other hand, they note that surge pricing is more robust to parameter uncertainties. Cachon et al. (2017) use a revenue management framework to show that surge pricing enhances the customers' welfare in addition to the platform's revenue. Castillo et al. (2017) argue that surge pricing may circumvent the so-called "wild goose chase" effect, which tends to assign drivers to distant riders during peak hours. Closer to our work, Taylor (2018), Bai et al. (2019), and Zhong et al. (2020) also devise queueing models, but that explicitly capture the impatience of customers. In the models by Taylor (2018) and Bai et al. (2019), one class of strategic customers decides whether to request a service or to leave the platform depending on the price and expected waiting time. These customers differ by their service valuations but have homogeneous impatience, which is materialized through a common marginal waiting cost. Zhong et al. (2020) build on their work by accounting for customers with heterogeneous impatience. Zhong et al. (2020) focus on the choice made by the platform between serving only customers with low impatience and serving customers with low and high impatience. In their paper, the service is homogeneous for the two customer classes. Our model proposes a novel perspective, as it considers a differentiated service, with distinct prices and average waiting times for each customer class. In contrast, Zhong et al. (2020) only consider an undifferentiated service with the same price and expected waiting time for all the customers. Further, we represent customer impatience through exponentially distributed maximum waiting times rather than as a strategic decision based on customers' expectations.

In the waiting time differentiation literature, a recurring challenge is to determine incentive-compatible prices so that the total welfare or the firm's revenue is maximized (see, Hassin, 2016, for a review). In a landmark paper, Mendelson and Whang (1990) prove that welfare-maximizing prices are also incentive-compatible. Afèche and Pavlin (2013, 2016) explore the issue of strategic delay, and find that a profit-maximizing firm may have an interest in

artificially delaying the service of its customers. Jayaswal et al. (2011) and Zhao et al. (2012) study the market characteristics (company's lead time, customer's price sensitivity) and the firm's operations strategies (dedicated vs. shared capacity) that favor the implementation of waiting time differentiation. Jacob and Roet-Green (2021) make a direct connection between the literature on waiting time differentiation and on-demand service platforms. In their model, a ride-hailing platform is concerned with determining the profit-maximizing and incentive-compatible menu of prices when customers can decide whether to share a ride (i.e., ride pooling) or to travel solo. They then identify the conditions under which the platform may benefit from offering pooled rides. The present chapter complements their research by deriving the optimal policy with respect to customer admission and service provider reservation, given that a menu of prices has already been derived and that customers have made their choice. As such, we do not formally define the incentive-compatibility constraints of the customers in our model, but we focus on more complex dynamic policies than the current literature.

The literature on controlled queueing systems is extensive, dating back to Naor (1969). The seminal papers by Smith et al. (1956) and Cox and Smith (1961) show that the so-called "$c\mu-$rule" is optimal to allocate a server to classes of customers with distinct holding costs and service rates in a M/G/1 queueing system. Since then, these results have been generalized to many different queueing scheduling problems, including those with multiple servers (Van Mieghem, 1995, Mandelbaum & Stolyar, 2004) and customer abandonment (Atar et al., 2010). Turning to more general queueing systems (i.e., systems that do not only consider server allocation as the sole decision), Stidham and Weber (1989) provide a first attempt toward a general method to characterize the optimal policy in a queueing system. Other approaches like the ones proposed by Koole (1998), Puterman (2014), and Vercraene et al. (2018) have also been developed to derive theoretical results on monotone optimal policies. Using these theoretical results, several applications have been examined. Subramanian et al. (1999) tackle the problem of allocating flight seats to multiple classes of customers. Papadaki and Powell (2002) focus on serving customers in batches. Iravani et al. (2012) investigate a make-to-stock/make-to-order system serving two customer classes. W. Ma et al. (2023) show the value of real-time information in a make-to-stock system with one class of impatient customers. More closely related to our work, Altman et al. (2001), Örmeci et al. (2001), and Savin et al. (2005) study different versions of the stochastic knapsack problem, introduced by Ross and Tsang (1989). This problem consists of determining the optimal admission policy for a multi-server loss queueing system serving several customer classes with different revenues and service rates. Once customers are admitted, resources are automatically allocated to them. In contrast, our model allows customers to wait for service in dedicated queues, and the decisions of admitting customers and allocating them resources are distinguished. A model that also allows admitted customers to wait for service is the one proposed by Benjaafar et al. (2010), which controls the admission

and service of two customer classes. However, their model applies to a production and inventory system rather than to an on-demand service platform. It therefore involves a single server serving infinitely patient customers, unlike our model, which is characterized by several servers and impatient customers.

In view of this literature review, this chapter makes two main contributions. First, despite its practical value, it appears that waiting time differentiation has only been partially covered by the literature on on-demand service platforms. Except for Jacob and Roet-Green (2021) in the context of ride pooling, it seems that we are the first to investigate the issue of waiting time differentiation for on-demand service platforms. We complement Jacob and Roet-Green (2021), who focus on the characterizations of prices, by looking at the optimal dynamic policy that would support a given menu of prices. Second, from a modeling perspective, we design a stylized model that combines both controls over customer admission and service provider allocation. While this model is particularly relevant to on-demand service platforms, the literature on controlled queueing systems has devoted only limited attention to it. Benjaafar et al. (2010) has designed an akin model, but in the case of inventory management.

2.3 Markovian Model

In this section, we introduce the problem of an on-demand service platform that serves two customer classes by making decisions with respect to customer admission and service provider allocation. We then formulate it as a continuous-time Markov decision process. We finally obtain analytical results concerning the structure of the optimal policy.

2.3.1 Problem Setting

We consider an on-demand service platform that serves two customer classes over an infinite horizon. The customer classes, indexed by i, have distinct willingness to wait and to pay, and are named express ($i = 1$) and regular customers ($i = 2$). We suppose that the platform has already determined a menu of prices, and the customers have chosen their corresponding service option. The platform focuses on deriving an optimal dynamic policy. At each arrival, the platform determines whether to admit the customer (admission control). This is done in order to serve a large number of customers while avoiding their abandonment. At any time, the platform also controls the allocation of service providers. As will be clearer subsequently, this essentially boils down to reserving service providers for future express customers. For this reason, we refer to it as reservation control.

Fig. 2.1 illustrates the problem and Table 2.1 gives its notations. As shown in Fig. 2.1, customers of each class arrive following a Poisson process with rate λ_i. Upon arrival, the platform decides whether to admit a class i customer to the system or to reject her, incurring a rejection cost r_i. A rejected customer

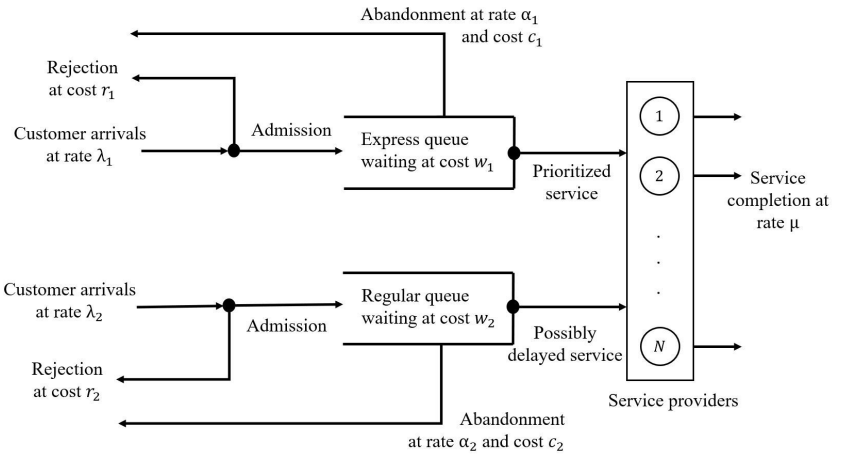


Figure 2.1: Model description.

λ_i	Arrival rate of class i customers.
α_i	Abandonment rate of class i customers.
N	Number of service providers.
μ	Service rate.
ρ	Total workload.
r_i	Rejection cost of class i customers.
w_i	Waiting cost of class i customers.
c_i	Abandonment cost of class i customers.

Table 2.1: Parameters overview.

is supposed to opt for other transportation means (e.g., the subway), and does not retry to enter the system. An admitted customer joins the queue dedicated to her class, waiting to be served by a service provider. Once she is waiting, the admitted customer cannot be rejected anymore. After admitting a customer, the platform must choose whether to allocate a service provider to this customer. If some service providers are available, one of them is automatically allocated to a waiting express customer, who has prioritized service. On the opposite, a service provider can be allocated to a waiting regular customer only if no express customer is waiting (we provide rigorous proof for this strict priority rule in Corollary 2.2, Section 2.3.3). Furthermore, the platform may decide to reserve available service providers for future arrivals of express customers. In that case, the service of a regular customer is delayed, although some service providers are available. To model customer impatience, we suppose that the waiting customers of each class have limited patience, and abandon after exponentially distributed maximum waiting times with mean $1/\alpha_i$. Express customers are on average more impatient than regular customers, so that $\alpha_1 \geq \alpha_2$.

Every time a class i customer abandons, the platform endures an abandonment cost c_i. In addition, for each class i customer waiting, the platform incurs a waiting cost w_i per unit of time. To serve the admitted customers, N service providers operate on the platform. The providers complete their service after an exponentially distributed service time with mean $1/\mu$. The total workload of the system is denoted by $\rho = (\lambda_1 + \lambda_2)/N\mu$.

We formulate the problem as a cost minimization rather than a profit maximization, which avoids rewards and costs of opposite signs. This formulation simplifies the derivation of analytical results (Section 2.3.3) and the interpretation of numerical results (Section 2.4). The marginal profit earned for serving a class i customer is implicitly included in both the rejection cost r_i and the abandonment cost c_i. Therefore, if a customer is rejected, the platform endures a rejection cost which reflects a loss of brand image, and a loss of profit margin. Similarly, if an admitted customer ends up abandoning, the platform endures an abandonment cost that consists of a loss of brand image, and a loss of profit margin. Because the profit margin loss is higher for express customers, their rejection and abandonment costs are always superior to those of regular customers (i.e., $r_1 \geq r_2$ and $c_1 \geq c_2$). Besides, the loss of brand image is more significant with customer abandonment than with customer rejection. This is because a customer abandoning has waited for nothing, while she may have saved time if the platform had rejected her upon arrival. Consequently, the abandonment costs are higher than the rejection costs for both customer classes (i.e., $c_i \geq r_i$, for all i). Lastly, express customers are typically less willing to wait than regular customers. It follows that the waiting costs of express customers are higher than the ones of regular customers (i.e., $w_1 \geq w_2$).

Three assumptions underlie the setting introduced above. First, we assume for simplicity that service providers operate from a single location, returning to the location once the service is completed. Second, we suppose that the platform has complete control over the service providers. This means that the service providers never refuse to serve customers. Third, we assume that each customer class pays the fixed (and not dynamic) price that has been decided by the platform in advance. We believe that this setting still captures the main trade-offs related to waiting time differentiation on on-demand service. Moreover, these three assumptions are sometimes close to reality. Concerning the first assumption, service providers on food-delivery platforms tend to offer services that occur from a specific location (e.g., the commercial district) to other areas (e.g., the residential districts). The setting also fits the case of drivers picking up travelers at an airport and coming back. Concerning the second assumption, Uber, for instance, requires drivers to maintain a high request acceptance rate (Rosenblat, 2016). Concerning the third assumption, some on-demand service platforms use time-based pricing because customers can find dynamic pricing unfair (Bai et al., 2019, Haws & Bearden, 2006). With time-based pricing, prices are fixed in advance for specific periods. This setting fits well with the assumption that prices for the two customer classes are predetermined by the platform.

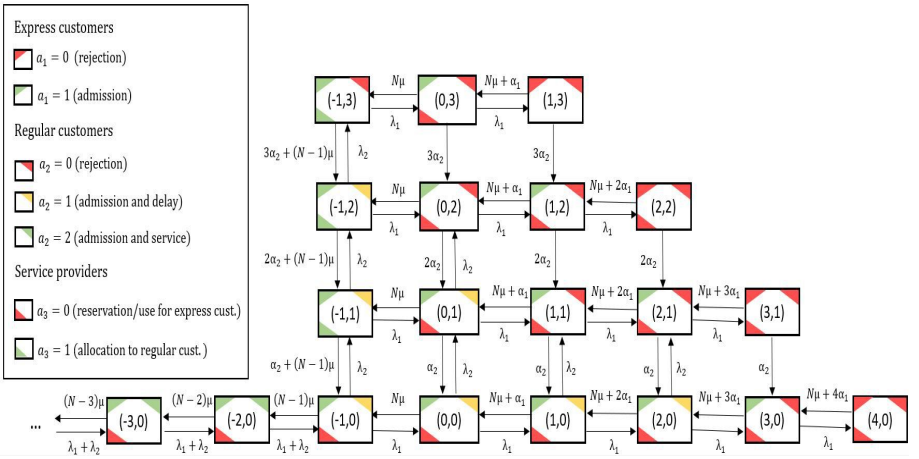


Figure 2.2: Continuous-time Markov chain model resulting from an illustrative policy that controls customer admission and service provider reservation.

2.3.2 MDP Formulation

Since the times between arrivals, services and abandonment events are exponentially distributed, we model the problem discussed above as a continuous-time Markov decision process (MDP). In this section, we first elaborate on the dynamics of our MDP, presenting the states, decisions, policies, and transitions. We then define the resulting objective function and optimality equations.

System Dynamics

States. The state diagram of our MDP is displayed in Fig. 2.2. The system state at time t is represented as a two-dimensional vector $\big(x(t),\, y(t)\big)$ with a state space $\Omega = \{(x,\, y) : -N \leq x,\, 0 \leq y\}$ for a platform managing N service providers. The component $x(t)$ expresses the number of service providers available or express customers waiting. It takes a negative value if service providers are available and takes a positive value if express customers are waiting. The component $y(t)$ denotes the number of regular customers waiting. We use this compact state space because available service providers are always allocated in strict priority to admitted express customers. This means that there cannot be both available service providers and express customers waiting. Furthermore, given that there is a constant number of providers with the same service rate, the number of providers in service can be deduced from the number of available service providers. For this reason, the number of providers in service is not included in the system state.

Decisions. Associated with each state (see the colored triangles in Fig. 2.2), decisions are made concerning the customer admission and service provider

reservation. Three components characterize the decision space:

$$\mathcal{A} = \big\{(a_1, a_2, a_3) : \mathrm{a}_1 \in \{0, 1\},\ \mathrm{a}_2 \in \{0, 1, 2\},\ \mathrm{a}_3 \in \{0, 1\}\big\}. \tag{2.1}$$

The first decision component concerns whether to reject ($a_1 = 0$) or admit ($a_1 = 1$) the incoming express customer. The second component refers to whether to reject ($a_2 = 0$), admit but delay ($a_2 = 1$), or admit and serve immediately ($a_2 = 2$) the next regular customer. Note that regular customers can be admitted but delayed for two reasons: either there is no available service provider, or the available service providers are reserved for express customers. The third decision component concerns a service provider who completed her service. There are two possible decisions. The first possible decision is to allocate her to a regular customer waiting for service ($a_3 = 1$). The second possible decision ($a_3 = 0$) refers to all the other cases: an express customer is waiting and the provider picks her up directly; a regular customer is waiting, but the service provider is reserved for future arrivals of express customers; the service provider waits as there is no customer waiting.

Policies. A policy π is a rule that determines the set of decisions taken in each state of the system. It is said to be optimal if it minimizes the total (abandonment, rejection, and waiting) costs incurred by the platform. Since our model satisfies the Markov property and is evaluated with stationary parameters, we can restrict our attention to stationary deterministic policies (see Puterman, 2014, Chapter 11) with decision epochs occurring at state transitions (see below). Consequently, the unique decision at state (x, y), under a policy π, can be represented by the decision vector $a^\pi(x, y) = (a_1, a_2, a_3)$.

State transitions. Five distinct events lead to state transitions: the arrival of an express or of a regular customer, the service completion of a service provider, and the abandonment of an express or of a regular customer. Then, the current state and the decision prescribed by the policy π determine the state transition that corresponds to each event.

The various state transitions can be seen in Fig. 2.2. If an incoming express customer is rejected ($a_1 = 0$), her arrival generates a transition to the state itself. This transition is omitted in Fig. 2.2 for sake of clarity (e.g., the transition $(4, 0) \to (4, 0)$). If an incoming express customer is admitted ($a_1 = 1$), it results in a decrease of the number of available service providers (e.g., the transition $(-2, 0) \to (-1, 0)$ in Fig. 2.2), until the moment when there is no more service provider available. Then, express customers start waiting (e.g., $(0, 0) \to (1, 0)$). Similarly, an incoming regular customer who is rejected ($a_2 = 0$) induces a self-transition (e.g., $(1, 1) \to (1, 1)$); an incoming regular customer who is admitted and delayed ($a_2 = 1$) waits in the queue (e.g., $(-1, 0) \to (-1, 1)$); and an incoming regular customer who is admitted and served ($a_2 = 2$) results in a decrease of the number of service providers available (e.g., $(-2, 0) \to (-1, 0)$). If a service provider who completes her service is not allocated to a regular customer ($a_3 = 0$), then she either serves one of the express customers waiting (e.g., $(1, 1) \to (0, 1)$), or is reserved for

future express customers (e.g., $(0, 1) \to (-1, 1)$). If the service provider is allocated to a regular customer ($a_3 = 1$), then one waiting regular customer leaves the system with the service provider (e.g., $(-1, 2) \to (-1, 1)$). When customers are waiting, one of them may abandon. This reduces the queue length of express or regular customers (e.g., $(1, 0) \to (0, 0)$ or $(0, 1) \to (0, 0)$ respectively).

Optimality Equations

Based on the system dynamics described above, the goal is to find an optimal policy π^* that minimizes the total expected cost. Let $X^\pi(t)$ and $Y^\pi(t)$ denote respectively the random number of express and regular customers waiting at time t following policy π. Additionally, let $N_i^\pi(t)$ be the total number of class i customers rejected over some time interval $[0, t]$ following a policy π and let $Q_i^\pi(t)$ be the total number of class i customers who abandon over the same time interval, still following policy π. Then, the optimal policy is defined as follows:

$$\pi^* = \arg\min_{\pi} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\Bigg[\int_0^T \big[w_1 X^\pi(t) + w_2 Y^\pi(t)\big]\,dt + \sum_{i=1}^{2} \big[r_i N_i^\pi(T) + c_i Q_i^\pi(T)\big]\Bigg]. \tag{2.2}$$

To determine the optimal policy, we derive a uniformized, equivalent, version of our MDP, where the time until the next transition becomes independent from the state and the decision (see Lippman, 1975). Given that the transition rate may increase indefinitely with the abandonment rates of the waiting customers, we need to set maximum queue sizes M_1 and M_2 for express and regular customers in order to apply uniformization. These maximum queue sizes can be chosen sufficiently large so that the states corresponding to the maximum queue sizes have negligible probabilities of being visited. Accordingly, the maximum transition rate is denoted by $\beta = \lambda_1 + \lambda_2 + N\mu + M_1\alpha_1 + M_2\alpha_2$. Without loss of generality, β is scaled to 1 to simplify the analysis. Following Puterman (2014, Section 8.4), it can then be shown that the optimal average value function v^* (with $v^* \equiv v^{\pi^*}$) satisfies the following optimality equations for all $(x, y) \in \Omega$:

$$\begin{aligned} v^*(x, y) + g &= T[v^*(x, y)] \\ &= w_1 x^+ + w_2 y + \lambda_1\, T_1[v^*(x, y)] + \lambda_2\, T_2[v^*(x, y)] + \mu\, T_3[v^*(x, y)] \\ &\quad + \alpha_1\, T_4[v^*(x, y)] + \alpha_2\, T_5[v^*(x, y)], \end{aligned} \tag{2.3}$$

where g is the steady-state average cost between transitions, T is the usual value iteration operator, and $x^+ = \max(0, x)$. The operator T is formulated as a combination of operators that represent possible event occurrences, as proposed by Koole (1998). Below, each operator in Eq. (2.3) is defined in mathematical

terms, and explained in the next paragraph.

$$T_1[v(x,\,y)] = \begin{cases} \min\left\{r_1 + v(x,\,y),\, v(x+1,\,y)\right\} & \text{if } x < M_1, \quad (2.4a) \\ r_1 + v(x,\,y) & \text{if } x = M_1, \quad (2.4b) \end{cases}$$

$$T_2[v(x,\,y)] =$$

$$\begin{cases} \min\left\{r_2 + v(x,\,y),\, v(x,\,y+1),\, v(x+1,\,y)\right\} & \text{if } x < 0,\, y < M_2, \quad (2.5a) \\ \min\left\{r_2 + v(x,\,y),\, v(x,\,y+1)\right\} & \text{if } x \geq 0,\, y < M_2, \quad (2.5b) \\ r_2 + v(x,\,y) & \text{if } y = M_2, \quad (2.5c) \end{cases}$$

$$T_3[v(x,\,y)] =$$

$$\begin{cases} (N+x) \cdot \min\left\{v(x,\,y-1), v(x-1,\,y)\right\} - x \cdot v(x,\,y) & \text{if } x \leq 0,\, y > 0, \quad (2.6a) \\ (N+x) \cdot v(x-1,\,y) - x \cdot v(x,\,y) & \text{if } x \leq 0,\, y = 0, \quad (2.6b) \\ N \cdot v(x-1,\,y) & \text{if } x > 0, \quad (2.6c) \end{cases}$$

$$T_4[v(x,\,y)] = \begin{cases} x \cdot \left[c_1 + v(x-1,\,y)\right] + (M_1 - x) \cdot v(x,\,y) & \text{if } x > 0, \quad (2.7a) \\ M_1 \cdot v(x,\,y) & \text{if } x \leq 0, \quad (2.7b) \end{cases}$$

$$T_5[v(x,\,y)] = \begin{cases} y \cdot \left[c_2 + v(x,\,y-1)\right] + (M_2 - y) \cdot v(x,\,y) & \text{if } y > 0, \quad (2.8a) \\ M_2 \cdot v(x,\,y) & \text{if } y = 0. \quad (2.8b) \end{cases}$$

The operator T_1 is associated with the event of an express customer arrival, and represents the platform's choice of whether to reject or admit the customer. The operator T_2 corresponds to the arrival of a regular customer. When providers are available ($x < 0$), the platform has the choice between rejecting, admitting, and delaying, and admitting and serving the incoming regular customer. When no provider is available ($x \geq 0$), the decision is only between rejecting and admitting (but delaying) them. The operator T_3 is associated with a service completion. When only regular customers are waiting ($x \leq 0,\, y > 0$), the platform decides between serving a regular customer waiting and keeping the provider who completed her service available for the potential arrival of an express customer (at least until the next event). When no express or regular customers are waiting ($x \leq 0,\, y = 0$), the service provider simply stays available. When express customers are waiting ($x > 0$), the arriving service provider is directly allocated to one of them. In Eq. (2.6a) and Eq. (2.6b), the first term indicates that one of the $N + x$ busy providers (remember that x is nonpositive in this case) completed the service. The second term refers to a fictive service completion and is the result of uniformization. The operators T_4 and T_5 correspond to express and regular customer abandonment. In Eq. (2.7a) and (2.8a), the first term accounts for the abandonment of one of the customers waiting while the second term represents fictive abandonment. If no customer is waiting (i.e., $x \leq 0$, or $y = 0$), fictive abandonment of customers may still occur due to uniformization.

2.3.3 Structure of the Optimal Policy

In this section, we investigate the structure of the optimal policy found by the MDP presented in the previous section. In particular, we prove that the optimal policy is characterized by two admission thresholds, and we show how these thresholds are impacted by various system parameters. These theoretical results may be useful in practice, as they ease the interpretation of the optimal policy, and possibly its computation through faster algorithms (see, e.g., Puterman, 2014, Section 4.7).

To prove the existence of state-dependent admission thresholds, we exploit the specific structure of v_n, the nth iteration of the value function in the value iteration algorithm. Lemma 2.1 states that the optimal average value function v^* $\big($cf. Eq. (2.3)$\big)$ can be made supermodular and Schur-convex (cf. Koole, 2007) for all $(x, y) \in \Omega$ with $x > 0$.

Lemma 2.1. *Let $\mathcal{F}$ be the set of functions f on the state space Ω that satisfy the following properties for all $(x,\, y) \in \Omega$ such that $x > 0$:*

(i) $f\,(x+1,\, y) + f\,(x,\, y+1) \leq f\,(x+1,\, y+1) + f\,(x,\, y)$ *(supermodularity),*

(ii) $f\,(x,\, y+1) \leq f\,(x+1,\, y)$ *(Schur convexity).*

If $v_n \in \mathcal{F}$, then $T[v_n] \in \mathcal{F}$ and $v^* \in \mathcal{F}$, with T and v^* as defined in Eq. (2.3).

The supermodular property (i) of Lemma 2.1 says that the incremental cost of one more waiting express (regular) customer grows with the number of waiting regular (express) customers. This is because an additional express (regular) increases congestion in the system, and thus the probability of regular (express) customers abandoning at cost c_2 (c_1). The Schur-convex property (ii) tells us that the incremental cost of an additional express customer in the system is higher than the incremental cost of an additional regular customer. This second property is a direct consequence of the fact that express customers have a higher abandonment cost, waiting cost, and abandonment rate than regular customers (i.e., $c_1 \geq c_2$, $w_1 \geq w_2$, and $\alpha_1 \geq \alpha_2$).

The proof of Lemma 2.1 is given in Appendix 2.A.1. It firstly demonstrates that each operator T_i, with $i = 1, 2, 3, 4, 5$, preserves the two properties ($T_i[v_n] \in \mathcal{F}$). Because T is a linear combination of these operators, T preserves the properties as well ($T[v_n] \in \mathcal{F}$). By induction over the value iteration algorithm, letting the initial value function v_0 satisfy the two properties, $\big($e.g., initializing $v_0(x,\, y) = 0,\ \forall\,(x,\, y)\big)$, the same property is shown for any v_n ($v_n \in \mathcal{F}$). It follows that the optimal value function v^* stays supermodular and Schur-convex (as $v^*(x,\, y) = \lim_{n\to\infty} v_n(x, y) \in \mathcal{F}$).

In addition to supermodularity and Schur-convexity, other properties regarding the structure of the optimal value function can be demonstrated (see Appendix 2.A.2). Although not exploited in this chapter, they can potentially help speed up computations (see, e.g., Papadaki and Powell, 2002).

As a corollary of Lemma 2.1, it can be shown that a service provider is always allocated in strict priority to an express customer.

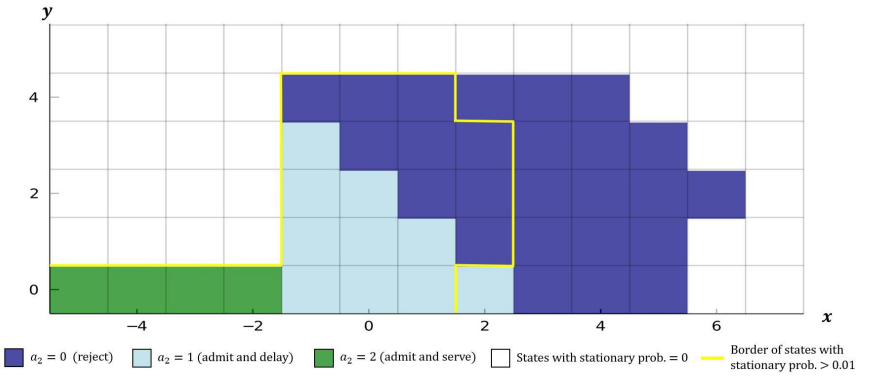


Figure 2.3: The optimal policy for regular customers, with $\lambda_1 = 0.12$, $\lambda_2 = 0.18$, $\alpha_1 = 0.045$, $\alpha_2 = 0.011$, $N = 50$, $\mu = 0.006$, $r_1 = 15$, $r_2 = 7.5$, $w_1 = 0.06$, $w_2 = 0.015$, $c_1 = 18$, and $c_2 = 9$.

Corollary 2.2. *Assuming customer maximum waiting times that are exponentially distributed, it is always optimal to allocate a service provider to a waiting express customer rather than to a waiting regular customer.*

The proof of Corollary 2.2 can be found in Appendix 2.A.3. In short, it is always better to have one less express customer waiting than one less regular customer waiting because the exponential distribution is memoryless, and because the optimal value function is Schur-convex.

Furthermore, since the optimal value function v^* can be made partially supermodular, the optimal policy can be characterized by two admission thresholds as stated in Proposition 2.3 (see Appendix 2.A.4 for a proof).

Proposition 2.3. *In the state $(x, y) \in \Omega$ with $x > 0$, the optimal customer admission decisions $a_1^*(x, y)$ and $a_2^*(x, y)$ (with $a_i^* \equiv a_i^{\pi^*}$) can be determined using the thresholds $t_1(x)$ and $t_2(y)$ such that*

$$a_1^*(x, y) = \begin{cases} 0 & \text{if } y \geq t_1(x) \\ 1 & \text{otherwise,} \end{cases} \quad \text{and} \quad a_2^*(x, y) = \begin{cases} 0 & \text{if } x \geq t_2(y) \\ 1 \text{ or } 2 & \text{otherwise.} \end{cases}$$

The interpretation of Proposition 2.3 is the following. When a given number of customers is already waiting in a queue (express or regular), an incoming customer of this class is rejected if and only if the number of customers waiting in the *opposite* queue exceeds some threshold. For instance, an incoming express customer is admitted if and only if the number of regular customers waiting y is below the threshold $t_1(x)$, which is a function of the number of express customers waiting x. This is referred to as a "switching-curve policy" (see e.g., Koole, 2007).

Fig. 2.3 illustrates the optimal switching-curve admission policy for regular customers. For example, if already one regular customer is waiting, a regular

customer is rejected as soon as the number of express customers already waiting is superior or equal to two $\big($i.e., $t_2(1) = 2\big)$. Moreover, we observe in Fig. 2.3 that the admission threshold of regular customers $t_2(y)$ decreases as a function of the number of regular customers waiting. With respect to service provider reservation, the existence of a third threshold t_3 is also observed. If the number of available providers is strictly superior to that threshold (i.e., $x < t_3$ where $t_3 = -1$), then an incoming regular customer is admitted and directly served. Conversely, if the number of providers is inferior or equal to the threshold (i.e., $x \geq t_3$), then an incoming regular customer is admitted but delayed.

Remark 1. Note that the admission thresholds are only valid for the states with waiting express customers ($x > 0$). The particular definition of our problem, which allows reserving service providers, results in operators T_2 and T_3 that do not propagate supermodularity when $x \leq 0$, similar to Koole (1998). Besides, in Proposition 2.3, the supermodular property is not sufficient to ensure that a reservation threshold t_3 exists. To do this, we would need to prove that $v^*(x+1,\, y) - v^*(x,\, y+1) \leq v^*(x+2,\, y) - v^*(x+1,\, y+1)$ when $x \leq 0$. Again, the definition of our problem, characterized by the operators T_2, T_3, prevents us from deriving that property.

Remark 2. As will be shown numerically in Section 2.4, the system parameters influence the admission thresholds and, more generally, the optimal policy. In Appendix 2.A.5, we provide some theoretical results about the impact of the abandonment costs, waiting costs, and abandonment rates on the admission thresholds.

Remark 3. While the optimal policy aims to minimize the average cost over an infinite horizon, all the results presented in this section extend to other cost criteria, and time horizons. For instance, over a finite time horizon or a discounted cost criterion, admission thresholds are guaranteed to exist as the value iteration operator $T[v_n]$ preserves the supermodular properties of v_n at each iteration (i.e., decision epoch), as stated in Lemma 2.1.

2.4 Numerical Experiments

In this section, we explore numerically how the platform and the users (customers as well as service providers) can benefit from a policy that controls both customer admission and service provider reservation. To this aim, we analyze the performance of the optimal policy found by our model in comparison with simpler policies. Section 2.4.1 presents the experimental setting. Section 2.4.2 examines the benefits to the platform in terms of costs. Sections 2.4.3 and 2.4.4 concentrate on the benefits for the customers and service providers, notably in terms of waiting times and utilization rates.

2.4.1 Experimental Setting

The default parameter values of our numerical study are given in Table 2.2. In the experiments, the effects of specific parameters are studied while keeping the other parameters to their default values. In particular, many of the insights will be illustrated by varying the total workload of the system ρ.

The main objective of our numerical study is to show when controlling customer admission and provider reservation is beneficial. To this end, we compare the optimal policy found by our model with three simpler policies. These policies have been chosen purposely to isolate the two control levers: admission and reservation control. They can roughly be described as follows: the simple priority policy (SP) neither controls customer admission nor service provider reservation; the reservation policy (RP) optimizes the provider reservation but does not manage the customer admission; the admission policy (AP) optimizes the customer admission but does not influence the provider reservation. To emphasize that the optimal policy of our initial model controls both the customer admission and service provider reservation, we will refer to it as the admission and reservation policy (ARP) for the rest of the chapter. Next, let us give a more thorough description of each of the three simpler policies.

The *simple priority policy* (SP) never rejects customers, and available service providers are always allocated to waiting customers. However, express customers still have priority. When both express and regular customers are waiting, a service provider completing her service serves one of the express customers.

When there are regular customers but no express customers waiting, the *reservation policy* (RP) can reserve service providers for potential arrivals of express customers, but it never rejects a customer. To compute the reservation policy, the operators T_1 and T_2 are replaced in Eq. (2.3) respectively by the operators T_6 and T_7, defined as follows:

$$T_6[v(x,\,y)] = \begin{cases} v(x+1,\,y) & \text{if } x < M_1, \quad (2.9a) \\ r_1 + v(x,\,y) & \text{if } x = M_1, \quad (2.9b) \end{cases}$$

$$T_7[v(x,\,y)] = \begin{cases} \min\big\{v(x+1,\,y),\,v(x,\,y+1)\big\} & \text{if } x < 0,\, y < M_2, \quad (2.10a) \\ v(x,\,y+1) & \text{if } x \geq 0,\, y < M_2, \quad (2.10b) \\ r_2 + v(x,\,y) & \text{if } y = M_2. \quad (2.10c) \end{cases}$$

The operator T_6, associated with an express customer arrival, does not allow for rejecting an incoming express customer except if the maximum queue length has been reached (compare Eq. (2.4a) with Eq. (2.9a)). The operator T_7,

Parameters	λ_1	λ_2	α_1	α_2	N	μ	ρ	r_1	r_2	w_1	w_2	c_1	c_2
Default values	60/h	90/h	24/h	6/h	50	3/h	1	15€	7.5€	24€/h	6€/h	18€	9€

Table 2.2: Default parameter values.

associated with a regular customer arrival, always admits a regular customer unless the queue is full (compare Eq. (2.5a) and Eq. (2.5b) with Eq. (2.10a) and Eq. (2.10b)). When service providers are available, the platform still has the choice between serving the customer and delaying her service (at least until the next event), as shown in Eq. (2.10a).

The *admission policy* (AP) can reject customers of either class, but it never reserves service providers for future arrivals of express customers. If both regular and express customers are waiting, then the providers serve the express customers in priority, similar to the simple priority policy (SP). To compute the admission policy, the operators T_8 and T_9 are used in Eq. (2.3) respectively instead of T_2 and T_3, where

$$T_8[v(x,\,y)] = \begin{cases} \min\left\{r_2 + v(x,\,y),\, v(x+1,\,y)\right\} & \text{if } x < 0,\, y < M_2, \quad (2.11a) \\ \min\left\{r_2 + v(x,\,y),\, v(x,\,y+1)\right\} & \text{if } x \geq 0,\, y < M_2, \quad (2.11b) \\ r_2 + v(x,\,y) & \text{if } y = M_2, \quad (2.11c) \end{cases}$$

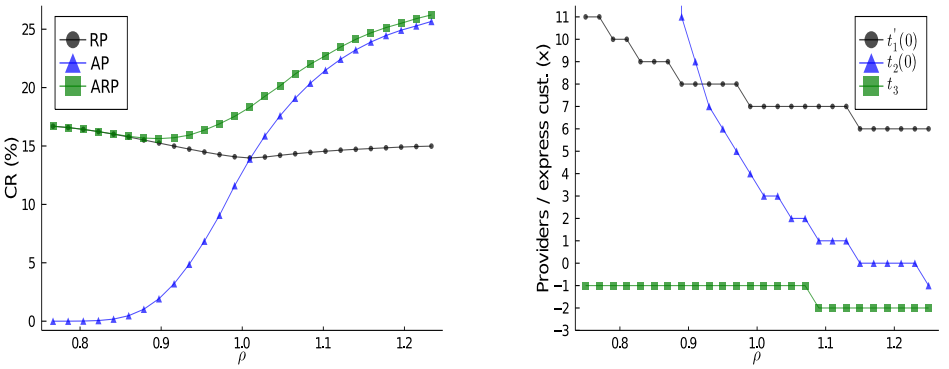
$$T_9[v(x,\,y)] = \begin{cases} (N+x)\cdot v(x,\,y-1) - x \cdot v(x,\,y) & \text{if } x \leq 0,\, y > 0, \quad (2.12a) \\ (N+x)\cdot v(x-1,\,y) - x \cdot v(x,\,y) & \text{if } x \leq 0,\, y = 0, \quad (2.12b) \\ N \cdot v(x-1,\,y). & \text{if } x > 0. \quad (2.12c) \end{cases}$$

The operator T_8, associated with the arrival of a regular customer, does not allow for reserving service providers for future express customers when regular customers are admitted (compare Eq. (2.11a) with Eq. (2.5a)). The operator T_9, associated with the service completion of a provider, does not permit the reservation of a service provider completing her service if regular customers are waiting (compare Eq. (2.12a) with Eq. (2.6a)).

By a direct adaptation of the relative value iteration algorithm (see Puterman, 2014, Section 8.5), these three simpler policies, and the admission and reservation policy (ARP), were computed for various parameter configurations. The algorithm was implemented using the programming language Julia. The experiments were performed on a laptop computer with Intel Core i7 CPU (4x2.30 GHz) and 16 GB RAM. For each instance, the optimal value function and corresponding policy were determined in a few seconds with four-digit accuracy. Various performance measures were then derived from the stationary probabilities of the resulting Markov chain.

2.4.2 Cost-Related Benefits

We first examine the cost-related benefits of admission and reservation control for the platform. To this end, we measure the relative cost reduction (CR) of the reservation policy (RP), the admission policy (AP), and the admission and reservation policy (ARP) with respect to the simple priority policy (SP). In mathematical terms, the relative cost reduction is defined as $CR(\pi) = \frac{g^{SP} - g^{\pi}}{g^{SP}}$, where g^{SP} and g^{π} are the average costs rates of SP and a policy π (i.e., either



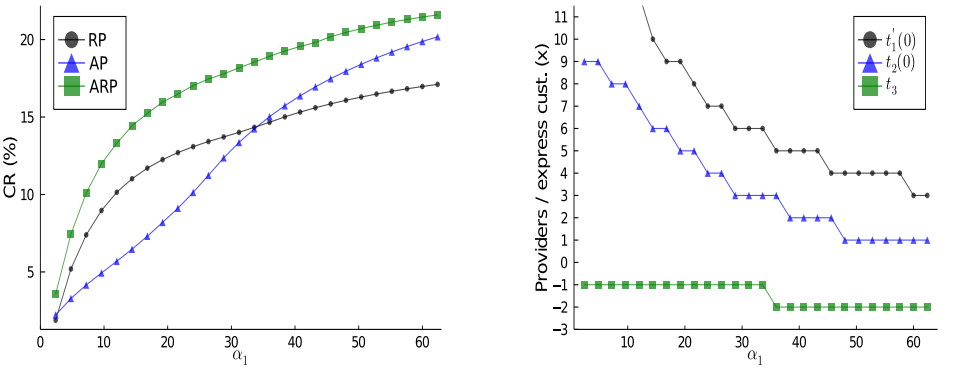
(a) (b)

Figure 2.4: Effect of changes in the total system workload (ρ) on the relative cost reduction (CR) in percentage for RP, AP, and ARP (a), and on the admission thresholds of express $t_1'(0)$, and regular customers $t_2(0)$, and on the reservation threshold t_3 when no regular customer is waiting for ARP (b).

RP, AP, or ARP). This quantity allows us to evaluate the value of each type of control when used alone, as in AP and RP, or when used jointly, as in ARP.

To reflect the decisions of the optimal policy ARP, the admission thresholds and the reservation threshold are also detailed. We facilitate the interpretation of the thresholds by expressing all of them as limits on the number of express customers waiting or providers available. In particular, to represent the admission of an express customer, we consider the threshold $t_1'(y)$ rather than the threshold $t_1(x)$ of Proposition 2.3. With $t_1(x)$, an express customer is rejected if a maximum number of regular customers in the queue is reached, knowing the number of express customers waiting (i.e., if $y < t_1(x)$). With $t_1'(y)$, an express customer is rejected if a maximum number of express customers in the queue is reached, knowing the number of regular customers waiting (i.e., $x < t_1'(y)$). Furthermore, for clarity, we only represent the admission thresholds $t_1'(y)$ and $t_2(y)$ when the number of regular customers waiting is equal to 0 (i.e., when $y = 0$). Similar insights carry on when there is a positive number of regular customers waiting.

Fig. 2.4 shows the evolution of the relative cost reduction for RP, AP, and ARP (a), and the evolution of the thresholds $t_1'(0)$, $t_2(0)$, and t_3 that define ARP (b), when the total workload varies. We observe in Fig. 2.4(a) that the cost reduction caused by RP remains almost constant, around 15% on average. In any case, reserving service providers is relevant to guarantee that admitted express customers are served quickly. On the contrary, the cost reduction of AP grows with the total workload, from 0% to 26%. The rejection of customers becomes gradually useful to maintain the system uncongested and avoid abandonment costs.



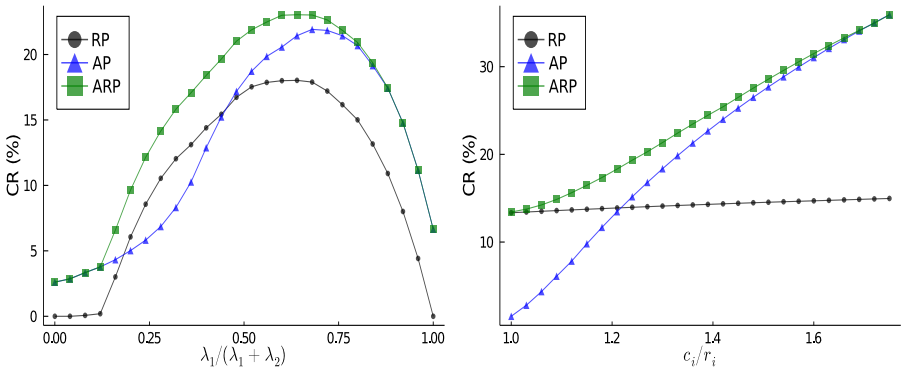
(a) (b)

Figure 2.5: Effect of changes in the two abandonment rates ($\alpha_2 = \alpha_1/4$) on the relative cost reduction (CR) in percentage for AP, RP, and ARP (a), and on the admission thresholds of express $t_1'(0)$, and regular customers $t_2(0)$ when no regular customer is waiting, and on the reservation threshold t_3 for ARP (b).

For any total workload, the optimal policy found by our model, ARP, results in the highest cost reduction by combining admission and reservation control. As seen in Fig. 2.4(b), ARP reserves one service provider for express customers at low workload. ARP even reserves two service providers for high workload. This ensures a fast and accessible service to the express customers. The decreasing admission thresholds (cf. Fig. 2.4(b)) also illustrate that ARP can reject regular and express customers, and does so increasingly as the workload increases (e.g., 1% of the regular customers are rejected at $\rho = 0.9$, 31% of them at $\rho = 1.2$). By combining the capabilities of AP and RP, ARP thus avoids system congestion and customer abandonment.

Interestingly, when the workload is balanced ($0.9 \leq \rho \leq 1.1$, approximately), ARP generates a cost reduction that could never be attained with one type of control alone. For instance, in Fig. 2.4(a), the cost reduction is 5 percentage points higher than both RP and AP when $\rho = 1.02$. To achieve such results, ARP uses admission and reservation control in a complementary way. When the number of express customers waiting in the system is low, ARP admits but delays the service of regular customers, employing reservation control. When many express customers are waiting, ARP rejects customers (especially regular customers, cf. Fig. 2.4(b)), employing admission control. Since the system evolves dynamically between phases of congestion and sufficient capacity at balanced workloads, the use of the two types of control is necessary.

Fig. 2.5 depicts the evolution of the relative cost reduction for RP, AP, and ARP as well as the evolution of the thresholds of ARP, i.e., $t_1'(0)$, $t_2(0)$, and t_3 when both abandonment rates vary proportionally. We see in Fig. 2.5(a) that the control of the provider reservation (RP) becomes increasingly important to



(a) (b)

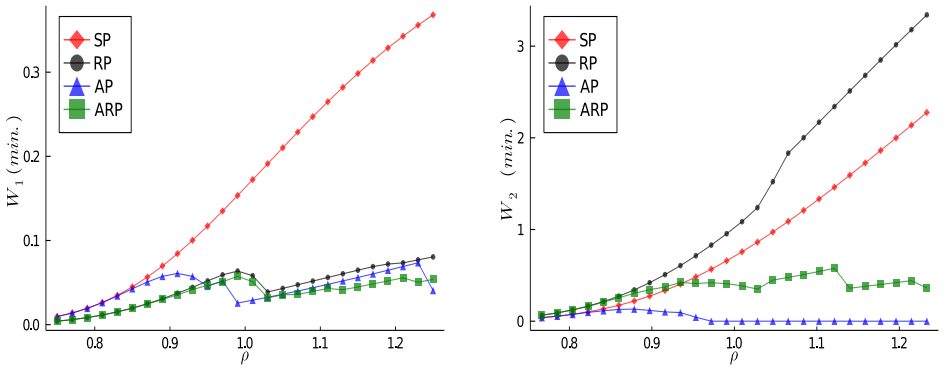
Figure 2.6: Effect of changes in the demand repartition $\big(\lambda_1/(\lambda_1+\lambda_2)$ with $\lambda_1+\lambda_2 = 150\big)$ (a) and the abandonment costs vs. rejection costs (c_i/r_i) (b) on the relative cost reduction (CR) for AP, RP, and ARP in percentage.

avoid waiting times of express customers as the latter get more impatient. At the same time, admission control also becomes more beneficial as the risk of customer abandonment rises. Thus, it gets better to directly reject customers rather than risking costly abandonment.

As discussed previously, ARP generates the highest cost reduction. In particular, for intermediate abandonment rates, ARP reaches a cost reduction up to 5 percentage points higher than the one of AP and RP (and 1.5 percentage points higher than both of them on average for any α_1). Again, ARP benefits from the complementarity between admission and reservation control. It serves a maximum number of regular customers but can still prioritize express customers. At the same time, it limits customer abandonment by gradually rejecting more customers as the abandonment rates grow $\big($cf. Fig. 2.5(b)$\big)$.

For varying demand repartitions, and abandonment costs and rejection costs, Fig. 2.6 confirms the findings presented above. ARP still yields substantial cost reductions in comparison with SP, often more than 20%. ARP also outperforms AP and RP for a significant range of parameter values. In Fig. 2.6(a), it can be noted that admission control is still useful with one customer class and that reservation control is only made relevant by the existence of two customer classes. In Fig. 2.6(b), the cost reduction of RP remains constant because the abandonment costs and the abandonment rates of the two customer classes stay in the same proportion. Thus, RP finds the same balance between the risks of seeing express and regular customers abandon, and the reservation threshold t_3 stays the same for any abandonment costs $\big($similar to Fig. 2.5(b)$\big)$.

In summary, the results presented in this section underline the adaptability of the admission and reservation policy (ARP) to parameter variations. In comparison with the simple priority policy (SP), it can yield a significant cost



(a) (b)

Figure 2.7: Effect of changes in the total workload on the average waiting time of express customers (W_1) (a), and the average waiting time of regular customers (W_2) (b), in minutes for SP, RP, AP, and ARP..

reduction, up to 25%. In addition, our results suggest that, with intermediary parameter values, control over the customer admission and service provider reservation complement each other, generating a cost reduction that could never be attained with one type of control alone (about 5 additional percentage point on the cost reduction). More generally, the results indicate when each type of control should be used, as a function of the system parameters.

2.4.3 Customer-Related Benefits

Besides minimizing costs, the platform's policy might affect the customers. We now analyze the extent to which the customers can benefit from the control of customer admission and service provider reservation, in terms of service quality and service accessibility. The service quality is measured by the average waiting times of the two customer classes, W_i for $i = 1, 2$, and their abandonment probabilities, $P(A \mid i)$ for $i = 1, 2$. The service accessibility is assessed through the rejection probabilities, $P(R \mid i)$ for $i = 1, 2$, and the service probabilities, $P(S \mid i)$ for $i = 1, 2$. These measures are directly computed from the stationary probabilities of the resulting Markov chain, once the optimal policy has been found (for either SP, RP, AP, or ARP). The rejection probability $P(R \mid i)$ and the average waiting time W_i of class i customers are trivial to compute, following the traditional queueing theory. The abandonment probability $P(A \mid i)$ is calculated by extending the formula of Garnett et al. (2002): $P(A \mid i) = \alpha_i W_i / \big[\lambda_i \big(1 - P(R \mid i)\big)\big]$. The service probability $P(S \mid i) = 1 - P(R \mid i) - P(A \mid i)$.

Fig. 2.7 displays the average waiting times of express (W_1) and regular customers (W_2) as a function of the total workload for the four policies. Table 2.3 informs us on the rejection, abandonment, and service probabilities of

the express and regular customers. In Fig. 2.7(a), the reservation policy (RP), the admission policy (AP), and the admission and reservation policy (ARP) constantly maintain the average waiting time of express customers below 0.1 minutes. As shown by the line discontinuities, these policies adapt their decisions to limit customer waiting times, depending on the customer workload. Although the simple priority policy (SP) still prioritizes express customers, it leads to a longer average waiting time for express customers (about 3 times longer than the average waiting time with ARP). The reason is that SP is unable to keep service providers for express customers, either through admission or reservation control. The longer waiting times with SP entail more abandonment of express customers. For instance, in Table 2.3, when $\rho = 1.2$, 17% of the express customers abandon for SP and 3% for ARP.

	SP		RP		AP			ARP		
ρ	$P(\text{A} \mid 1)$	$P(\text{S} \mid 1)$	$P(\text{A} \mid 1)$	$P(\text{S} \mid 1)$	$P(\text{R} \mid 1)$	$P(\text{A} \mid 1)$	$P(\text{S} \mid 1)$	$P(\text{R} \mid 1)$	$P(\text{A} \mid 1)$	$P(\text{S} \mid 1)$
0.9	0.04	0.96	0.02	0.98	0.0	0.03	0.97	0.0	0.02	0.98
1	**0.08**	**0.92**	**0.03**	**0.97**	**0.0**	**0.01**	**0.99**	**0.0**	**0.03**	**0.97**
1.1	0.13	0.87	0.03	0.97	0.0	0.02	0.98	0.0	0.02	0.98
1.2	0.17	0.83	0.04	0.96	0.0	0.03	0.97	0.0	0.03	0.97
	$P(\text{A} \mid 2)$	$P(\text{S} \mid 2)$	$P(\text{A} \mid 2)$	$P(\text{S} \mid 2)$	$P(\text{R} \mid 2)$	$P(\text{A} \mid 2)$	$P(\text{S} \mid 2)$	$P(\text{R} \mid 2)$	$P(\text{A} \mid 2)$	$P(\text{S} \mid 2)$
0.9	0.03	0.97	0.04	0.96	0.03	0.01	0.96	0.01	0.03	0.96
1	**0.07**	**0.93**	**0.10**	**0.90**	**0.17**	**0.0**	**0.83**	**0.08**	**0.04**	**0.88**
1.1	0.12	0.88	0.21	0.79	0.26	0.0	0.74	0.20	0.04	0.76
1.2	0.18	0.82	0.30	0.70	0.34	0.0	0.66	0.31	0.03	0.66

Table 2.3: Effect of total workload changes on the rejection, abandonment, and service probabilities of express and regular customers for the four policies (SP, RP, AP, and ARP). The rejection probabilities of SP and RP are not included because they are always equal to 0 by definition. Results are written in bold for the instance with default parameters.

Three observations can be made from Fig. 2.7(b). First, by looking at ARP and AP, we see that admission control reduces the average waiting time of regular customers. Admission control is employed to reject regular customers when the system becomes congested in order to prevent excessively long waiting times. Admission control thus leads to less abandonment of regular customers as shown in Table 2.3. For example, the abandonment probabilities of AP and ARP are always maintained below 5% for any workload, while they grow up to 18% and 30% for SP and RP, respectively.

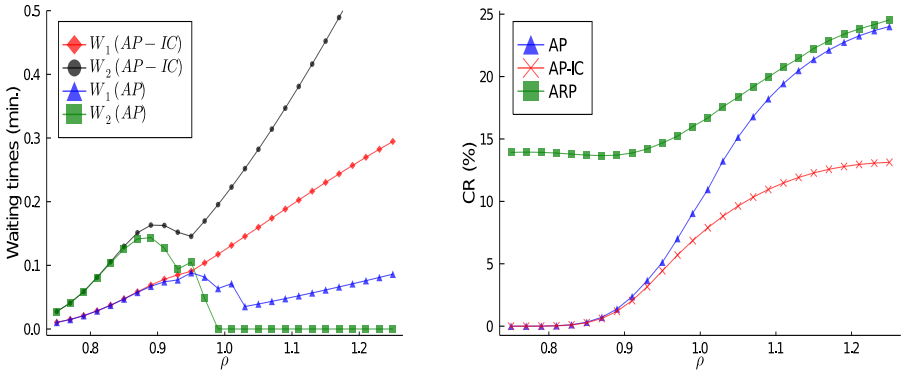
Second, by comparing SP with RP and AP with ARP in Fig. 2.7(b), we observe that reservation control slightly raises the average waiting time of regular customers. This is because reservation control delays the service of regular customers so as to keep service providers available for express customers. More specifically, given that the service of express customers is ensured when regular customers are waiting, reservation control leads to admitting more regular customers. Therefore, more regular customers are served, although a few more

may also abandon. In Table 2.3, when $\rho = 1$, 8% of the regular customers are rejected with ARP, compared to 17% with AP. It follows that the service probability of the regular customers is higher by 5 percentage points with ARP than with AP, while the abandonment probability with ARP also gains 4 percentage points.

Lastly, we notice in Fig. 2.7(b) that AP often generates no waiting time for regular customers when the workload gets high, contrary to ARP. While the model does not capture customer decisions, this may create issues in practice, as express customers can prefer to choose the option intended for regular customers, which is quicker and cheaper than their own option. The explanation for this surprising result is the following. When the capacity is not sufficient to serve the whole demand, AP only admits regular customers, provided that enough service providers are available. This ensures that service providers stay available to serve future express customers. In this way, admission control acts as a substitute for reservation control. However, this also leads AP to reject regular customers even when service providers are available. Therefore, regular customers never wait because they are either rejected or immediately served.

AP thus poses a problem with respect to customer incentives. A simple way to align AP with customer incentives is to artificially inflate the waiting times of regular customers, related to the notion of strategic delay (Afeche, 2013). To do this, we constrain the admission threshold $t_2(y)$ of AP. By doing this, the modified AP, denoted by AP-IC (for "incentive-compatible"), ensures longer average waiting times for regular customers than for express customers, forcing AP to admit even when the number of regular customers is low. For instance, if we set $t_2(y) = 1$ (i.e., regular customers are only rejected when at least one express customer is waiting), the average waiting time of regular customers is increased with AP-IC compared to AP (see Fig. 2.8(a)). This ensures that the waiting times of the two customer classes are compatible with customer incentives. Nevertheless, incentive compatibility comes at a cost. In Fig. 2.8(b), we observe that AP-IC leads to a significantly lower cost reduction (lower by 5 and 13 percentage points than the cost reduction of AP and ARP, respectively). This result seems to indicate that ARP would perform even better comparatively in a setting with incentive-compatible waiting times.

In view of these results, we conclude that ARP has positive effects on customers. On the one hand, it uses admission and reservation control to guarantee a reliable service with short waiting times for express customers. On the other hand, it uses admission control to reject regular customers who cannot be served, and reservation control to delay the service of those who can still be served afterward. For the admitted regular customers, this leads to longer but acceptable waiting times (unlike RP). Unlike with AP, express customers can expect to wait for a shorter time than regular customers with ARP, in accordance with the customer incentives.



(a) (b)

Figure 2.8: Evolution of (a) the average waiting times of express and regular customers for AP-IC and AP, and (b) the cost reduction in percentage for AP-IC, AP, and ARP, as the total workload varies. AP-IC differs from AP by fixing $t_2(y) = 1$.

2.4.4 Provider-Related Benefits

This section evaluates how the optimal policy affects the service providers by looking at their utilization rate $u = \big(\lambda_1 \cdot P(S \mid 1) + \lambda_2 \cdot P(S \mid 2)\big) \,/\, N\mu$. In Fig. 2.9, the utilization rate is plotted as a function of the total workload, and the abandonment rates for the four policies. The curve discontinuities in RP, AP, and ARP arise from the policies being adapted to varying parameters.

As it never rejects any customers and always allocates service providers, SP systematically generates the highest utilization rates. Using only reservation control (RP) leads to a slightly lower utilization rate because service providers are sometimes reserved and left idle to serve express customers. Admission control alone (AP) results in the lowest utilization rate, as it tends to reject additional regular customers to compensate for the lack of reservation control and assure quick services to express customers (see Section 2.4.3). The utilization rate with ARP is located between that of AP and RP, at more than 85% on average on each of the two plots. This is because ARP can reject customers like AP, but it tends to reject fewer of them by reserving service providers like RP.

The slightly lower utilization rate with ARP is potentially compensated by the substantial cost reductions (see Fig. 2.4(a) and Fig. 2.5(a)). The benefits resulting from ARP can be shared among the platform and the service providers, provided that the platform sets an adapted compensation scheme (e.g., setting a commission rate that is proportional to the marginal revenue, as is the case on many platforms). For instance, when the system is balanced (i.e., $\rho = 1$), ARP reduces the costs by 17% in comparison with SP while having a utilization rate that is only 2 percentage points lower. The higher cost reduction of ARP is mainly due to the ability to serve express customers who are more

profitable. (Remember, for instance, that 2% of the express customers abandon with ARP, compared to 16% with SP.) If the earnings of the providers remain proportional to the platform's earnings, then the providers enjoy a revenue rise by serving more express customers. This revenue raise outweighs the minor loss in utilization rates.

2.5 Extensions

In this section, we discuss the performance of the admission and reservation policy (ARP) in two extensions. The goal is to evaluate the performance of ARP, derived by the MDP in Section 2.3, but applied to more general settings. This gives some indication of the robustness and practical relevance of ARP. The first extension considers self-scheduling service providers. The second extension removes the assumption that customer abandonment is exponentially distributed.

2.5.1 Providers with Self-Scheduling Behavior

So far, our model has supposed that the number of service providers who are available on the platform is fixed over time. However, on-demand service platforms have the specific feature that service providers can self-schedule their availability, deciding whether and when to work (see, e.g., Gurvich et al., 2019; Benjaafar & Hu, 2020). Thus, the number of service providers available on the platform is subject to variations. In our case, it matters to evaluate how the admission and reservation policy (ARP) found by the model in Section

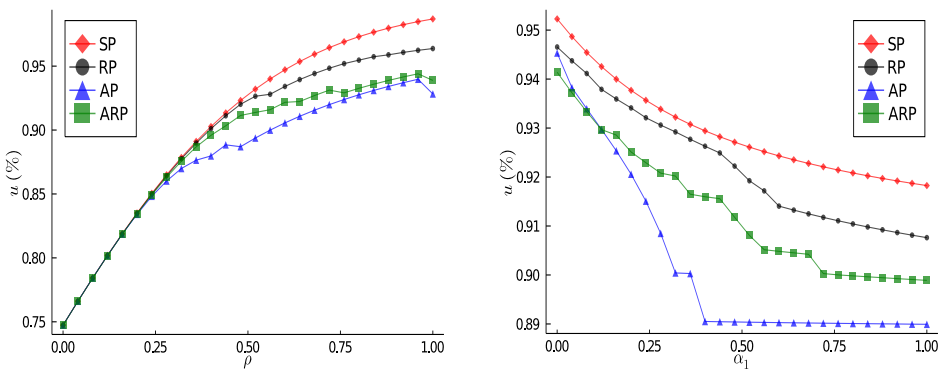


Figure 2.9: Effect of changes in the total workload (a) and the two abandonment rates ($\alpha_2 = \alpha_1/4$) (b) on the utilization rate of service providers (u) for the four policies (SP, RP, AP, ARP).

2.3 will perform in a setting where the service providers exhibit self-scheduling behavior.

To this aim, we develop a server vacation model, close to the one proposed by Azriel et al. (2019). The setting is illustrated in Fig. 2.10. Upon each service completion, the service providers decide with some probability p (referred to as the vacation probability) whether to stay available on the platform or to leave temporarily, going on "vacation". Then, at a specific rate δ (referred to as the return rate), service providers on vacation return to the platform, and become available again to serve customers. Note that this setting can be seen as a generalization of the initial setting presented in Section 2.3. If $p = 1$ or $\delta \to \infty$, then the setting reduces to one where the number of service providers working on the platform is fixed at any time.

An MDP can be designed to compute the optimal policy in this modified setting with self-scheduling service providers. The optimal policy after solving the MDP is called ARP-SS (SS being for self-scheduled service providers). This policy selects a decision (admit vs. reject; serve vs. delay the service) as a function of the number of waiting customers, the number of available service providers, and the number of service providers who are on vacation. In contrast, we call the optimal policy of the initial model ARP-Fixed. ARP-Fixed is found by the MDP of Section 2.3 but can be applied to the modified setting with service provider vacation: it only takes a decision depending on the number of service providers available/customers waiting, and does not differentiate between service providers who are completing service and who are on vacation. The service rate given as an input is adjusted to reflect that $(1-p) \times 100\%$ of the service providers become available after expected times $1/\mu$ (completion of the service) and $p \times 100\%$ of the service providers become available after expected times $(1/\mu + 1/\delta)$ (completion of the service, and return from vacation). Naturally, ARP-Fixed is suboptimal because it does not capture the number

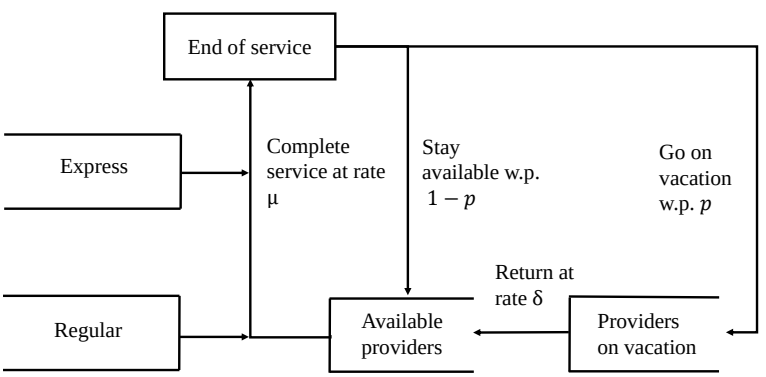


Figure 2.10: Modified setting with self-scheduling service providers.

	δ					
p	0.5	1.0	1.5	2	2.5	3
0.1	2.43%	1.07%	0.19%	0%	0%	0%
0.2	5.28%	2.35%	0.67%	0%	0%	0%
0.3	5.57%	3.00%	0.86%	0%	0.40%	0%
0.4	4.46%	2.78%	0.53%	0.85%	0.10%	0%
0.5	2.28%	4.28%	1.52%	0.31%	0%	0%

Table 2.4: Additional costs with ARP-Fixed in comparison with ARP-SS in percentage, as the vacation probability (p) and the return rate (γ) per hour change.

of service providers on vacation. Our goal is therefore to compare the performance of ARP-Fixed with ARP-SS. Particularly, we evaluate the additional costs that result from applying ARP-Fixed instead of ARP-SS to the modified setting, with self-scheduling providers.

In Table 2.4, we display the additional costs in percentage that are generated by ARP-Fixed in comparison with ARP-SS, for varying vacation probabilities and return rates. The other system parameters are kept to their default values. We observe that ARP-Fixed leads to low additional costs, 1.29% on average. When the return rate is high, the additional costs are negligible. When the return rate is low, the additional costs are due to the fact that a large number of service providers are on vacation, so there is not a sufficient number of service providers to serve the whole demand. For instance, when $p = 0.3$ and $\delta = 0.5$, the service probabilities of express and regular customers $\big($i.e., $P(S \mid 1)$ and $P(S \mid 2)\big)$ are equal to 0.82 and 0.06 respectively. In practice, such situations are unlikely to occur because service providers on vacation are more likely to return if many customers are waiting. In conclusion, our results suggest that the initial policy found in Section 2.3 (ARP-Fixed here) is robust and leads to high performance when there are self-scheduling service providers.

2.5.2 Customers with Erlang-Distributed Abandonment

In Section 2.3, we made the assumption that customers abandon after exponentially distributed maximum waiting times. Even though this assumption is often found to be robust in practice (cf. Brown et al., 2005), situations may exist where customer abandonment follows other distributions. For instance, in the general context of "virtual queues", several studies have shown that customer impatience distribution can be influenced by the amount of available information, priority status, service type, and past experiences (see e.g., Zohar et al., 2002). Furthermore, when impatience is not memoryless, the platform may want to implement waiting-dependent policies, which may elect to serve customers after some time threshold, rather than at the next state transition.

In the following, we assess the robustness of our results when customer abandonment is not exponentially distributed (and thus not memoryless). To

keep a Markovian setting, we consider customers who abandon after maximum waiting times that follow Erlang-k distributions with mean $1/\alpha_i$, $i = 1, 2$. The central idea is that customers of each class abandon after a finite number, k_i, of exponential phases, each occurring with a mean transition time $1/k_i\alpha_i$. These distributions are bell-shaped, characterized by lower coefficients of variation, and can approximate other distributions.

For this modified setting, we develop several MDPs with Erlang-k distributions where $k \leq 5$, i.e., with waiting customers going through at most five phases.[1] According to the number of phases characterizing the maximum waiting time of each customer class, the optimal policies found by these MDPs are called ARP-Erl(k_1, k_2). While these policies account for the time waited by customers, we note that they are often not feasible in practice because the phases of the waiting customers cannot be observed. We can however use these policies as benchmarks. The policy found by the MDP in Section 2.3, which we call ARP-Exp here, can also be applied to these modified settings. The decision is then selected depending on the number of waiting customers, without considering their phase. The policy ARP-Exp is not optimal, but it is probably more feasible, as it does not depend on (unobservable) phases. Next, we evaluate if ARP-Exp performs well in the presence of Erlang-distributed maximum waiting times. We do so by measuring the additional costs that result from applying ARP-Exp to instances where customers of each class abandon following Erlang distributions, in particular Erlang-2 distributions.

Table 2.5(a) shows the additional costs in percentage when ARP-Exp is applied to the setting with Erlang-2 abandonment distributions (thus compared to ARP-Erl(2, 2)), for varying total workload and abandonment rates (the other parameters are kept to default values). We observe that ARP-Exp is close to optimality in every instance, leading to 1.49% additional costs on average. To provide further evidence, we run 16 more instances where the ratio c_1/c_2 (with values from 1 to 2), and the ratios α_1/α_2 and w_1/w_2 (both jointly from 1 to 4) vary. For this other set of instances, ARP-Exp generates 1.54% additional costs on average. To conclude, we observe that the initial policy, derived in Section 2.3 (ARP-Exp here), is robust and performs well when customer abandonment is not exponentially distributed.

Table 2.5(b) shows that regular customers tend to abandon more with ARP-Exp than with ARP-Erl (cf. $P(A \mid 2)$). This is because ARP-Erl has access to more information concerning the precise moment when customers abandon. Therefore, ARP-Erl can prioritize regular customers who are likely to abandon soon (i.e., in the last phase of the Erlang distribution) while ARP-Exp cannot. Because ARP-Erl allocates more service providers to regular customers, more express customers abandon with ARP-Erl than with ARP-Exp (cf. $P(A \mid 1)$). Overall, it appears that, when abandonment is not exponentially distributed, ARP-Exp gives slightly too high priority to express customers. In some extreme

[1] For tractability issues, we focus on Erlang-k with $k \leq 5$. However, we expect the results to hold for Erlang-k with $k > 5$, as in the case of Erlang-distributed service times (cf. Ha, 2000).

	α_1					
ρ	10	20	24	30	40	50
0.8	3.63%	1.97%	3.38%	0.82%	1.18%	1.08%
0.9	0.64%	1.49%	2.53%	1.96%	1.88%	3.14%
1.0	0.20%	1.47%	**1.62%**	1.11%	2.37%	2.44%
1.1	0.12%	2.27%	2.43%	1.12%	1.57%	0.60%
1.2	0.34%	1.24%	0.58%	0.38%	0.75%	0.43%

(a)

	$P(A \mid 1)$		$P(A \mid 2)$	
ρ	ARP-Erl	ARP-Exp	ARP-Erl	ARP-Exp
0.8	0	0	0.005	0.035
0.9	0.012	0.011	0.024	0.097
1.0	**0.023**	**0.019**	**0.034**	**0.094**
1.1	0.019	0.013	0.026	0.096
1.2	0.021	0.01	0.014	0.063

(b)

Table 2.5: Additional costs of ARP-Exp in comparison with ARP-Erl$(2, 2)$ in percentage (a), abandonment probabilities of each customer class with ARP-Exp and ARP-Erl$(2, 2)$ (b), as the total workload and abandonment rates change ($\alpha_2 = \alpha_1/4$). Results are written in bold for the instance with default parameters.

Instance	ARP-Erl$(2, 2)$	ARP-Erl$(3, 3)$	ARP-Erl$(4, 4)$
Additional costs	1.62%	$(6.19 \pm 0.26)\%$	$(5.03 \pm 0.50)\%$
Instance	ARP-Erl$(5, 1)$	ARP-Erl$(1, 5)$	
Additional costs	$(0.51 \pm 0.37)\%$	$(2.21 \pm 0.79)\%$	

Table 2.6: Additional costs of ARP-Exp in comparison with ARP-Erl(k_1, k_2) in percentage, for the default parameters.

cases (0.44% of the time on average for all the $30 + 16$ instances), it may even be more relevant to allocate a service provider to a waiting regular customer rather than to a waiting express customer when both classes of customers are waiting.

Lastly, Table 2.6 informs us about the evolution of the additional costs generated by ARP-Exp in comparison with the optimal policy ARP-Erl(k_1, k_2) in the setting where the maximum waiting times follow Erlang(k_1, k_2) distributions. As more phases are included, the coefficients of variation of the maximum waiting time distributions decrease, and the distributions deviate further from the exponential distribution. The parameters of the model are kept to their default values for all instances.[2] On average, we observe that ARP-Exp results

[2]Because the stationary probabilities cannot be computed for the instances with more than two phases, the obtained policies are simulated in the discretized MDPs over ten million timesteps and the average rewards are computed over 99% confidence intervals.

in 3.08% additional costs for these six instances (1.67% on average over all the instances). Finally, we also note that ARP-Exp requires dramatically less computation time. ARP-Exp can be found in a few seconds while ARP-Erl(k_1, k_2) is deduced after several hours (or several minutes for ARP-Erl(2, 2)).

2.6 Conclusion and Further Research

This chapter studies waiting time differentiation in the context of on-demand service platforms. Two classes of customers arrive on the platform with distinct willingness to pay and to wait. The platform differentiates the customer services by deciding whether to admit or reject customers of each class, and whether to allocate service providers to them or rather reserve service providers for more profitable arrivals.

We formulate this problem as a Markov decision process that combines control over customer admission and over service provider reservation. We then take advantage of the properties of the value function to show analytically that the optimal policy is characterized by two state-dependent admission thresholds. These two thresholds depend on the number of customers waiting in the opposite queue.

In a numerical study, we show that the policy found by our model, which combines admission and reservation control, leads to significant improvement compared to three simpler policies. In particular, the following insights are derived:

(i) Admission control is relevant to manage system congestion while ensuring service to as many customers as possible. It is particularly effective when supply is limited relative to demand, and when both customer classes are impatient with high abandonment and waiting costs.

(ii) Reservation control is relevant to manage the allocation of limited resources between customer classes with distinct willingness to wait and to pay. It is most valuable when express customers are more impatient, or are characterized by higher waiting and abandonment costs than regular customers.

(iii) Admission and reservation control can complement each other to reduce costs across a wide range of parameter values. Especially, at intermediary parameter values, both controls are needed to handle system fluctuations between periods of congestion and sufficient capacity.

(iv) Customers can benefit from shorter waiting times and improved access to services with admission and reservation control. Express customers obtain faster service, while regular customers still experience reasonable delays and enjoy greater access than under admission control alone.

(v) Benefits for service providers arise only if the platform shares with them the gains obtained by admission and reservation control. This can be done through appropriate compensation schemes (e.g., commissions or hourly wages). Although the utilization rates decrease, the earnings of service providers can improve.

(vi) The policy derived in our stylized problem setting performs well in more realistic environments. Specifically, it remains near-optimal when customer abandonment is not exponentially distributed and when providers self-schedule their services. Overall, these findings suggest that the policy is likely to be robust in practice.

Given the recent and evolving nature of on-demand service platforms, our model has been kept fairly general but may be complexified in future research. To generalize the model, a first avenue is to consider arrival and service time distributions that are not exponential and/or time-dependent. As in Section 2.5.2, non-exponential distributions can be constructed through the use of phase-type distributions such as the Coxian distribution. For time-dependent distributions, the transient version of the problem could be investigated rather than assuming a system at steady state. In both cases, it would be useful to examine how it would affect the optimal policy. Another relevant avenue is to extend the model to several locations. On-demand service platforms usually have service providers who transit between multiple areas, causing supply imbalances. The impact of these imbalances on the accessibility and quality of differentiated services is interesting to study. A third avenue is to consider more than two customer classes. Such an extension could allow for studying delay-differentiated services in other environments, such as healthcare and emergency systems. In the context of ride-hailing, the customer classes may also value other factors than time and money, e.g., the vehicle space, fuel consumption, and whether the ride is shared with other customers (as is the case in ride-pooling). Finally, a last avenue is to integrate a pricing decision into our model. In such a case, the arrival rates of the two customer classes would be managed indirectly through dynamic prices in addition to admission decisions. The platform would then determine the price so that each class of customers would prefer the option intended for them. The optimal compensation scheme of the service providers could also be studied in this setting since the compensation usually depends on the price of on-demand service platforms.

Appendix 2.A Proofs and Theoretical Results

2.A.1 Proof of Lemma 1

Proof. We only need to prove that $T_i[v_n] \in \mathcal{F}, \forall i = 1, 2, 3, 4, 5$, assuming that $v_n \in \mathcal{F}$. Then, because $T[v_n]$ is a linear combination of these operators, it follows that $T[v_n] \in \mathcal{F}$. We divide the subsequent proof into five parts. Each part is devoted to demonstrate that $T_i[v_n] \in \mathcal{F}$, provided that $v_n \in \mathcal{F}$ initially. For brevity, we omit the proof for property (ii), which follows the same reasoning as for property (i) and is available upon request. For simplicity, v_n is referred to as v in the rest of the proof.

Operator T_1
We want to show that

$$T_1[v(x+1,\, y)] + T_1[v(x,\, y+1)] \leq T_1[v(x,\, y)] + T_1[v(x+1,\, y+1)].$$

To this end, we next consider each possibility for the minimizers in states $(x,\, y)$ and $(x+1,\, y+1)$, denoted by a_1 and a_2. Thus, we consider four cases.
Case 1. $a_1 = 0,\, a_2 = 0$. The equation

$$\begin{aligned}\min\left\{v(x+2,\, y),\, r_1 + v(x+1,\, y)\right\} + \min\left\{v(x+1,\, y+1),\, r_1 + v(x,\, y+1)\right\} \\ \leq r_1 + v(x,\, y) + r_1 + v(x+1,\, y+1)\end{aligned}$$

is valid, either using the supermodularity assumption or the fact that $r_1 \geq 0$.
Case 2. $a_1 = 1,\, a_2 = 0$. The fact that

$$\begin{aligned}\min\left\{v(x+2,\, y),\, r_1 + v(x+1,\, y)\right\} + \min\left\{v(x+1,\, y+1),\, r_1 + v(x,\, y+1)\right\} \\ \leq v(x+1,\, y) + r_1 + v(x+1,\, y+1)\end{aligned}$$

shows that this second case is valid.
Case 3. $a_1 = 0,\, a_2 = 1$. The equation

$$\begin{aligned}\min\left\{v(x+2,\, y),\, r_1 + v(x+1,\, y)\right\} + \min\left\{v(x+1,\, y+1),\, r_1 + v(x,\, y+1)\right\} \\ \leq v\left(x+2,\, y\right) + r_1 + v\left(x,\, y+1\right) \\ \leq r_1 + v(x,\, y) + v\left(x+2,\, y+1\right),\end{aligned}$$

is valid, using the supermodularity assumption twice.
Case 4. $a_1 = 1,\, a_2 = 1$. Similar to Case 1, the equation

$$\begin{aligned}\min\left\{v(x+2,\, y),\, r_1 + v(x+1,\, y)\right\} + \min\left\{v(x+1,\, y+1),\, r_1 + v(x,\, y+1)\right\} \\ \leq v(x+1,\, y) + v(x+2,\, y+1)\end{aligned}$$

is valid, using the supermodularity assumption.
Hence, in any case, T_1 preserves the supermodular property.

Operator T_2

The proof for this operator follows the same reasoning as the one for operator T_1 when $x \geq 0$. For conciseness, we omit this proof, although available on demand.

Operator T_3

To show that T_3 preserves the supermodular property, we need to show that

$$N \cdot v(x,\, y) + N \cdot v(x-1,\, y+1) \leq N \cdot v(x-1,\, y) + N \cdot v(x,\, y+1),$$

that is,

$$v(x,\, y) + v(x-1,\, y+1) \leq v(x-1,\, y) + v(x,\, y+1).$$

By the initial assumption, we see that the operator T_3 preserves the property.

Operator T_4

Let us prove that T_4 preserves the supermodular property. Proving this is equivalent to showing that

$$\begin{aligned}
&(x+1) \cdot \big[c_1 + v(x,\, y)\big] + (M_1 - x - 1) \cdot v(x+1,\, y) \\
&+ x \cdot \big[v(x-1,\, y+1) + c_1\big] + (M_1 - x) \cdot v(x,\, y+1) \\
&\leq x \cdot \big[c_1 + v(x-1,\, y)\big] + (M_1 - x) \cdot v(x,\, y) \\
&+ (x+1) \cdot \big[c_1 + v(x,\, y+1)\big] + (M_1 - x - 1) \cdot v(x+1,\, y+1).
\end{aligned}$$

This equation can be rewritten as

$$\begin{aligned}
&x \cdot \big[v(x,\, y) + v(x-1, y+1) - v(x+1, y) - v(x,\, y+1)\big] \\
&+ v(x,\, y) + c_1 - v(x,\, y+1) - c_1 \\
&+ (M_1 - x - 1)\big[v(x+1, y) + v(x,\, y+1) - v(x,\, y) - v(x+1, y+1)\big] \\
&+ v(x,\, y+1) - v(x,\, y) \leq 0.
\end{aligned}$$

Since the terms on the second and the fourth lines cancel out, we see that the operator T_4 preserves the supermodular property, by the initial assumption.

Operator T_5

The proof for this operator follows the same reasoning as the proof for T_4 and is omitted for conciseness.

Since we have shown that $T_i[v] \in \mathcal{F}$, for $i = 1, 2, 3, 4, 5$, we know that $T[v] \in \mathcal{F}$. It only remains to demonstrate that $v^* \in \mathcal{F}$. This can be proven by a standard iterative argument. We just set v_0 such that $v_0 \in \mathcal{F}$ (e.g., initializing $v_0(x, y) = 0, \forall (x, y)$). As $v_{n+1} = T[v_n(x,\, y)] - g$, we know that $v_{n+1} \in \mathcal{F}$ for all $n \in \mathbb{N}$. Therefore, it follows that $v^*(x,\, y) = \lim_{n \to \infty} v_n(x,\, y) \in \mathcal{F}$. □

2.A.2 Properties of the Value Function

Lemma 2.1. *Let $\mathcal{G}$ be the set of functions g on Ω that satisfy the following properties:*

(i) $g(x, y) \leq g(x+1, y), \forall (x, y) : x > 0$ *or* $x \leq 0, y = 0$,

(ii) $g(x, y) \leq g(x, y+1), \forall (x, y)$.

If $v_n \in \mathcal{G}$, then $T[v_n] \in \mathcal{G}$. Therefore, the optimal value function v^* also belongs to $\mathcal{G}$, that is, $v^* \in \mathcal{G}$.

Sketch of the proof. The full proof, available upon request, uses the same ideas as employed for Lemma 2.1. We first show that $T_i[v_n] \in \mathcal{G}, \forall i = 1, 2, 3, 4, 5$, assuming that $v_n \in \mathcal{G}$ initially. Then, since $T[v_n]$ is a linear combination of the five operators, we know that $T[v_n] \in \mathcal{G}$ as well. Finally, an iterative argument is again used to conclude that $v^* \in \mathcal{G}$, provided that v_0 has been initialized adequately.

Proof. **Property (i)**

Operator T_1
It can be seen that

$$\begin{aligned}\min\{r_1 + v(x, y), v(x+1, y)\} &\leq v(x+1, y) \\ &\leq \min\{r_1 + v(x+1, y), v(x+2, y)\}\end{aligned}$$

for all $(x, y) \in \Omega$. Hence, T_1 preserves Property (i).

Operator T_2
We consider two cases.
Case 1. $x \leq 0$, $y = 0$. We have

$$\min\{r_2 + v(x, 0), v(x, 1), v(x+1, 0)\} \leq r_2 + v(x, 0) \leq r_2 + v(x+1, 0),$$

$$\min\{r_2 + v(x, 0), v(x, 1), v(x+1, 0)\} \leq v(x, 1) \leq v(x+1, 1),$$

and

$$\min\{r_2 + v(x, 0), v(x, 1), v(x+1, 0)\} \leq v(x+1, 0) \leq v(x+2, 0).$$

It follows that

$$\min\{r_2+v(x,0), v(x,1), v(x+1,0)\} \leq \min\{r_2+v(x+1,0), v(x+1,1), v(x+2,0)\},$$

that is,

$$T_2[v(x, 0)] \leq T_2[v(x, 1)],$$

when $x \leq 0$. This proves that T_2 preserves Property (i) for the first case.
Case 2. $x > 0$. The proof for this case is similar to the proof for T_1, and is therefore omitted.

Operator T_3
We consider two cases.
Case 1. $x \leq 0$, $y = 0$. We must have

$$(N + x) \cdot v(x - 1,\, 0) - x \cdot v(x,\, 0) \leq (N + x + 1) \cdot v(x,\, 0) - (x + 1) \cdot v(x + 1, 0).$$

We can rewrite it as

$$(N+x)\cdot\big[v(x-1,\, 0)-v(x,\, 0)\big]-v(x,\, 0)-(x+1)\cdot\big[v(x,\, 0)-v(x+1,\, 0)\big]+v(x,\, 0) \leq 0.$$

This equation is true by our initial assumption, verifying this case.
Case 2. $x > 0$. It can be seen that

$$N \cdot v(x - 1, y) \leq N \cdot v(x, y)$$

is true by our initial assumption. Hence, T_3 preserves Property (i).

Operator T_4
We again consider two cases.
Case 1. $x < 0$. It can be observed that

$$M_1 \cdot v(x,\, y) \leq M_1 \cdot v(x + 1,\, y),$$

proving Property (i) for this case.
Case 2. $x \geq 0$. It suffices to show that

$$\begin{aligned} x \cdot \big[c_1 + v(x - 1,\, y)\big] + (M_1 - x) \cdot v(x,\, y) \leq \\ (x + 1) \cdot \big[c_1 + v(x,\, y)\big] + (M_1 - x - 1) \cdot v(x + 1,\, y). \end{aligned}$$

We can rewrite the equation above as

$$\begin{aligned} x \cdot \big[v(x - 1,\, y) - v(x,\, y)\big] - c_1 - v(x,\, y) + v(x,\, y) \\ + (M_1 - x - 1) \cdot \big[v(x,\, y) - v(x + 1,\, y)\big] \leq 0. \end{aligned}$$

This equation is true, using the initial assumption and the fact that $c_1 > 0$. Hence, T_4 preserves Property (i).

Operator T_5
We have

$$y\cdot\big[c_2+v(x,\, y-1)\big]+(M_2-y)\cdot v(x,\, y) \leq y\cdot\big[c_2+v(x+1,\, y-1)\big]+(M_2-y)\cdot v(x+1,\, y),$$

by the initial assumption. Hence, the operator T_5 preserves Property (i).

Property (ii)

Operator T_1
Because

$$\min\big\{r_1 + v(x,\, y),\, v(x + 1,\, y)\big\} \leq v(x + 1,\, y) \leq v(x + 1,\, y + 1),$$

and

$$\min\{r_1 + v(x, y),\, v(x+1, y)\} \le r_1 + v(x, y) \le r_1 + v(x, y+1),$$

it follows that

$$\min\{r_1 + v(x, y),\, v(x+1, y)\} \le \min\{r_1 + v(x, y+1),\, v(x+1, y+1)\}.$$

Therefore, the operator T_1 preserves Property (ii).

Operator T_2
We consider two cases.
Case 1. $x < 0$. Since

$$\min\{r_2 + v(x, y),\, v(x, y+1),\, v(x+1, y)\} \le v(x, y+1),$$

and

$$\min\{r_2 + v(x, y),\, v(x, y+1),\, v(x+1, y)\} \le v(x, y+1),$$

it is true that

$$\min\{r_2 + v(x, y),\, v(x, y+1), v(x+1, y)\} \le \min\{r_2 + v(x, y+1),\, v(x, y+2),\, v(x+1, y+1)\},$$

which proves that T_2 preserves Property (ii).

Case 2. $x \ge 0$. The proof for this case is analogous to the proof for T_1 and is therefore omitted.

Operator T_3
We consider three cases.
Case 1. $x \le 0$, $y > 0$. We need to verify that

$$(N + x) \cdot \min\{v(x-1, y), v(x, y-1)\} - x \cdot v(x, y) \le (N+x) \cdot \min\{v(x-1, y+1),\, v(x, y)\} - x \cdot v(x, y+1).$$

Since $v(x-1, y) \le v(x-1, y+1)$ and $v(x, y-1) \le v(x, y)$ by assumption, the equation is verified.
Case 2. $x \le 0$, $y = 0$. We want to show that

$$(N+x)\cdot v(x-1, 0) - x \cdot v(x, 0) \le (N+x)\cdot \min\{v(x-1, 1), v(x, 0)\} - x \cdot v(x, 1).$$

Because $v(x-1, 0) \le v(x-1, 1)$ by the initial assumption and $v(x-1, 0) \le v(x, 0)$ by Property (i), the above equation is true.
Case 3. $x > 0$. It is easy to see that

$$N \cdot v(x-1, y) \le N \cdot v(x-1, y+1)$$

by the initial assumption. This completes the proof for T_3

Operator T_4
We consider two cases.
Case 1. $x < 0$. Because

$$M_1 \cdot v(x,\, y) \leq M_1 \cdot v(x,\, y+1),$$

the first case is true by assumption.
Case 2. $x \geq 0$. It can be seen that

$$x\cdot\big[c_1{+}v(x{-}1,\, y)\big]{+}(M_1{-}x)\cdot v(x,\, y) \leq x\cdot\big[c_1{+}v(x{-}1,\, y{+}1)\big]{+}(M_1{-}x)\cdot v(x,\, y{+}1)$$

by assumption. This proves that T_4 satisfies Property (ii).

Operator T_5
Similar to the proof for T_4 for the first property, it suffices to show that

$$y\cdot\big(c_2{+}v(x,\, y{-}1)\big){+}(M_2{-}y)\cdot v(x,\, y) \leq (y{+}1)\cdot\big(c_2{+}v(x,\, y)\big){+}(M_2{-}y{-}1)\cdot v(x,\, y{+}1).$$

We can rewrite the equation above as

$$\begin{aligned} y \cdot \big(v(x,\, y-1) - v(x,\, y)\big) - c_2 - v(x,\, y) + v(x,\, y) \\ + (M_2 - y - 1) \cdot \big(v(x,\, y) - v(x,\, y+1)\big) \leq 0, \end{aligned}$$

which is true, using the initial assumption and the fact that $c_2 > 0$. This proves that T_5 preserves Property (ii).

Similar to Lemma 1, because $T_i[v] \in \mathcal{G}$, $\forall\, i = 1,\, 2,\, 3,\, 4,\, 5$, it follows that $T[v] \in \mathcal{G}$. We again use a standard iterative argument to demonstrate that $v^* \in \mathcal{G}$. Suppose that $v_0 \in \mathcal{G}$ and $v_n \in \mathcal{G}$ for some n. Since $v_{n+1} = T[v_n(x,\, y)] - g$, it follows that $v_{n+1} \in \mathcal{G}$. Hence, $v^* = \lim_{n\to\infty} v_n \in \mathcal{G}$, which completes the proof. □

2.A.3 Proof of Corollary 1

Proof. Consider the case where a service provider can be allocated to a regular customer, even when at least one express customer is also waiting. Then, the only modification to the optimality equation is that the operator T_3 is redefined as:

$$T_3[v(x,y)] = \begin{cases} (N+x)\cdot \min\big\{v(x-1,y), v(x,y-1)\big\} - x\cdot v(x,y) & \text{if } x \leq 0, y > 0, \\ (N+x)\cdot v(x-1,y) - x\cdot v(x,y) & \text{if } x \leq 0, y = 0, \\ N\cdot \min\big\{v(x-1,y), v(x,y-1)\big\} & \text{if } x > 0, y > 0 \\ N\cdot v(x-1,y) & \text{if } x > 0, y = 0 \end{cases}$$

To prove Corollary 2.2, we need to show that $v^*(x-1,\, y) \leq v^*(x,\, y-1)$ when $x > 0$, which is equivalent to Schur convexity, $v^*(x,\, y+1) \leq v^*(x+1,\, y)$ (cf. Lemma 2.1). Since the operators T_1, T_2, T_4, and T_5 are unchanged, we

know from Lemma 2.1 that they preserve Schur convexity if $x > 0$. All we are left to show is that the modified operator T_3 preserves the property. If $x > 0$ and $y = 0$, we already know that this is true from Lemma 2.1. If $x > 0$ and $y > 0$, we want to show that

$$N \cdot \min\big\{v(x-1,\, y+1), v(x,\, y)\big\} \leq N \cdot \min\big\{v(x,\, y), v(x+1,\, y-1)\big\}.$$

To this end, we consider each possibility for the minimizers in states $(x+1,\, y)$, denoted by a_3:
Case 1. $a_3 = 0$. $N \cdot \min\big\{v(x-1,\, y+1), v(x,\, y)\big\} \leq N \cdot v(x,\, y)$ is true by definition of the minimum operator.
Case 2. $a_3 = 0$. $N \cdot \min\big\{v(x-1,\, y+1), v(x,\, y)\big\} \leq N \cdot v(x+1,\, y-1)$ is true by definition of the minimum operator and using the Schur convexity twice.

Hence, in any case, T_3 preserves Schur convexity, and Corollary 2.2 follows. □

2.A.4 Proof of Proposition 1

Proof. We only prove this proposition for the threshold $t_1(x)$. An analogue reasoning can be applied to $t_2(y)$. The existence of the threshold $t_1(x)$ boils down to prove that if rejection is optimal in $(x,\, y)$ $\big($i.e., $r_1 + v^*(x,\, y) \leq v(x+1,\, y)\big)$, then it is also optimal in state $(x,\, y+1)$ $\big($i.e., $r_1 + v^*(x,\, y+1) \leq v^*(x+1,\, y+1)\big)$. In mathematical terms, this can be stated as follows:

$$r_1 + v^*(x,\, y) \leq v^*(x+1,\, y) \Rightarrow r_1 + v^*(x,\, y+1) \leq v^*(x+1,\, y+1).$$

For this expression to be true, it is sufficient to show that

$$v^*(x+1,\, y) - r_1 - v^*(x,\, y) \leq v^*(x+1,\, y+1) - r_1 - v^*(x,\, y+1),$$

which is equivalent to saying that

$$v^*(x+1,\, y) + v^*(x,\, y+1) \leq v^*(x,\, y) + v^*(x+1,\, y+1).$$

By Lemma 1, we know that this last equation is true when $x > 0$. □

2.A.5 Impact of Parameters on Admission

Proposition 2.2. *In the state $(x,\, y) \in \Omega$ with $x > 0$, the admission thresholds $t_1(x)$ and $t_2(y)$ of the optimal policy satisfy the following properties:*

- *The admission threshold $t_1(x)$ is nonincreasing in c_1, w_1 and α_1, and nondecreasing in c_2 and w_2.*
- *The admission threshold $t_2(y)$ is nonincreasing in c_2, w_2 and α_2 and nondecreasing in c_1 and w_1.*

Proof. We only prove the first statement of this proposition, since the proof of the second statement follows a similar approach. To demonstrate the first statement, we compare the optimal value function of our original Markov decision process, $v^*(x, y)$, with the optimal value function of a Markov decision process, $v^*_{\gamma_i}(x, y)$, where the parameter γ_i ($\gamma_i = c_i, \alpha_i, i = 1, 2$) has been increased by a factor Δ.

In particular, we demonstrate that

$$v^*(x+1,\, y) + v^*_{\gamma_i}(x,\, y) \leq v^*(x,\, y) + v^*_{\gamma_i}(x+1,\, y), \quad \gamma_i = h_1,\, c_1,\, \alpha_1, \quad (2.14)$$

$$v^*(x+1,\, y) + v^*_{\gamma_i}(x,\, y) \geq v^*(x,\, y) + v^*_{\gamma_i}(x+1,\, y), \quad \gamma_i = h_2,\, c_2. \quad (2.15)$$

when $x > 0$. These are sufficient properties to show that the operators $t_1(x)$ and $t_2(y)$ evolve monotonically with $\gamma_i = c_i,\ i = 1, 2$ or $\gamma_i = \alpha_1$. For instance, when $\gamma_i = c_1$, because $v^*(x+1,\, y) + v^*_{c_1}(x,\, y) \leq v^*_{c_1}(x+1,\, y) + v^*(x,\, y)$, it follows that

$$r_1 + v^*(x,\, y) \leq v^*(x+1,\, y) \Rightarrow r_1 + v^*_{c_1}(x,\, y+1) \leq v^*_{c_1}(x+1,\, y+1).$$

This means that the threshold $t_1(x)$ is decreasing with c_1, in the same spirit as in Proposition 1.

Similar to the proof of Lemma 1, we show that $T[v_n]$ preserves Eq. (2.14) and Eq. (2.15), provided that these properties are initially respected by v_n (which we will simply refer to as v thereafter). We divide the proof into two main parts. In the first part, we show that each operator T_i, $i = 1, 2, 3, 4, 5$ maintains Eq. (2.14) and Eq. (2.15) with $\gamma_i = c_i,\ i = 1, 2$. In the second part, we show that the value iteration operator T maintains Eq. (2.14) with $\gamma_i = \alpha_1$. In that second part, we create a fictive operator occurring at a rate Δ in the original Markov decision process to ensure that the two Markov decision processes (i.e., the one with the usual parameter and the one with the parameter γ_i) live in the same time scale, with the same total transition rate, following the approach proposed by Çil et al. (2009).

Part 1. Proof for Eq. (2.14) ($\gamma_i = c_1$) and Eq. (2.15) ($\gamma_i = c_2$).
Operator T_1

Case 1. $\gamma_i = c_1$.

We show that

$$T_1[v(x+1,\, y)] + T_1[v_{c_1}(x,\, y)] \leq T_1[v_{c_1}(x+1,\, y)] + T_1[v(x,\, y)].$$

To this end, we consider each possibility for the minimizers on the right-hand side, denoted by a_1 and a_2 respectively. If $a_1 = a_2$, it is easy to see that T_1 maintains Property 2.14. If $a_1 = 0$ and $a_2 = 1$, then

$$\begin{aligned}\min\left\{r_1 + v(x+1,\, y),\, v(x+2,\, y)\right\} + \min\left\{r_1 + v_{c_1}(x,\, y),\, v_{c_1}(x+1,\, y)\right\} \\ \leq r_1 + v_{c_1}(x+1,\, y) + v(x+1,\, y).\end{aligned}$$

If $a_1 = 1$ and $a_2 = 0$, then

$$\min\{r_1 + v(x+1,\, y),\, v(x+2,\, y)\} + \min\{r_1 + v_{c_1}(x,\, y),\, v_{c_1}(x+1,\, y)\} \\ \leq v_{c_1}(x+2,\, y) + r_1 + v(x,\, y),$$

by using Eq. (2.14) twice. Therefore, T_1 preserves Property 2.14 in any case.
Case 2. $\gamma_i = c_2$.

Following the reasoning of Case 1, it can be shown that T_1 preserves Eq. (2.15).

Operator T_2

We show that $T_2[v(x+1,\, y)] + T_2[v_{c_1}(x,\, y)] \leq T_2[v_{c_1}(x+1,\, y)] + T_2[v(x,\, y)]$. We show this by considering each possibility for the minimizers on the right-hand side, denoted by a_1 and a_2 respectively. If $a_1 = a_2$, then the result follows easily. If $a_1 = 0$ and $a_2 = 1$, then we need to show that

$$\min\{r_2 + v(x+1,\, y),\, v(x+1,\, y+1)\} + \min\{r_2 + v_{c_1}(x,\, y),\, v_{c_1}(x,\, y+1)\} \\ \leq r_2 + v_{c_1}(x+1,\, y) + v(x,\, y+1).$$

Note that the term on the left-hand side is bounded above by $r_2 + v(x+1,\, y) + v_{c_1}(x,\, y+1)$, which is a direct consequence of the definition of the minimizers. Thus, it suffices to show that

$$r_2 + v(x+1,\, y) + v_{c_1}(x,\, y+1) \leq r_2 + v_{c_1}(x+1,\, y) + v(x,\, y+1).$$

By the initial assumptions, this equation is true:

$$v_{c_1}(x,\, y+1) - v(x,\, y+1) \leq v_{c_1}(x,\, y) - v(x,\, y) \leq v_{c_1}(x+1,\, y) - v(x+1,\, y).$$

If $a_1 = 1$ and $a_2 = 0$, then

$$\begin{aligned} \min\{r_2 + v(x+1,\, y),\, v(x+1,\, y+1)\} + \min\{r_2 + v_{c_1}(x,\, y),\, v_{c_1}(x,\, y+1)\} \\ \leq r_2 + v(x+1,\, y) + v_{c_1}(x,\, y+1) \\ \leq v_{c_1}(x+1,\, y+1) + r_2 + v(x,\, y). \end{aligned}$$

The last inequality is true because

$$\begin{aligned} v(x+1,\, y) - v(x,\, y) &\leq v(x+1,\, y+1) - v(x,\, y+1) && \text{(by Proposition 1)} \\ &\leq v_{c_1}(x+1,\, y+1) - v_{c_1}(x,\, y+1) && \text{(by Eq. 2.14).} \end{aligned}$$

Case 2. $\gamma_i = c_2$. Following the reasoning of Case 1, it can be shown that T_2 preserves Eq. (2.15).

Operator T_3

Proving that T_3 preserves Eq. 2.14 and Eq. 2.15 is equivalent to proving that

$$N \cdot v(x,\, y) + N \cdot v_{c_1}(x-1,\, y) \leq N \cdot v(x-1,\, y) + N \cdot v_{c_1}(x,\, y+1),$$

and

$$N \cdot v(x,\, y) + N \cdot v_{c_2}(x-1,\, y) \geq N \cdot v(x-1,\, y) + N \cdot v_{c_2}(x,\, y+1).$$

We see that the operator T_3 preserves Eq. (2.14) and Eq. (2.15) with $\gamma_i = c_1,\, c_2$, by the initial assumption.

Operator T_4

Case 1. $\gamma_i = c_1$.

Assume that Eq. 2.14 is true. Then, it follows that

$$\begin{aligned}
&(x+1) \cdot \big[c_1 + v(x,\, y)\big] + (M_1 - x - 1) \cdot v(x+1,\, y) \\
&+ x \cdot \big[c_1 + \Delta + v_{c_1}(x-1,\, y)\big] + (M_1 - x) \cdot v_{c_1}(x,\, y) \\
&\leq x \cdot \big[c_1 + v(x-1,\, y)\big] + (M_1 - x) \cdot v(x,\, y) \\
&+ (x+1) \cdot \big[c_1 + \Delta + v_{c_1}(x,\, y)\big] + (M_1 - x - 1) \cdot v_{c_1}(x+1,\, y).
\end{aligned}$$

This equation can be rewritten as

$$\begin{aligned}
&x \cdot \big[v(x,\, y) + v_{c_1}(x-1, y) - v(x-1, y) - v_{c_1}(x,\, y)\big] \\
&+ c_1 + v(x,\, y) - c_1 - v_{c_1}(x,\, y) - \Delta \\
&+ (M_1 - x - 1)\big[v(x+1, y) + v_{c_1}(x,\, y) - v(x,\, y) - v_{c_1}(x+1, y)\big] \\
&+ v_{c_1}(x,\, y) - v(x,\, y) \leq 0.
\end{aligned}$$

Since the first and third lines are negative by the initial assumption, and the sum of the second and fourth lines is also negative, we see that Eq. (2.14) is true in this first case.

Case 2. $\gamma_i = c_2$. Following the reasoning of Case 1, it can be shown that T_4 preserves Eq. (2.15) with $\gamma_i = c_2$.

Operator T_5**.** The proof is similar to the proof for T_4 and is omitted for conciseness.

Part 2. Proof for Eq. (2.14) ***($\gamma_i = \alpha_1$).***

To ensure that the initial Markov decision process and the Markov decision process with an increased abandonment rate have the same transition rates, we modify the initial Markov decision process to allow fictive transition at rate Δ. Therefore, we can rewrite Eq. (2.14), with $\gamma_i = \alpha_1$ as follows:

$$\begin{bmatrix}
\lambda_1 \cdot \big(T_1[v(x+1,\, y)] + T_1[v_{\alpha_1}(x,\, y)]\big) \\
+ \lambda_2 \cdot \big(T_2[v(x+1,\, y)] + T_2[v_{\alpha_1}(x,\, y)]\big) \\
+ \mu \cdot \big(T_3[v(x+1,\, y)] + T_3[v_{\alpha_1}(x,\, y)]\big) \\
+ \alpha_1 \cdot \big(T_4[v(x+1,\, y)] + T_4[v_{\alpha_1}(x,\, y)]\big) \\
+ \alpha_2 \cdot \big(T_5[v(x+1,\, y)] + T_5[v_{\alpha_1}(x,\, y)]\big) \\
+ \Delta \cdot \big(v(x+1,\, y) + T_4[v_{\alpha_1}(x,\, y)]\big)
\end{bmatrix}$$

$$\leq \begin{bmatrix} \lambda_1 \cdot \big(T_1[v(x,\, y)] + T_1[v_{\alpha_1}(x+1,\, y)]\big) \\ +\lambda_2 \cdot \big(T_2[v(x,\, y)] + T_2[v_{\alpha_1}(x+1,\, y)]\big) \\ +\mu \cdot \big(T_3[v(x,\, y)] + T_3[v_{\alpha_1}(x+1,\, y)]\big) \\ +\alpha_1 \cdot \big(T_4[v(x,\, y)] + T_4[v_{\alpha_1}(x+1,\, y)]\big) \\ +\alpha_2 \cdot \big(T_5[v(x+1,\, y)] + T_5[v_{\alpha_1}(x,\, y)]\big) \\ +\Delta \cdot \big(v(x,\, y) + T_4[v_{\alpha_1}(x+1,\, y)]\big) \end{bmatrix}$$

Because α_1 is not involved in any of the operators, it can be seen that the first five terms of the equation on each side preserve Eq. (2.14). We are therefore left to show that

$$\Delta \cdot (v(x+1,\, y) - v(x,\, y)) \leq \Delta \cdot (T_4[v_{\alpha_1}(x+1,\, y)]) - T_4[v_{\alpha_1}(x,\, y)]).$$

By Eq. (2.14), we know that

$$v(x+1,\, y) - v(x,\, y) \leq v_{\alpha_1}(x+1,\, y) - v_{\alpha_1}(x,\, y),$$

so it suffices to show that

$$\begin{aligned} v_{\alpha_1}(x+1,\, y) - v_{\alpha_1}(x,\, y) \leq\ & (x+1) \cdot [c_1 + v_{\alpha_1}(x, y)] \\ & + (M_1 - x - 1) \cdot v_{\alpha_1}(x+1,\, y) \\ & - x \cdot [c_1 + v_{\alpha_1}(x-1,\, y)] - (M_1 - x) \cdot v_{\alpha_1}(x,\, y), \end{aligned}$$

which can be rewritten as

$$\begin{aligned} 0 \leq x \cdot \big[v_{\alpha_1}(x, y) - v_{\alpha_1}(x-1, y)\big] & + (M_1 - x - 2) \cdot \big[v_{\alpha_1}(x+1,\, y) - v_{\alpha_1}(x,\, y)\big] \\ & + c_1 + v_{\alpha_1}(x,\, y) - v_{\alpha_1}(x,\, y). \end{aligned}$$

By Lemma 2 and the fact that $c_1 \geq 0$, the above inequality is true, completing the proof. □

3 Pricing, Repositioning and Admission in Ride-Hailing Networks

De Munck, T., Tancrez, J.-S., and Chevalier, P. Reinforcement learning with pretraining for pricing, driver repositioning, and customer admission in ride-hailing networks. CORE Discussion Paper 2025/03 (under revision at European Journal of Operational Research).

3.1 Introduction

Over recent years, ride-hailing platforms (e.g., Uber, Lyft, or Bolt) have emerged as prominent components of urban mobility. These platforms typically operate in a city characterized by a network of locations in which drivers move around according to customer requests. They differ from ride-sharing (e.g., Blablacar) or car-sharing platforms (e.g., Cambio) in that customers do not drive the car but are served by drivers dedicated to the service.

When it comes to managing ride-hailing networks, balancing driver supply with customer demand is particularly challenging due to four main reasons. A first challenge is related to the *heterogeneity of the demand*, both in time and space. That is, the expected demand changes from one location to another (e.g., business district vs. residential areas), and depends on the time of the day (e.g., morning vs. evening peak hours). Second, ride-hailing platforms must deal with *interdependent supply and demand*, as satisfied customer requests affect the current availability of drivers in origin locations and the future availability of drivers in destination locations. Third, the users of ride-hailing platforms tend to exhibit *strategic behaviors*. On the one hand, customers determine whether to request services depending on their willingness to pay. On the other hand, drivers may decide to self-relocate to different locations due to higher expected earnings and their own preferences. Lastly, these platforms offer *on-demand services* rather than scheduled services. Customer requests must be served in real time, or they will be lost. This implies that the platforms need to rebalance their network within short periods.

Given these issues, ride-hailing platforms employ multiple decision levels to connect supply and demand. In this study, we consider three practical decisions: pricing, driver repositioning, and customer admission. These decisions occur over a rather large time scale (e.g., every five minutes), as they are

mainly used to balance aggregate flows of customers and drivers in the network. We distinguish them from operational decisions, which are made at a higher spatio-temporal granularity (e.g., driver dispatching, every thirty seconds). The pricing decision influences demand in time and space by pushing customers with low willingness to pay out of the system. It also impacts supply, as drivers tend to self-relocate to locations with higher prices (e.g., being guided by the platform through heatmaps, Uber, 2023). The repositioning decision affects supply, ordering drivers to reposition to other locations. It is expected to become prevalent with the rise of autonomous vehicles, and is already experienced by some ride-hailing platforms (Jiao et al., 2021). Lastly, the customer admission decision, either applied directly (e.g., warning that no driver is available) or indirectly (e.g., announcing long waiting times), allows further control of demand. This decision occurs when driver supply is insufficient to serve all the requests, and determines which customers are served based on their destinations. It is observed in practice, especially in markets where dynamic pricing is negatively perceived (Kanoria & Qian, 2024).

The mentioned decisions are often optimized separately in the literature, without accounting for potential interactions (cf. Section 3.2). However, as underlined by Qin et al. (2022), this can lead to inefficiencies since locally optimal decisions can be suboptimal for the whole system. For example, prices can be adjusted to reduce demand in a location, even though drivers could have easily been repositioned at the location in the first place. The goal of this chapter is therefore to model these three decisions and evaluate their respective effectiveness in balancing supply and demand. For this purpose, we develop a model formulated as a Markov decision process in which a ride-hailing platform determines the prices, the drivers to reposition, and the customers to admit in a network of locations over a finite time horizon. Besides the platform's repositioning decision, the model also allows all the drivers to relocate according to their own choices. Impatient customers arrive according to non-stationary Poisson processes, and decide whether to request service based on their willingness to pay. In this setting, the platform's objective is to find a dynamic policy that maximizes the revenues for serving customers at a specific price while minimizing the costs incurred by repositioning drivers and leaving some requests unsatisfied.

Due to the size and complexity of the problem, our model cannot be solved to optimality with conventional methods such as dynamic programming. Even advanced reinforcement learning algorithms take an excessively long time to reach policies that are still underperforming (Section 3.5.2). We thus deploy an original methodology that blends mathematical optimization with deep reinforcement learning (DRL). Our approach finds an efficient policy in three steps. First, a rolling horizon strategy is derived by repeatedly solving multiple optimization problems based on the expected values of stochastic parameters. Second, two neural networks are pretrained to reproduce the rolling horizon strategy, and to approximate its value function. Third, the policy is further improved using DRL. We demonstrate that this approach computes an im-

proved policy, using less computational effort in comparison with alternative methods such as rolling horizon strategies and DRL algorithms used alone. The resulting policy is then analyzed to explore the relevance of pricing, driver repositioning, and customer admission.

The main findings of our research can be summarized as follows.

- First, we introduce a stochastic model that optimizes pricing, driver repositioning, and customer admission. To the best of our knowledge, this chapter is the first attempt to integrate these three levers of decision in a single mathematical model. We do so in a comprehensive setting that includes spatio-temporal demand imbalances, stochastic demand and transit times, and independent drivers who can elect to self-relocate.
- Second, we develop an efficient approach that combines mathematical optimization with DRL. In numerical experiments, we show that our approach increases profit by 16.7% and 4% on average, compared to alternative rolling horizon strategies and DRL algorithms, while being 42%-65% faster than DRL algorithms. Our approach also benefits from stable learning, allowing us to handle instances with many zones and extended horizons.
- Third, we provide managerial insights on how pricing, driver repositioning, and customer admission help cope with supply-demand imbalances. In particular, we find that both pricing and driver repositioning are necessary to mitigate imbalances, while customer admission substitutes driver repositioning when driver supply becomes too scarce to be repositioned. Our results also suggest that driver repositioning by the platform is more effective than when drivers relocate by themselves.

The rest of the chapter is structured as follows. Section 3.2 discusses related literature, highlighting our contributions. Section 3.3 defines the problem, and formulates it as a Markov decision process. Section 3.4 first provides some background, and then presents our reinforcement learning approach. Section 3.5 presents numerical experiments to show the performance of our approach and bring out managerial insights. Section 3.6 concludes and suggests future research avenues.

3.2 Literature Review

This chapter is related to the recent literature on ride-hailing platforms and reinforcement learning. In this section, we first present these two literature streams, and then outline our contributions.

3.2.1 Ride-Hailing Platforms

In the ride-hailing literature (reviewed by H. Wang & Yang, 2019), we focus on research about pricing, driver repositioning, and customer admission, either

studied separately or together.

A significant number of papers examine the implications of pricing on supply and demand using single-location models (see, e.g., Bai et al., 2019, L. Sun et al., 2019). Other studies investigate how pricing can mitigate supply-demand imbalances in multiple locations. Bimpikis et al. (2019) develop a model in which a ride-hailing platform sets prices in multiple locations to influence the entry choice of customers and drivers, and the self-relocating behavior of drivers. S. Li et al. (2021) and Banerjee et al. (2022b) extend this research by examining alternative network structures, objective functions, and system constraints in related settings. In these studies, a common goal is to search for a pricing policy that is optimal in steady state. However, since real-world applications are dynamic and stochastic, they can lead to overly simplified problems. Similar to us, Nourinejad and Ramezani (2020), and Meskar et al. (2023) address this shortcoming by considering dynamic pricing under non-stationary arrival rates. Nevertheless, the former focus on temporal pricing while we also include spatial pricing; the latter consider deterministic demand while we capture stochastic customer arrivals.

Driver repositioning has also garnered extensive attention in the literature on ride-hailing platforms. On the one hand, many studies examine the repositioning decision as a *centralized* decision made by the platform. In this setting, early works derive static policies with analytical properties (e.g., Braverman et al., 2019), while recent works deduce dynamic policies that are more actionable in practice (e.g., Hosseini et al., 2024, Beojone et al., 2024). On the other hand, several studies consider driver repositioning as a *decentralized* decision made by drivers (called driver self-relocation in this chapter). Following this view, a few papers explore the use of subsidies (Zhu, Ke, & Wang, 2021), or information sharing (Beojone & Geroliminis, 2023) to incentivize drivers to relocate. Some papers, like the one by Shou et al. (2020), also address the repositioning problem at the individual driver level. All the mentioned papers suppose that the repositioning decision is either achieved by the platform or by drivers, not both. We take a broader perspective by developing a model where both platform-repositioning and driver-repositioning can occur. By doing this, we can compare and evaluate the two repositioning mechanisms.

Only limited research has investigated customer admission despite its practical value. De Munck et al. (2023) study customer admission decisions for a ride-hailing platform serving multiple customer classes with differentiated services. G. Wang et al. (2024) use a spatial queueing model to approximate the customer pickup time, and to admit customers whose pickup time is inferior to some threshold. Kanoria and Qian (2024) propose dynamic admission policies that do not require any knowledge of the customer arrival rates. These works mainly cover customer admission in a single location, without considering its integration with pricing and driver repositioning.

A few papers have jointly analyzed multiple decision levels on ride-hailing platforms. Notably, the repositioning and dispatching (also called matching) decisions are sometimes optimized together, by solving an integer optimiza-

tion model (Alonso-Mora et al., 2017, Tuncel et al., 2023). There also exist papers that integrate pricing with dispatching decisions in equilibrium models (e.g., Özkan, 2020). Closer to our work, Afèche et al. (2023) analyze driver repositioning and customer admission in a stylized queueing network. They find that customer admission can induce drivers to relocate to high-demand locations, and suggest that further research should examine the interactions with pricing under non-stationary arrival rates. Q. Chen et al. (2023) propose a policy that can adjust prices and reposition drivers dynamically in a ride-hailing network with non-homogeneous arrival rates. Yet, their policy does not consider customer admission, drivers with self-relocating behavior, and stochastic transit times. Their methodology also differs from ours, as they obtain a policy using mathematical optimization while we combine optimization with reinforcement learning.

3.2.2 Reinforcement Learning

We next discuss relevant literature on reinforcement learning by discussing recent developments in deep reinforcement learning (DRL), imitation learning, and applications to ride-hailing platforms.

The literature on reinforcement learning is extensive, dating back to the seminal work of Barto et al. (1983), and others in the late '80s. Following recent advances in deep learning, reinforcement learning benefits from a renewed interest, which has led to multiple successful applications (e.g., Silver et al., 2016). Several works, like those by Haarnoja et al. (2018), and Schulman et al. (2017), develop stable algorithms that can deal with continuous action spaces. In this chapter, we rely on Proximal Policy Optimization (PPO), an algorithm proposed by Schulman et al. (2017).

In contrast to traditional DRL methods, we leverage mathematical optimization in a pretraining process to enhance the overall training process. Our approach can be related to imitation learning, an umbrella of methods (see Hussein et al., 2017 for a review) that exploit external knowledge to make the learning process faster and more effective. Although promising, these methods are still underexplored in DRL, especially when applied to operations management problems (Boute et al., 2022). To date, the few successful applications have addressed inventory optimization problems. De Moor et al. (2022) design a deep Q-network algorithm guided by a modified reward term (the shaped reward) to mimic a given heuristic policy. As a result, the training process becomes faster and more stable. In the same vein, Oroojlooyjadid et al. (2022) solve the Beer Game with a deep Q-network algorithm. In their approach, the neural network of an agent is used to initialize the neural network of another agent, saving computational effort. Like the method proposed by Oroojlooyjadid et al. (2022), our approach uses pretrained neural networks. Unlike their method, the parameters of our neural networks are initialized from an optimization problem rather than from another trained agent. Furthermore, while these works apply imitation learning to systems with discrete actions, our

approach can be employed for systems with continuous actions.

Closest to our research are studies applying reinforcement learning to ride-hailing platforms (see Qin et al., 2022 for a review). However, many of them consider a single driver as the primary agent (e.g., Jiao et al., 2021, Qian et al., 2022). Only a few of them optimize ride-hailing systems in a centralized manner, considering the platform as the main agent. Among those, C. Chen et al. (2021) use the PPO algorithm to optimize continuous prices in a network of locations. Al-Kanj et al. (2020), Mao et al. (2020), and Kullman et al. (2022) investigate dispatching problems for autonomous vehicles, and employ value function approximation, policy-gradient, and attention-based DRL to deal with the explosion of the action space. Turan et al. (2020) apply DRL to manage a fleet of electric vehicles through pricing, dispatching, and charging decisions. Haliem et al. (2021) resort to the deep Q-network algorithm to address a joint routing and repositioning problem. P. Yan et al. (2023) propose a model-based SARSA algorithm to optimize dispatching and charging decisions for electric vehicles. In contrast to most of these works, we focus on ride-hailing platforms that manage independent drivers who choose to self-relocate rather than autonomous vehicles. Our model also includes stochastic transit times and arrival rates, which significantly complicate the derivation and interpretation of the resulting policy. More fundamentally, while the majority of these works focus on (dispatching) decisions over short time intervals, our model considers decisions to balance supply and demand over longer time intervals. In that respect, this chapter complements the mentioned studies by providing policies to complement decisions at a higher time granularity.

3.2.3 Contributions

This chapter makes two principal contributions to the literature. First, it provides a modeling contribution, as being the first to jointly optimize pricing, driver repositioning, and customer admission on ride-hailing platforms. Furthermore, in contrast to previous research, our framework offers extensive modeling of the system, including features such as (i) spatio-temporal supply-demand imbalances, (ii) stochastic demand and transit times, and (iii) idle drivers with self-relocating behavior. This original setting leads us to gather relevant insights on the relevance of pricing, driver repositioning, and customer admission. Second, this chapter provides a methodological contribution. While the application of imitation learning methods to operations management problems is still in its early stages, our approach stands out by its recourse to mathematical optimization as a way to accelerate the training process of a reinforcement learning algorithm. In contrast to alternative DRL algorithms, our approach can derive a better policy in a reduced amount of time through a stable learning process.

3.3 Markovian Model

In this section, we introduce the problem of a platform that manages its network through pricing, driver repositioning, and customer admission. We then formulate it as a Markov decision process (MDP). A summary of the notations used in this section can be found in Table 3.1.

Notation	Definition
Indexes and sets	
$i,\, j \in \mathcal{N} \equiv \{1, ..., N\}$	Locations i and j over the network $\mathcal{N}$.
$t \in \mathcal{T} \equiv \{0, ..., T-1\}$	Decision epoch t over the finite horizon $\mathcal{T}$.
Platform's decision variables	
$\mathbf{p}_t \equiv (p_{1,t}, ..., p_{i,t}, ..., p_{N,t})$	Price rates in location i at epoch t.
$\mathbf{r}_t \equiv (r_{11,t}, ..., r_{ij,t}, ..., r_{NN,t})$	Number of repositioned drivers from location i to j at epoch t.
$\mathbf{q}_t \equiv (q_{11,t}, ..., q_{ij,t}, ..., q_{NN,t})$	Percentages of admitted customers from location i to j at epoch t.
Demand parameters	
$F(\cdot)$ with $[0,\, \overline{f}]$	Cumulative distribution and support of the customer willingness to pay.
C^{lost}	Cost per unsatisfied customer request.
$\mathbf{\Lambda}_t \equiv (\Lambda_{11,t}, ..., \Lambda_{ij,t}, ..., \Lambda_{NN,t})$	Arrival rates of potential customers from location i to j at epoch t.
$\boldsymbol{\lambda}_t \equiv (\lambda_{11,t}, ..., \lambda_{ij,t}, ..., \lambda_{NN,t})$	Arrival rates of customer requests from location i to j at epoch t.
Demand variables	
$\mathbf{c}_t^{\text{req}} \equiv (c_{11,t}^{\text{req}}, ..., c_{ij,t}^{\text{req}}, ..., c_{NN,t}^{\text{req}})$	Number of customer requests from location i to j at epoch t.
$\mathbf{c}_t^{\text{adm}} \equiv (c_{11,t}^{\text{adm}}, ..., c_{ij,t}^{\text{adm}}, ..., c_{NN,t}^{\text{adm}})$	Number of admitted customer requests from location i to j at epoch t.
$\mathbf{c}_t^{\text{rej}} \equiv (c_{11,t}^{\text{rej}}, ..., c_{ij,t}^{\text{rej}}, ..., c_{NN,t}^{\text{rej}})$	Number of rejected customer requests from location i to j at epoch t.
Supply parameters	
D	Total number of drivers in the network.
$1/\tau_{ij}$	Expected transit time from location i to j.
δ_{ij}	Distance from location i to j.
β^{cons}, β^{price}, β^{trans}	Preference coefficients of self-relocating drivers.
γ	Temperature parameter of the dispersion of the self-relocating choice.
C^{rep}	Cost rate per repositioned driver.
Supply variables	
$\mathbf{x}_t \equiv (x_{1,t}, ..., x_{i,t}, ..., x_{N,t})$	Number of drivers available in location i at epoch t.
$\mathbf{y}_t \equiv (y_{11,t}, ..., y_{ij,t}, ..., y_{NN,t})$	Number of drivers in transit from location i to j at epoch t.
$\mathbf{d}_t^{\text{self}} \equiv (d_{11,t}^{\text{self}}, ..., d_{ij,t}^{\text{self}}, ..., d_{NN,t}^{\text{dis}})$	Number of drivers who self-relocate from location i to j at epoch t.
$\mathbf{d}_t^{\text{dis}} \equiv (d_{11,t}^{\text{dis}}, ..., d_{ij,t}^{\text{dis}}, ..., d_{NN,t}^{\text{dis}})$	Number of drivers who are dispatched from location i to j at epoch t.
$\mathbf{d}_t^{\text{fin}} \equiv (d_{11,t}^{\text{fin}}, ..., d_{ij,t}^{\text{fin}}, ..., d_{NN,t}^{\text{fin}})$	Number of drivers who finish their journey from location i to j at epoch t.

Table 3.1: Summary of the notations.

3.3.1 Problem Setting

We consider a ride-hailing platform that operates in a region (e.g., a city) over a finite time horizon (e.g., a working day). The region is divided into a set $\mathcal{N} \equiv \{1, \ldots, N\}$ of disjoint locations that represent zones of reasonable size (e.g., neighborhoods). The platform makes decisions over a set $\mathcal{T} \equiv \{0, 1, \ldots, T-1\}$ of discrete epochs, separated by small periods of time Δt (e.g., every five minutes). Next, we describe the sequence of decisions and events between two epochs, as depicted in Fig. 3.1.

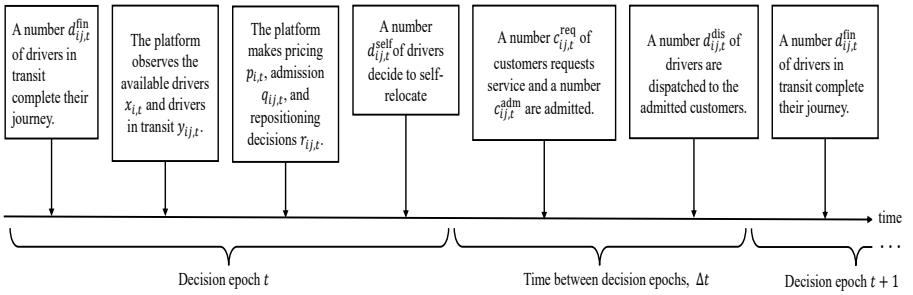


Figure 3.1: Timeline of decisions and events.

1. At each decision epoch t, the platform observes the number of drivers $d_{ij,\,t}^{\text{fin}}$ who finish their journey from location i to j. It can then deduce the number of available drivers in location i, denoted by $x_{i,\,t}$, and the number of drivers in transit from location i to j, denoted by $y_{ij,\,t}$.

2. Then, the platform maximizes its profit rate through pricing, driver repositioning, and customer admission decisions:

   – *The pricing decision* consists of setting the price rate $p_{i,\,t}$ that is charged per unit of distance for a service starting from location i. As frequently observed in practice, the price rate depends on the origin location of the customer and the distance of the trip, but is independent of the destination location (Uber, 2022).

   – *The repositioning decision* refers to the number of drivers $r_{ij,\,t}$ who are moved empty by the platform from location i to j (with $i,\, j \in \mathcal{N}$ such that $i \neq j$). For each repositioned driver, the platform incurs a repositioning cost rate C^{rep} per unit of distance, as in industry practices (e.g., Lyft's bonus zones, 2021a).

   – *The admission decision* determines the probability $q_{ij,\,t}$ for a customer going from location i to j to be admitted to the system (possibly with $i, j \in \mathcal{N}$ such that $i = j$). Since future demand is random, this decision is expressed as a probability rather than a number of customers. For every rejected customer, the platform incurs a fixed lost sales cost C^{lost}, reflecting a loss of brand image. While this lost-sale cost may depend on the origin and destination location (or even, the prices paid by the customers), here we assume it to be constant for simplicity.

3. After the platform's decisions, any driver who has not been repositioned by the platform can choose to self-relocate to another location. The number of drivers who elect to self-relocate from a location i to another location j is denoted by $d_{ij,\,t}^{\text{self}}$ (with $i,\, j \in \mathcal{N}$ such that $i \neq j$). Throughout

this chapter, these drivers are called the "self-relocating drivers". We distinguish them from the "repositioned drivers" who follow the platform's repositioning decision, and the "dispatched drivers" who satisfy customer requests.

4. Between decision epochs t and $t+1$, potential customers who seek to go from location i to location j arrive following a non-stationary Poisson process with rate $\Lambda_{ij,\,t}$ (possibly with $i,\, j \in \mathcal{N}$ such that $i = j$). Advance booking is not modeled as it only represents a minor share of ride-hailing revenues (Aten, 2023). Upon arrival, potential customers observe the price rate in their location, and decide whether to request service based on their willingness to pay. Then, realized requests are admitted depending on the admission decision. Following each admission, an available driver is dispatched to the admitted request if possible. The served customer is considered to be picked up as soon as a driver is dispatched. If the request cannot be satisfied, the customer opts for an alternative mode of transportation (e.g., the subway).[1] Like for rejected requests, each unsatisfied request generates a lost sales cost C^{lost}. The number of realized requests, admitted requests, and dispatched drivers are aggregated at the location level, and denoted by $c^{\text{req}}_{ij,\,t}$, $c^{\text{adm}}_{ij,\,t}$, and $d^{\text{dis}}_{ij,\,t}$ from location i to j.

To complete the problem definition, the following system dynamics are specified:

- *Driver availability*: Similar to recent literature (e.g., Q. Chen et al., 2023, and P. Yan et al., 2023), we suppose that the total number of drivers in the network is fixed to a constant D (we relax this assumption in Section 3.5.4). In addition, we assume that drivers in transit finish their journey from i to j after exponentially distributed transit times with mean $1/\tau_{ij}$. The randomly distributed transit times reflect the fact that transit times can be highly uncertain and are not always validly approximated by deterministic values. The assumption that their distributions are exponential also allows us to keep the model Markovian, improving the model tractability.

- *Driver self-relocation*: Although any discrete choice model can be used, the driver self-relocating choice is represented with a multinomial logit model. Let β^{cons} be the constant term, β^{price} be the expected driver preference toward locations with prices higher than in their location, and β^{trans} be the expected aversion toward locations with long transit times. The expected utility of a self-relocating driver from location i to j is defined as

$$u_{ij,\,t} = \beta^{\text{cons}} + \beta^{\text{price}} \cdot (p_{j,\,t} - p_{i,\,t}) - \beta^{\text{trans}} \cdot 1/\tau_{ij}. \tag{3.1}$$

[1] Since we focus on decisions over long time intervals (e.g., every 5 minutes), it is reasonable to suppose that customers are directly picked up, and leave if not served between epochs.

If the utility of staying in the current location i is normalized to 0, then the probability for a driver to self-relocate from location i to j is given by $\psi_{ij,\,t} = \exp(u_{ij,\,t}/\gamma)/\big[1 + \sum_{\substack{k=1\\k\neq i}}^{N} \exp(u_{ik,\,t}/\gamma)\big]$, with $\psi_{ii,\,t}$ being the probability that drivers stay idle in location i. The temperature parameter γ measures the dispersion of the driver choice. As $\gamma \to 0$, drivers choose with certainty the option with the highest expected utility. As $\gamma \to \infty$, drivers choose each option with equal probability. In our experiments, we vary γ to capture different levels of randomness in the driver self-relocation choice.

- *Customer willingness to pay*: The customer willingness to pay is heterogeneous, and characterized by the concave[2] cumulative distribution $F(\cdot)$ over the support $[0,\, \overline{f}]$ with $\overline{f}$ being the maximum customer willingness to pay. To simplify exposition, we assume that the function $F(\cdot)$ is location-independent and time-independent. Given price rate $p_{i,\,t}$, the expected number of customer requests from location i to j can be written as

$$\lambda_{ij,\,t} = \Lambda_{ij,\,t} \cdot \big[1 - F(p_{i,\,t})\big]. \tag{3.2}$$

  The realized number of customer requests is thus sampled from a Poisson distribution with rate $\lambda_{ij,\,t}$. As each realized request has an independent probability of being admitted, the number of admitted customer requests $c^{\text{adm}}_{ij,\,t}$ satisfies the Poisson thinning property, and follows a Poisson distribution, with rate $\lambda_{ij,\,t} \cdot q_{ij,\,t}$.

- *Driver dispatching*: For simplicity, we assume that drivers can only be dispatched to admitted requests in the same location, on a first-come, first-served (FCFS) basis. The first assumption reflects the practical idea that ride-hailing platforms match drivers and customers close to each other to avoid long pick-up times. The second assumption is common in theory (e.g., Afèche et al., 2023) and practice (e.g., Lyft, 2024), as it reduces the problem complexity and is perceived as fairer than other queueing regimes. Consequently, the number of dispatched drivers from location i to j, $d^{\text{dis}}_{ij,\,t}$, depends on the number of admitted requests and available drivers in i. If there are fewer admitted requests than available drivers, then $d^{\text{dis}}_{ij,\,t} = c^{\text{adm}}_{ij,\,t}$ for all $j \in \mathcal{N}$. Otherwise, only a fraction of the demand is served, and $d^{\text{dis}}_{ij,\,t} < c^{\text{adm}}_{ij,\,t}$ for some $j \in \mathcal{N}$ (depending on the order of arrivals). Note that the dispatching process could be complexified, e.g., by adopting "batch" matching. Yet, it would require an additional layer of decisions, which is beyond the scope of this paper.

[2]The concavity assumption is in line with the revenue management literature (cf. Talluri et al., 2004), and fits many common distributions including the uniform, exponential, and logit distributions.

3.3.2 MDP Formulation

In this section, we formalize the problem introduced above as a discrete-time Markov decision process (MDP). To this end, we detail the states, decisions, transitions, rewards, and policies.

States and Decisions

The system state $\mathbf{s}_t$ at decision epoch t can be characterized by the number of drivers available $x_{i,\,t}$ in each location i, and the number of drivers in transit $y_{ij,\,t}$ from each location i to each location j. It is defined as the tuple $\mathbf{s}_t = (\mathbf{x}_t,\, \mathbf{y}_t)$, with vector $\mathbf{x}_t = (x_{1,\,t},\, \dots,\, x_{N,\,t})$ and matrix $\mathbf{y}_t = (y_{11,\,t},\, \dots,\, y_{NN,\,t})$. Therefore, the state space comprises $N+N^2$ dimensions. It only contains states such that the total number of drivers in the network equals D, the fixed number of drivers. The state space thus comprises $\frac{\left(D+N+N^2-1\right)!}{D!\left(N+N^2-1\right)!}$ possible states, and is described by the set

$$\mathcal{S} = \Big\{(\mathbf{x}_t,\, \mathbf{y}_t) : t \in \mathcal{T},\, \mathbf{x}_t \in \mathbb{Z}^N,\, \mathbf{y}_t \in \mathbb{Z}^{N\times N},\, \sum_{i=1}^{N} x_{i,\,t} + \sum_{i=1}^{N}\sum_{j=1}^{N} y_{ij,\,t} = D\Big\}. \tag{3.3}$$

Note that the system state does not include the number of customers waiting for service, as we assume customers are impatient.

At decision epoch t, after observing the system state $\mathbf{s}_t$, the platform makes the pricing $p_{i,\,t}$, repositioning $r_{ij,\,t}$, and admission $q_{ij,\,t}$ decisions, which can be represented by a tuple, denoted by $\mathbf{a}_t = (\mathbf{p}_t,\, \mathbf{r}_t,\, \mathbf{q}_t)$. The first component $\mathbf{p}_t = (p_{1,\,t},\, \dots,\, p_{N,\,t})$ is the vector comprising the price rates. Each price rate is continuous and bounded above by the maximum customer willingness to pay $\overline{f}$ as higher prices would result in no customer requests. The second component $\boldsymbol{r}_t = (r_{11,\,t},\, \dots,\, r_{N(N-1),\,t})$ is the matrix composed of the number of repositioned drivers. The number of repositioned drivers from a location i is always inferior or equal to the number of available drivers in the location, $x_{i,t}$. The third component $\boldsymbol{q}_t = (q_{11,\,t},\, \dots,\, q_{NN,\,t})$ is the matrix that contains the probabilities that a request arriving after the decision epoch is admitted, depending on the origin and destination location. Consequently, the decision space has $2 \times N^2$ dimensions, and is given by

$$\mathcal{A}(\mathbf{s}_t) = \Big\{(\mathbf{p}_t,\, \mathbf{r}_t,\, \mathbf{q}_t) \,:\, \mathbf{p}_t \in [0,\, \overline{f}]^N,\, \mathbf{r}_t \in \mathbb{Z}^{N\times(N-1)},\, \mathbf{q}_t \in [0,\, 1]^{N\times N},\, \sum_{j=1}^{N} r_{ij,t} \le x_{i,t}\ \ \forall\, i \in \mathcal{N}\Big\}. \tag{3.4}$$

State Transitions

Following the platform's decisions, the state transitions from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. As shown in Fig. 3.1, the state $\mathbf{s}_t = (\mathbf{x}_t,\, \mathbf{y}_t)$ first evolves deterministically, due

to the repositioning decision $\mathbf{r}_t$. Then, it changes stochastically, as a function of the self-relocating drivers $\mathbf{d}_t^{\text{self}} = (d_{11,\,t}^{\text{self}}, \ldots, d_{NN,\,t}^{\text{self}})$, the dispatched drivers $\mathbf{d}_t^{\text{dis}} = (d_{11,\,t}^{\text{dis}}, \ldots, d_{NN,\,t}^{\text{dis}})$, and the drivers in transit who finish their journey $\mathbf{d}_{t+1}^{\text{fin}} = (d_{11,\,t+1}^{\text{fin}}, \ldots, d_{NN,\,t+1}^{\text{fin}})$, respecting that order. The self-relocating drivers $\mathbf{d}_{ij,\,t}^{\text{self}}$ are influenced by the pricing decision $\mathbf{p}_t$ while the dispatched drivers $d_{ij\,t}^{\text{dis}}$ indirectly depend on the pricing decision $\mathbf{p}_t$ and the admission decision $\mathbf{q}_t$. In mathematical terms, the number of drivers available in location i at the decision epoch $t+1$ is expressed as

$$x_{i,\,t+1} = x_{i,\,t} - \sum_{j=1}^{N} r_{ij,\,t} - \sum_{j=1}^{N} d_{ij,\,t}^{\text{self}} - \sum_{j=1}^{N} d_{ij,\,t}^{\text{dis}} + \sum_{j=1}^{N} d_{ji,\,t+1}^{\text{fin}} \qquad \forall\, i \in \mathcal{N},\, t \in \mathcal{T}. \tag{3.5}$$

On the right-hand side, the first term represents available drivers; the second (drivers repositioned by the platform), third (self-relocating drivers), and fourth (dispatched drivers) terms stand for drivers leaving the location, while the last term refers to drivers entering the location.

Following the same logic as in Eq. (3.5), the number of drivers in transit from location i to j at the decision epoch $t+1$ is given by

$$y_{ij,\,t+1} = y_{ij,\,t} + r_{ij,\,t} + d_{ij,\,t}^{\text{self}} + d_{ij,\,t}^{\text{dis}} - d_{ij,\,t+1}^{\text{fin}} \qquad \forall\, i \in \mathcal{N},\, j \in \mathcal{N},\, t \in \mathcal{T}. \tag{3.6}$$

Note that we always have $r_{ii,\,t} = 0$ and $d_{ii,\,t}^{\text{self}} = 0$ since drivers are never repositioned, and never self-relocate to their current location.

In Eq. (3.5) and Eq. (3.6), the three stochastic components $\mathbf{d}_t^{\text{self}}$, $\mathbf{d}_t^{\text{dis}}$ and $\mathbf{d}_t^{\text{fin}}$ are derived as follows. First, the number of self-relocating drivers $\mathbf{d}_t^{\text{self}}$ is modeled according to a multinomial distribution specific to each location i. This is because drivers elect to self-relocate according to a multinomial logit model. For each distribution, the number of trials is given by the number of drivers remaining in location i after platform repositioning. The probability for drivers to self-relocate to another location j is encoded by the probability vector $(\psi_{i1,\,t}, \ldots, \psi_{ij,\,t}, \ldots, \psi_{iN,\,t})$, deduced from the multinomial logit model. Second, the number of dispatched drivers $\mathbf{d}_t^{\text{dis}}$ follows a hypergeometric distribution[3] that is also specific to each location i. This is because admitted requests arrive according to independent Poisson processes (due to the Poisson thinning property), and are served in an FCFS order. The distribution is characterized by the number of drivers in location i after driver repositioning and driver self-relocating, and the number of admitted requests going from location i to any location j, summarized by the vector $(c_{i1,\,t}^{\text{adm}}, \ldots, c_{ij,\,t}^{\text{adm}}, \ldots, c_{iN,\,t}^{\text{adm}})$. Lastly, the number of drivers finishing their journey $\mathbf{d}_{t+1}^{\text{fin}}$ follows a binomial distribution as transit times are exponentially distributed and thus memoryless. For each distribution, the number of trials corresponds to the drivers in transit between

[3]The multivariate hypergeometric distribution characterizes sampling without replacement from a finite population (the available drivers in location i) when each outcome falls into one of several categories (the service of a customer going to location j).

the decision epoch t and $t+1$. The probability for drivers to finish their journey from location i to j between two decision epochs can be expressed as $(1 - e^{-\tau_{ij} \cdot \Delta t})$.

Rewards

Service revenues, lost sales costs, and repositioning costs occur randomly at each decision epoch, along with the state transition. These are captured by the following reward function:

$$\begin{aligned} R_t(\mathbf{s}_t,\, \mathbf{a}_t) = &\sum_{i=1}^{N}\sum_{j=1}^{N} p_{i,\,t} \cdot d^{\text{dis}}_{ij,\,t} \cdot \delta_{ij} \\ &- C^{\text{lost}} \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} (c^{\text{req}}_{ij,\,t} - d^{\text{dis}}_{ij,\,t}) - C^{\text{rep}} \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij,\,t} \cdot \delta_{ij}, \end{aligned} \tag{3.7}$$

where the first term stands for the platform revenues, and the second and third terms refer to the lost sales costs and repositioning costs. According to Eq. (3.7), the platform collects some revenue for every driver dispatched from location i to j. This revenue is proportional to the price rate $p_{i,t}$ in the origin location i, and the distance δ_{ij} between the origin i and destination j. For every unsatisfied request, the platform incurs a fixed lost sales cost C^{lost}. Lastly, the cost for repositioning a driver depends on the distance δ_{ij} traveled by the driver from location i to j. In our model, the repositioning cost is entirely accounted for when a driver starts repositioning, with a cost rate C^{rep} per unit of distance.

Policies and Value Function

Over a finite horizon $\mathcal{T}$, a policy is a sequence $\pi = \{\pi_0,\, \pi_1,\, \pi_2, \ldots,\, \pi_{T-1}\}$ where π_t is a function mapping the system states to decisions made in those states (at epoch t). Let Π be the set of feasible policies and $\mathbf{s}_0$ be the state at the beginning of the decision horizon. Our objective is to find an optimal policy π^* that maximizes the expected total reward:

$$\pi^* \equiv \arg\max_{\pi \in \Pi} \mathbb{E}_{\pi}\Big[\sum_{t=0}^{T-1} R_t\big(\mathbf{s}_t,\, \pi_t(\mathbf{s}_t)\big) \,\Big|\, \mathbf{s}_0 \Big]. \tag{3.8}$$

To find the optimal policy π^*, the optimal value function can be computed. The optimal value function $v^*(\mathbf{s}_t)$ $\big($with $v^*(\mathbf{s}_t) \equiv v_{\pi^*}(\mathbf{s}_t)\big)$ measures the total future reward that can be expected from a given state $\mathbf{s}_t$ following an optimal policy π^* (Powell, 2007). In our case, it stands for the total future profit expected until the end of the horizon. We can define it recursively as

$$v^*(\mathbf{s}_t) = \max_{\pi(\mathbf{s}_t) \in \mathcal{A}(\mathbf{s}_t)} \mathbb{E}_{\pi}\Big[R_t\big(\mathbf{s}_t,\, \pi_t(\mathbf{s}_t)\big) + v^*(\mathbf{s}_{t+1}) \,\Big|\, \mathbf{s}_t \Big]. \tag{3.9}$$

In theory, Eq. (3.9) can be solved using dynamic programming so as to maximize the expected total profit (Bellman, 1957). Nevertheless, for our specific model, the curse of dimensionality makes this method intractable, as the number of states and transitions grows exponentially with the number of drivers operating and the number of locations. The decision space is also partially continuous (due to the pricing and admission decisions), while dynamic programming is more suited for discrete decisions. Deep reinforcement learning (DRL) can handle large state spaces, and continuous decisions. Yet, given the complexity of our model, even advanced DRL algorithms fail to learn efficient policies. For this reason, we devise a method to improve upon existing DRL in the next section.

3.4 Reinforcement Learning with Pretraining

In this section, we develop an approach that combines deep reinforcement learning (DRL) with mathematical optimization. At its core, the approach relies on a pretraining process that teaches actor-critic neural networks to reproduce decisions, followed by a rolling horizon strategy. The approach proceeds in three steps. First, the rolling horizon strategy is simulated in the MDP by repeatedly solving an optimization problem based on expected values of stochastic parameters. Second, during actor pretraining, the actor learns a policy using the state-decision pairs obtained by simulating the rolling horizon strategy. Third, during critic pretraining, the critic neural network approximates the value function associated with the actor policy. After these steps, the DRL algorithm starts learning from an already efficient policy rather than from scratch, saving computational efforts while reaching improved performances. In the following, Section 3.4.1 introduces our actor-critic framework. Then, Section 3.4.2 details our pretraining process and its three components: the rolling horizon strategy, actor pretraining, and critic pretraining.

3.4.1 Actor-Critic Framework

Our approach is grounded in actor-critic methods, which are made of two components (cf. Sutton & Barto, 2018 for a more detailed exposition). On the one hand, the *actor* looks for a near-optimal policy. To this end, it derives a policy function $\pi(\boldsymbol{\theta})$ that selects decisions according to the set of parameters $\boldsymbol{\theta}$ (which depends on the approximation method). On the other hand, the *critic* approximates the corresponding value function using the set of parameters $\boldsymbol{\eta}$ (which also depends on the approximation method). This approximation, noted $\widehat{v}_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t, \boldsymbol{\eta})$, guides the search for a near-optimal policy by the actor.

Although any parametric function can approximate the value and policy functions, we choose the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ to be the weights of two distinct neural networks. Neural networks are flexible tools that can, in theory, approximate any function (LeCun et al., 2015). In our approach, the critic neu-

ral network returns the approximate value function $\widehat{v}_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t, \boldsymbol{\eta})$, but the actor neural network does not directly predict decisions. Instead, the actor neural network learns $N \cdot (N+1)$ mean parameters and $N \cdot (N+1)$ standard deviation parameters. These parameters are then used to define a set of $N \cdot (N+1)$ Gaussian distributions. A vector $\mathbf{p}_t^{\text{net}} \in [0, 1]^N$, associated with the pricing decision, and a matrix $\mathbf{rq}_t^{\text{net}} \in [-1, 1]^{N \times N}$, associated with the repositioning and admission decisions, are then sampled from these distributions, ensuring the exploration of new decisions.

At each epoch t, the three decisions can be derived from $\mathbf{p}_t^{\text{net}} \in [0, 1]^N$ and $\mathbf{rq}_t^{\text{net}} \in [-1, 1]^{N \times N}$. The pricing decision $\mathbf{p}_t$ is obtained by scaling each component of $\mathbf{p}_t^{\text{net}}$ with the customer maximum willingness to pay $\overline{f}$ such that $p_{i,t} = p_{i,t}^{\text{net}} \cdot \overline{f}$. The repositioning decision $\mathbf{r}_t$ and the admission decision $\mathbf{q}_t$ are both determined from the matrix $\mathbf{rq}_t^{\text{net}}$, hence reducing the number of parameters in the actor neural network. This follows the intuition that the platform should never reposition drivers and reject customers simultaneously, for a pair of locations.

From location i to j, the percentage of admitted customers $q_{ij,t}$, and the number of repositioned drivers $r_{ij,t}$ are deduced from $\mathbf{rq}^{\text{net}}$ using the following transformations:

$$r_{ij,t} = \begin{cases} \left\lfloor x_{i,t} \cdot \frac{rq_{ij,t}^{\text{net}}}{\sum_{k=1}^{N} (rq_{ik,t}^{\text{net}})^+} \right\rfloor & \text{if } rq_{ij,t}^{\text{net}} > 0,\ i \neq j, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{and} \tag{3.10}$$

$$q_{ij,t} = \begin{cases} |rq_{ij,t}^{\text{net}}| & \text{if } rq_{ij,t}^{\text{net}} \leq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

where $\lfloor \cdot \rfloor$ is the round down operator, and $(\cdot)^+ = \max(0, \cdot)$. If $rq_{ij,t}^{\text{net}} > 0$, then all the customer requests are admitted ($q_{ij,t} = 0$), and a fraction of the available drivers $x_{i,t}$ reposition from location i to j. In this case, the component $rq_{ii,t}^{\text{net}} > 0$ determines the share of drivers who are left available in location i.[4] Conversely, if $rq_{ij,t}^{\text{net}} \leq 0$, then only a percentage given by $|rq_{ij,t}^{\text{net}}|$ of the customer requests are admitted, and no driver is repositioned from location i to j ($r_{ij,t} = 0$).

Various DRL algorithms could train the actor-critic neural networks. In this chapter, we use the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) for two reasons. First, PPO benefits from high stability and sample efficiency, reducing the amount of data needed to improve the policy. Second, PPO is known to require little hyperparameter tuning. PPO is thus convenient in our situation because the pretraining process in Section 3.4.2 involves additional hyperparameters. For a short introduction to the PPO algorithm, we refer to Chapter 1, Section 1.3.2.

[4] Note that our formulation does not allow customers to be rejected (i.e., $rq_{ii,t}^{\text{net}} < 0$) while drivers are left available (i.e., $rq_{ii,t}^{\text{net}} > 0$). This limitation does not cause notable profit loss, as shown in Section 3.5.2.

3.4.2 Pretraining Process

Next, we present the process used to pretrain the actor-critic neural networks. The process has three components: the rolling horizon strategy, actor pretraining, and critic pretraining.

Rolling Horizon Strategy

This section defines a rolling horizon strategy, which periodically solves a deterministic optimization problem. Such a strategy serves two purposes. First, it generates state decision pairs that are used to pre-train the neural network of the actor (Section 3.4.2). Second, it provides a proper benchmark to compare our reinforcement learning approach (Section 3.5.2).

Before defining the rolling horizon strategy, we first formalize its underlying optimization problem. When all uncertainty is neglected, a natural way to represent our problem is to formulate a multi-period optimization problem based on the expected values (Birge & Louveaux, 2011). Specifically, we propose an optimization problem, named the Expected Value Problem (EVP), that (i) replaces the stochastic number of requests, dispatched drivers, and service completions by their expected values, (ii) ignores self-relocating drivers (that would otherwise make the model intractable as it would lead to non-convex constraints), and (iii) allows continuous rather than integer variables. The parameters of the problem can be assessed from available historical data. It takes an initial state as input and deduces prices, rejected customers, and repositioned drivers for future decision epochs in a deterministic setting.

Let us introduce $\mathbf{s}_0 = (\mathbf{x}_0, \mathbf{y}_0)$ the initial system state, and $\rho_{ij,\,t} = e^{-\tau_{ij}\cdot\Delta t}\cdot(1 - e^{-\tau_{ij}\cdot\Delta t})^t$ the probability that a driver takes t decision epochs ($t \geq 0$) to finish her journey from location i to j. Then, the EVP can be formulated as

$$\max \sum_{t=0}^{T-1}\sum_{i=1}^{N}\sum_{j=1}^{N} p_{i,t}\cdot d^{\text{dis}}_{ij,t}\cdot\delta_{ij} - C^{\text{lost}}\cdot\sum_{t=0}^{T-1}\sum_{i=1}^{N}\sum_{j=1}^{N} c^{\text{rej}}_{ij,t} - C^{\text{rep}}\cdot\sum_{t=0}^{T-1}\sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij,t}\cdot\delta_{ij} \tag{3.12a}$$

$$\text{s.t.}\quad c^{\text{req}}_{ij,\,t} = \Lambda_{ij,\,t}\cdot\big(1 - F(p_{i,\,t})\big) \qquad \forall\, i,\, j,\, t, \tag{3.12b}$$

$$d^{\text{dis}}_{ij,\,t} = c^{\text{req}}_{ij,\,t} - c^{\text{rej}}_{ij,\,t} \qquad \forall\, i,\, j,\, t, \tag{3.12c}$$

$$x_{i,\,0} \geq \sum_{j=1}^{N}\big(d^{\text{dis}}_{ij,\,0} + r_{ij,\,0}\big) \qquad \forall\, i, \tag{3.12d}$$

$$x_{i,\,0} + \sum_{k=0}^{t-1}\sum_{s=0}^{t-k-1}\sum_{j=1}^{N}\rho_{ji,\,s}\cdot\big(d^{\text{dis}}_{ji,\,k} + r_{ji,\,k}\big) + \sum_{s=0}^{t-1}\sum_{j=1}^{N}\rho_{ji,\,s}\cdot y_{ji,\,0} \geq \sum_{k=0}^{t}\sum_{j=1}^{N}\big(d^{\text{dis}}_{ij,\,k} + r_{ij,\,k}\big) \qquad \forall\, i,\, t \geq 1, \tag{3.12e}$$

$$p_{i,t},\, c^{\text{req}}_{ij,t},\, c^{\text{rej}}_{ij,t},\, d^{\text{dis}}_{ij,t},\, r_{ij,t} \geq 0 \qquad \forall\, i,\, j,\, t. \qquad (3.12f)$$

Similar to Eq.(3.7), objective function (3.12a) includes revenues for serving customers and costs for rejecting customers and relocating drivers. Constraints (3.12b) determine the number of requests between two locations based on the arrival rates of potential customers, prices, and the distribution $F(\cdot)$ of the customer willingness to pay. Constraints (3.12c) ensure that the number of dispatched drivers between two locations equals the number of admitted customers. Because of the absence of uncertainty, requests can never be admitted but not served. Constraints (3.12d) and constraints (3.12e), interpreted as "flow constraints," enforce that the number of dispatched and repositioned drivers is inferior to the number of available drivers in each location for each epoch. In constraints (3.12e), the first term is associated with the initial number of available drivers; the second term refers to the number of drivers initially in transit, who finished their journey; the third term represents the number of dispatched and repositioned drivers who finished their journey; the fourth term, on the right-hand side, corresponds to the number of dispatched and repositioned drivers. Finally, constraints (3.12f) are nonnegativity constraints. The EVP is nonlinear due to the first term in objective function (3.12a) and the convexity of constraints (3.12b). However, it can be made convex using McCormick envelopes (1976), and solved within a few seconds (typically less than a minute), using a commercial solver.

Having introduced the deterministic optimization model, we can now present the rolling horizon strategy. The strategy progresses through decision epochs by periodically re-optimizing the EVP every T^{RH} decision epochs and applying the decisions until the next re-optimization, over and over. At each re-optimization, the current state $\mathbf{s}^{\text{RH}}_t$ is taken as the initial system state in the EVP to account for dynamic variations that occurred between two optimizations. The EVP is re-optimized over the next Q^{RH} decision epochs (with $Q^{\text{RH}} \geq T^{\text{RH}}$) so as to capture future demand patterns. Because the EVP considers continuous variables and neglects uncertainty, its optimal solutions cannot be directly applied in the MDP. Therefore, the optimal solutions must be converted into feasible decisions $\mathbf{p}^{\text{RH}}$, $\mathbf{q}^{\text{RH}}$, and $\mathbf{r}^{\text{RH}}$. The pricing decision is determined as $p^{\text{RH}}_{i,t} = p_{i,t}$ in location i at epoch t. The admission decision is calculated as $q^{\text{RH}}_{ij,t} = 1 - c^{\text{rej}}_{ij,t}/c^{\text{req}}_{ij,t}$, and the repositioning decision as $r^{\text{RH}}_{ij,t} = \lfloor r_{ij,t} \rfloor$ from location i to j at epoch t.

As part of the rolling horizon strategy, the optimization interval T^{RH} and optimization horizon Q^{RH} can balance solution quality with computational effort and memory use. On the one hand, the optimization interval T^{RH} finds a trade-off between static decisions involving few optimizations (with high T^{RH}), and dynamic decisions involving many optimizations (with low T^{RH}). On the other hand, the optimization horizon Q^{RH} balances myopic decisions requiring little computation and memory per optimization (with low Q^{RH}), with farsighted decisions requiring more computations and memory per optimization (with high Q^{RH}). By varying these two parameters, the rolling horizon strategy

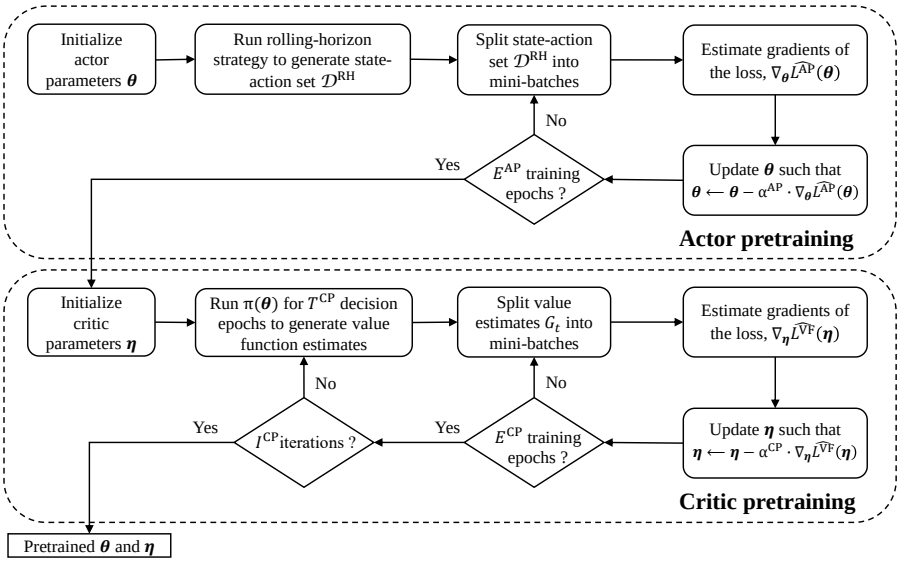


Figure 3.2: Flow diagram of the pretraining process.

can be adapted to our purpose. High T^{RH} and low Q^{RH} are favored to quickly pretrain the actor neural network (Section 3.4.2). Low T^{RH} and high Q^{RH} are used to benchmark our approach against a strong baseline (Section 3.5.2).

Actor Pretraining

To pretrain our actor neural network, we simulate the rolling horizon strategy in the MDP and collect information about the states visited and the decisions prescribed. That is, at each decision epoch t, the system state $\mathbf{s}_t^{\mathrm{RH}}$ and the decision $\mathbf{a}_t^{\mathrm{RH}}$ are stored in a state-decision set containing all the state-decision pairs observed so far, and defined as $\mathcal{D}^{\mathrm{RH}} = \left\{(\mathbf{s}_k^{\mathrm{RH}}, \mathbf{a}_k^{\mathrm{RH}})\right\}_{k=0}^{t}$. By repeating the operation for many decision epochs (possibly through parallel computing), the state-decision set $\mathcal{D}^{\mathrm{RH}}$ gradually covers more states that are likely to be visited, and records the associated decisions. In the end, the set includes various state-decision pairs, producing a partial policy (i.e., a mapping from a subset of the states to decisions). This partial policy offers an accurate representation of the rolling horizon strategy.

The actor neural network then uses the state-decision set $\mathcal{D}^{\mathrm{RH}}$ to learn a policy that mimics the rolling horizon strategy. Following standard machine learning terminology (LeCun et al., 2015), the state-decision set can be viewed as a set of training examples in which each state $\mathbf{s}_t^{\mathrm{RH}}$ is a feature vector, and each decision $\mathbf{a}_t^{\mathrm{RH}}$ is the corresponding label. Accordingly, we can use supervised learning methods, and train the policy function $\pi(\boldsymbol{\theta})$ to replicate the decisions contained in the set $\mathcal{D}^{\mathrm{RH}}$. Such a procedure is commonly referred

to as imitation learning (Boute et al., 2022). In our case, it boils down to minimizing the following loss function:

$$L^{\mathrm{AP}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}^{\mathrm{RH}}|} \sum_{(\mathbf{s}_t^{\mathrm{RH}},\, \mathbf{a}_t^{\mathrm{RH}}) \in \mathcal{D}^{\mathrm{RH}}} \Big[\frac{1}{2}\big(\pi(\mathbf{s}_t^{\mathrm{RH}},\, \boldsymbol{\theta}) - \mathbf{a}_t^{\mathrm{RH}}\big)^2\Big], \tag{3.13}$$

where $|\mathcal{D}^{\mathrm{RH}}|$ is the cardinality of the state-decision set. Also named the "mean squared error", this loss function measures the squared gaps between the decisions predicted by the actor neural network, $\pi(\mathbf{s}_t^{\mathrm{RH}},\, \boldsymbol{\theta})$, and the decisions $\mathbf{a}_t^{\mathrm{RH}}$ taken for the same states according to $\mathcal{D}^{\mathrm{RH}}$.

The first box of Fig. 3.2 details the actor pretraining. After deriving the state-decision set $\mathcal{D}^{\mathrm{RH}}$, the function $L^{\mathrm{AP}}(\boldsymbol{\theta})$ is minimized by means of gradient descent. First, the state-decision set $\mathcal{D}^{\mathrm{RH}}$ is divided into mini-batches. Second, for each mini-batch, the approximate loss function $\widehat{L^{\mathrm{AP}}}(\boldsymbol{\theta})$ and its gradient $\nabla_\theta \widehat{L^{\mathrm{AP}}}(\boldsymbol{\theta})$ are computed with respect to $\boldsymbol{\theta}$. Third, the parameters $\boldsymbol{\theta}$ are updated by performing one gradient step per mini-batch such that $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha^{\mathrm{AP}} \nabla_{\boldsymbol{\theta}} \widehat{L^{\mathrm{AP}}}(\boldsymbol{\theta})$ where α^{AP} is the learning rate. One training epoch is completed after an update for each mini-batch. The procedure is repeated, making new mini-batches from $\mathcal{D}^{\mathrm{RH}}$ until a total of E^{AP} training epochs are completed. As more training epochs occur, $L^{\mathrm{AP}}(\boldsymbol{\theta})$ converges to a local minimum, and $\pi(\boldsymbol{\theta})$ approximates the rolling horizon strategy described by $\mathcal{D}^{\mathrm{RH}}$.

Critic Pretraining

During critic pretraining, the critic neural network approximates the value function of the policy previously learned by the actor neural network, which is related to the general concept of policy evaluation (see, e.g., Chapter 4 in Sutton & Barto, 2018). To approximate the value function, we minimize the loss function given by

$$L^{\mathrm{CP}}(\boldsymbol{\eta}) = \sum_{t=0}^{T-1} \sum_{\mathbf{s}_t \in \mathcal{S}} \mathbb{P}(\mathbf{s}_t) \Big[\frac{1}{2}\big(\hat{v}_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t,\, \boldsymbol{\eta}) - G_t\big)^2\Big], \tag{3.14}$$

where G_t denotes the total realized profit from state $\mathbf{s}_t$ until the end of the horizon under policy $\pi(\boldsymbol{\theta})$, i.e., $G_t = \sum_{k=t}^{T-1} R_k\big(\mathbf{s}_k, \pi_k(\mathbf{s}_k, \boldsymbol{\theta})\big)$. Eq. (3.14) aims to quantify the difference between the approximate value function $\hat{v}_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t,\, \boldsymbol{\eta})$ and G_t, an unbiased estimate of the true value function $v_{\pi(\boldsymbol{\theta})}(\mathbf{s}_t,\, \boldsymbol{\eta})$ for following policy $\pi(\boldsymbol{\theta})$ at epoch t.

An overview of the critic pretraining is displayed in the second box of Fig. 3.2. First, the pretrained policy function $\pi(\boldsymbol{\theta})$ is simulated in the MDP for T^{CP} decision epochs, and the states and rewards are collected. The collected rewards are then used to compute the value function estimates $(G_t,\, \ldots,\, G_{t+T^{\mathrm{CP}}-1})$ associated with each visited state. Second, the stochastic gradient descent algorithm is executed to minimize Eq. (3.14). That is, for E^{CP} training epochs,

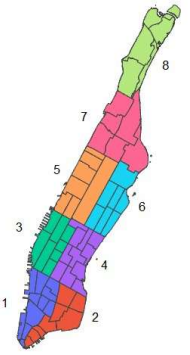


Figure 3.3: Partition of Manhattan into 8 locations.

the visited states and associated value function estimates are shuffled into mini-batches. For each mini-batch, the gradient of the loss function $\nabla_{\boldsymbol{\eta}} \widehat{L^{\mathrm{VF}}}(\boldsymbol{\eta})$ is estimated, and the parameters $\boldsymbol{\eta}$ are updated with learning rate α^{CP}. The collected rewards are finally discarded when E^{CP} epochs are completed. The whole procedure is repeated until a total of I^{CP} iterations have taken place, with parameter I^{CP} determining the accuracy of the value function approximation.

3.5 Numerical Experiments

In this section, we numerically explore the performance of our approach, and collect managerial insights on the resulting policies. Section 3.5.1 presents the base experimental setting of our numerical study. Section 3.5.2 demonstrates the effectiveness of our approach in terms of final performance and computation time. Section 3.5.3 analyzes how the resulting policies employ each of the main decisions: pricing, driver repositioning, and customer admission. Finally, Section 3.5.4 examines the scalability of our approach.

3.5.1 Experimental Setting

Data and assumptions. Our numerical experiments are conducted on data made publicly available by the NYC Taxi and Limousine Commission (2023). The dataset consists of trips using Uber in Manhattan in February 2019, and divides the borough into 64 locations. For each trip, the dataset indicates the date on which the trip occurred, the pickup and drop-off times, the pickup and drop-off locations, and the trip distance. We curate the data by removing trips with incomplete or incoherent features, or trips that originate or end outside Manhattan. Furthermore, trips with incoherent or incomplete features are eliminated, e.g., trips with travel times under 30 seconds or over 60 minutes.

Parameters	C^{lost}	C^{rep}	β^{cons}	β^{price}	β^{trans}	γ	D	$\overline{\Lambda}$
Values	0.4	0.2	0	6	1.5	1	900	1

Table 3.2: Default values of the model parameters.

The base experimental setting includes two simplifications, which allow interpretable results while still capturing the main features of a ride-hailing network. First, similar to Mao et al. (2020), we aggregate adjacent zones to form a network of 8 locations, as illustrated in Fig. 3.3. Second, we limit the study to morning peak hours (7 a.m. - 10 a.m.) on Wednesday, Feb. 6. We divide this horizon into 36 intervals so that a decision epoch occurs every five minutes. The resulting dataset records 18,868 trips. In Section 3.5.4, we consider a larger network size (20 locations) and a longer time horizon (6 hours) to investigate the scalability of our approach.

Parameters. For the horizon considered, we extract the arrival rates of potential customers $\Lambda_{ij,t}$, expected transit times $1/\tau_{ij}$, and distances δ_{ij} from the dataset. To simplify exposition, we assume that the customer willingness to pay $F(\cdot)$ follows a uniform distribution within $[0, 1]$. Drivers in transit are initially distributed according to past expected demand one hour before the start of the horizon, and available drivers are distributed according to future expected demand one hour after the start of the horizon. This follows the view that drivers can serve all customer requests before peak hours while anticipating future demand patterns.

Table 3.2 lists the default values of the remaining parameters. In our experiments, several instances are created by changing one of these values at a time. The lost sales costs, and repositioning cost rate are arbitrarily set to $C^{\text{lost}} = 0.4$, and $C^{\text{rep}} = 0.2$, representing about 20% of the maximum revenue for an average trip. The preference parameters are set to $\beta^{\text{cons}} = 0$, $\beta^{\text{price}} = 6$, and $\beta^{\text{trans}} = 1.5$, to reflect that drivers care more about high earnings than short transit times, as shown in (Lu et al., 2018). The number of drivers in the network is roughly balanced with the demand using Little's law, such that $D = \frac{1}{2T} \sum_t \sum_i \sum_j (\Lambda_{ij,t}/\tau_{ij}) \approx 900$. The parameter $\overline{\Lambda}$ is a constant multiplying each arrival rate of potential customers (i.e., $\Lambda_{ij,t} \leftarrow \overline{\Lambda} \cdot \Lambda_{ij,t}$) to depict various demand conditions.

Experimental setup. Our approach is implemented with the programming language Python, and the open-source library Gym. The Expected Value Problem (EVP) is solved using the solver Gurobi, up to 1% optimality gap, with a maximum running time of 1 minute. The Gurobi solver notices the non-convex structure of EVP, and automatically applies McCormick envelopes to solve the problem efficiently. Following limited manual tuning, our approach has been executed in all instances with the hyperparameters given in Appendix 3.A. The implementation of the Proximal Policy Optimization (PPO) algorithm relies on the package Stable Baselines. During active learning, we evaluated the policy at regular intervals on 300 horizon runs.

If the best average total profit had not changed more than 0.1% over five consecutive policy evaluations, then we considered the algorithm had converged. After deriving the policy, 300 horizon runs were again carried out to estimate all relevant metrics. All the computations were performed with NVIDIA A100 GPU, and 16 GB RAM. All data and codes can be accessed from `https://github.com/ThomasDeMunck12/TRL_RHNetworks`.

3.5.2 Approach Performance

This section aims to demonstrate the efficiency of our approach in terms of computation time and final performance. To this end, we compare our approach with alternative methods, including (i) two applications of the Proximal Policy Optimization (PPO) algorithm without pretraining,[5] and (ii) three rolling horizon (RH) strategies.

The comparison between our approach and these methods is carried out through two metrics. The first metric, named the computation time difference, is expressed as $\%\text{Time(AM)} = \frac{CT^{\text{TL}} - CT^{\text{AM}}}{CT^{\text{AM}}} \times 100\%$, where CT^{TL} and CT^{AM} are the total computation times resulting from our approach, and from one of the alternative methods. The second metric, named the profit difference, is defined as $\%\text{Profit(AM)} = \frac{TP^{\text{TL}} - TP^{\text{AM}}}{TP^{\text{AM}}} \times 100\%$, where TP^{TL} and TP^{AM} represent the average total profits found by our approach, and by one of the alternative methods.

Table 3.3 displays the computation time and profit differences between our approach, the applications of PPO without pretraining, and the RH strategies, for various instances. The instances are generated by varying one value at a time of the default parameters in Table 3.2. The computation time difference of our approach relative to RH strategies is not included, given that these strategies do not compute policies (i.e., mappings from all states to decisions) and thus entail negligible computation times compared with our approach (a few minutes vs. a few hours).

Comparison with PPO without Pretraining

The applications of *PPO without pretraining* use the PPO algorithm and do not pretrain the neural networks beforehand. The hyperparameter configuration is the same as in Appendix 3.A, except for the number of iterations and the learning rate (denoted α^{PPO}) of the PPO algorithm. To ensure algorithm convergence, the maximum number of iterations is increased to 2×10^4 (instead of 6×10^3). Two learning rates α^{PPO} are considered: $\alpha^{\text{PPO}} = 3 \times 10^{-4}$ as recommended by Schulman et al. (2017), and $\alpha^{\text{PPO}} = 10^{-4}$, which shows to be better suited to our problem.

[5] Our approach was also compared to other DRL algorithms such as DDPG, TRPO, and TD3. The comparisons are not included in the experiments, given their notably worse performances.

	Our Approach		vs. PPO without Pretraining				vs. RH Strategies		
	Time	Profit	%Time		%Profit		%Profit		
	α^{PPO}		α^{PPO}	α^{PPO}	α^{PPO}	α^{PPO}	$T^{\text{RH}} = 36$	$T^{\text{RH}} = 4$	$T^{\text{RH}} = 1$
Instances	$= 6 \times 10^{-5}$		$= 3 \times 10^{-4}$	$= 10^{-4}$	$= 3 \times 10^{-4}$	$= 10^{-4}$	$Q^{\text{RH}} = 36$	$Q^{\text{RH}} = 12$	$Q^{\text{RH}} = 12$
$\overline{\Lambda} = 0.5$	1880	5576	−35%	−66%	36.6%	55.8%	5.9%	0.1%	0.2%
$\overline{\Lambda} = 0.75$	1529	7917	−63%	−78%	25.7%	82.6%	6.1%	1.4%	1.2%
$\boldsymbol{\overline{\Lambda} = 1.0}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$\overline{\Lambda} = 1.25$	1619	11510	−67%	−56%	19.7%	*	19.2%	3.7%	2.6%
$\overline{\Lambda} = 1.5$	1911	12626	−34%	−73%	17.9%	7.3%	29.7%	3.1%	1.8%
$D = 450$	1015	6861	−69%	−59%	52.1%	8.3%	23.9%	2.6%	1.5%
$D = 675$	1438	8850	−59%	−61%	21%	10.2%	23.8%	4.1%	2.8%
$\boldsymbol{D = 900}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$D = 1125$	1931	10495	−43%	−79%	26.1%	8.5%	6.4%	1.6%	1.2%
$D = 1350$	2701	10862	−43%	−75%	21.6%	12.9%	4.7%	0%	−0.1%
$\gamma = 0.25$	2010	9966	−45%	−70%	20.8%	9.5%	13.4%	1.2%	0.4%
$\gamma = 0.5$	1909	9995	−18%	−68%	29%	36.7%	13.8%	1.4%	0.8%
$\boldsymbol{\gamma = 1}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$\gamma = 2$	2038	7970	−52%	−58%	2.6%	−1.2%	19.5%	16.6%	13.3%
$\gamma = 4$	1929	3463	110%	9%	−34%	*	51.7%	45.6%	33.9%
$\beta^{\text{trans}} = -1$	1649	10002	−47%	−80%	20.3%	10.9%	14.6%	1.9%	1.1%
$\beta^{\text{trans}} = -2$	1990	9347	−37%	−60%	17%	5.2%	37.8%	10.7%	8.6%
$\boldsymbol{\beta^{\text{trans}} = -1.5}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$\beta^{\text{cons}} = 3$, $\beta^{\text{price}} = 0$	1721	9837	−38%	−72%	30.6%	9.7%	18.9%	4.5%	3.4%
$\beta^{\text{cons}} = -5$, $\beta^{\text{trans}} = 0$	1781	9923	−59%	−71%	34.3%	9%	13.1%	4.1%	2.8%
$C^{\text{lost}} = 0$	1711	9946	−45%	−72%	18.6%	8%	7.3%	0.7%	0.2%
$C^{\text{lost}} = 0.2$	1949	9926	−49%	−59%	20%	6.3%	10.9%	1.9%	1.1%
$\boldsymbol{C^{\text{lost}} = 0.4}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$C^{\text{lost}} = 0.8$	1924	9911	−63%	−55%	26.7%	*	50.8%	6.1%	4.4%
$C^{\text{lost}} = 1.2$	1853	9828	−75%	−68%	53.3%	20.34%	146.1%	7.9%	5.6%
$C^{\text{rep}} = 0$	1857	10299	−34%	−67%	13.1%	5.9%	14.7%	3.8%	2.9%
$C^{\text{rep}} = 0.1$	1822	10074	−40%	−70%	14.8%	4.5%	11.9%	1.2%	0.3%
$\boldsymbol{C^{\text{rep}} = 0.2}$	**2039**	**9978**	**−42%**	**−72%**	**22.7%**	**10.2%**	**15.2%**	**3.9%**	**2.8%**
$C^{\text{rep}} = 0.3$	1989	9916	−60%	−77%	48.8%	39.7%	15.7%	4.3%	3.1%
$C^{\text{rep}} = 0.4$	1913	9872	−40%	−76%	25.4%	9.4%	19.4%	4.2%	3%
Mean	**1844**	**9390**	**−42%**	**−65%**	**23.4%**	**16.7%**	**23.5%**	**5.5%**	**4.0%**
Median	1909	9916	−45%	−70%	21.6%	9.2%	15.2%	3.7%	2.6%

Table 3.3: Performance of our approach across different problem instances. The reported values include the average computation time and profit, in absolute terms (in minutes and euros, respectively) and relative to PPO without pretraining and RH strategies (in percentage). The default instance and the average values for all the instances are written in bold. If PPO without pretraining yields negative profits, the profit difference is replaced by "*".

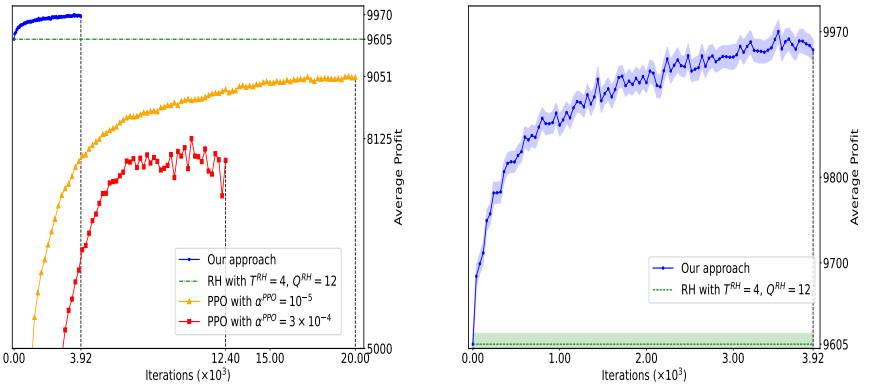
In Table 3.3, the benefits of pretraining appear clearly as our approach improves profit by at least 16.7% on average. When $\alpha^{\text{PPO}} = 10^{-4}$, PPO without pretraining obtains a lower profit in around double the computation time than PPO with pretraining (see the computation time difference of -65%). When $\alpha^{\text{PPO}} = 3 \times 10^{-4}$, PPO without pretraining becomes faster but still slower than with pretraining (the average computation time difference going from -65% and -42%), while the final performance deteriorates (the average profit difference going from 16.7% to 23.4%). The fact that our approach outperforms PPO without pretraining can be further visualized in the learning curves in Fig. 3.4. In Fig. 3.4(a), we note that the PPO algorithm employed by our approach consistently outperforms PPO algorithms without pretraining. In addition, Fig. 3.4(b) shows that the PPO algorithm gradually improves upon the pretrained policy provided by our approach (i.e., the RH strategy with $T^{\text{RH}} = 4$ and $Q^{\text{RH}} = 12$).

Concerning the computation time difference, its variations between instances (usually ranging from -80% to -20%) reflect that the total training times are more stable with our approach than with PPO without pretraining. To further quantify these results, we measured the standard deviation of the computation times between instances. We observed a standard deviation that is 58% ($\alpha^{\text{PPO}} = 3 \times 10^{-4}$) and 156% ($\alpha^{\text{PPO}} = 10^{-4}$) higher for PPO without pretraining than for our approach. One possible reason for this result is that the pretraining process trains the neural networks to learn a unique policy and value function. In contrast, PPO updates the policy and value function many times by trial and error, making the learning process more volatile.

Concerning the profit difference, Table 3.3 shows that PPO with pretraining leads to profit improvement for 23 out of 25 instances (for both PPO parameter configurations). Moreover, our approach always generates positive profits, an outcome not always reached by PPO without pretraining (cf. the (*) symbols in Table 3.3). This result further supports the idea that our approach benefits from a more stable learning process. The only two cases where our approach performs worse than PPO without pretraining occur when the temperature parameter γ is high, i.e., when many drivers elect to self-relocate. As RH strategies ignore self-relocating drivers, they prescribe inadequate decisions (cf. the corresponding performances of RH strategies in Table 3.3) so that pretraining becomes counterproductive and delays learning.

Comparison with RH Strategies

As discussed in Section 3.4.2, *RH strategies* can solve the EVP with different optimization intervals T^{RH} and optimization horizons Q^{RH}. Therefore, we benchmark our approach against three RH strategies. The first RH strategy produces static decisions by solving the EVP only at the first decision epoch ($T^{\text{RH}} = 36$) for the entire horizon ($Q^{\text{RH}} = 36$); the second RH strategy, which is used for actor pretraining, generates more dynamic decisions by solving the EVP every 4 epochs ($T^{\text{RH}} = 4$) over a horizon of 12 epochs ($Q^{\text{RH}} = 12$); the



(a) (b)

Figure 3.4: Learning curves of PPO with and without pretraining, compared to the ride-hailing (RH) strategy with $T^{\mathrm{RH}} = 4, Q^{\mathrm{RH}} = 12$. The figure on the left-hand side (a) shows the evolution over all the training iterations. The figure on the right-hand side (b) zooms into the early training phase of pretrained PPO, compared to the RH strategy (b). The shaded areas indicate 95% confidence intervals.

third RH strategy yields fully dynamic decisions by solving the EVP every epoch ($T^{\mathrm{RH}} = 1$) over a horizon of 12 epochs ($Q^{\mathrm{RH}} = 12$). Computational efforts grow from the first to the third RH strategy, due to the decrease of the optimization interval T^{RH}.

Table 3.3 notably indicates that our approach results in profits that are 4% higher on average than those of the best performing RH strategy ($T^{\mathrm{RH}} = 1$, and $Q^{\mathrm{RH}} = 12$). This RH strategy is the most dynamic one as it adapts its decisions at each epoch as a function of the state. It appears as a near-optimal baseline that is not easy to outperform. However, contrary to our approach, it solely relies on expected values to make its decisions. The 4% profit improvement hence reveals the benefits of taking stochastic variations into account, and the ability of our approach to seize them. As the impact of uncertainty increases, our approach yields stronger performance relative to the RH strategy (e.g., when the lost sales cost C^{lost} are high)

Comparing our approach to the two other RH strategies, we observe that the profit difference grows even more on average (23.5% when $T^{\mathrm{RH}} = 1$, $Q^{\mathrm{RH}} = 36$; 5.5% when $T^{\mathrm{RH}} = 4$, $Q^{\mathrm{RH}} = 12$). This result shows the value of our approach in generating a dynamic policy, in opposition to more static RH strategies. It also confirms that our approach effectively uses PPO to improve upon the pretrained policy (a replication of the RH strategy with $T^{\mathrm{RH}} = 4$, $Q^{\mathrm{RH}} = 12$), although the pretrained policy already performs better than PPO alone (cf. Fig. 3.4(b)).

As a final comment, note that our approach generates a policy mapping all states to decisions. As such, it leads to an output that can readily be executed once the training is over, so that no computation occurs during the actual operations (though decisions are still based on the system states). In contrast, RH strategies only seek decisions for a few states over some decision epochs. They need to periodically solve an optimization problem during operations, with limited computation time to make decisions. For this reason, we expect the performance of RH strategies to deteriorate in more complex settings. Our approach extends to larger applications, as shown in Section 3.5.4.

3.5.3 Policy Analysis

We now analyze the policies found by our approach to comprehend the use of pricing, driver repositioning, and customer admission. For each decision, we mainly look at the effect on the flow of customers and drivers entering (inflow) or leaving (outflow) each location. The net flow, which is the difference between total inflow and total outflow over the horizon, offers a simple measure of the imbalances in a location. A positive net flow indicates that more customers or drivers are entering the location than leaving it, whereas a negative net flow means the opposite. The absolute sum of the net flows in all the locations characterizes the total network imbalances. By comparing the net flows of the potential demand with the net flows of admitted customers, served customers, and dispatched and repositioned drivers, we can quantify the impact of each decision on the network imbalances.

Pricing Decision

For the default instance, Table 3.4 shows the average net flows of customer requests for each location under the policy resulting from our approach, and under a policy that sets a single price $p = 0.59$ that corresponds to the average price found by our policy over the 8 locations, weighted as a function of demand in each location. The table also displays the average price prescribed by our policy in each location. In Table 3.4, we observe that the net flows of customer requests decrease substantially due to the pricing decision. In the whole network, the pricing decision reduces imbalances, with the absolute sum of net flows falling by roughly 25%. This observation illustrates the *spatial*

Customer requests	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6	Loc. 7	Loc. 8	Abs. sum
With Single Pricing	-42	-289	177	901	-376	-298	-64	-7	2156
With Pricing Decision	27	-187	136	631	-299	-225	-69	-15	1589
Average Price	0.62	0.65	0.57	0.52	0.64	0.62	0.59	0.56	

Table 3.4: Average net flows of customer requests with single pricing ($p = 0.59$), and with the pricing decision obtained by our approach, in each location, for the default instance. The last row indicates the average prices decided in each location.

	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6	Loc. 7	Loc. 8	Abs. Sum
Dispatched Drivers	23	-174	114	583	-272	-207	-61	-7	1441
Repositioned Drivers	8	109	-61	-349	148	115	32	-2	824
Self-Relocating Drivers	-3	72	-32	-194	83	42	23	8	457

Table 3.5: Average net flows of dispatched, repositioned, and self-relocating drivers, in each location for the default instance. The last column gives the absolute sum of the average net flows over the network.

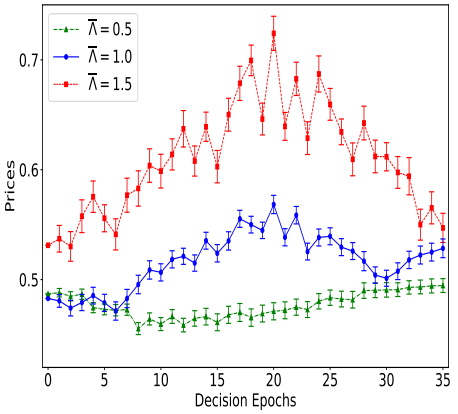
role of the pricing decision in balancing demand between network locations. As shown in Table 3.4, the average prices are higher in locations with negative net flows and lower in locations with positive net flows. Consequently, more requests start from locations with positive net flows and end in locations with negative net flows, thereby reducing demand imbalances.

Besides its spatial role, the pricing decision also has a *temporal role*. Fig. 3.5 illustrates this role by presenting the evolution over time of the arrival rate of potential customers (a), the average price (b), and the arrival rate of customer requests (c) in the fourth location (the location with the largest demand imbalances), for various magnitudes of demand. In Fig. 3.5(a), we note that a higher level of demand leads to more pronounced demand peaks. In response, prices are adjusted more dynamically over time so as to follow the demand peaks (cf. Fig. 3.5(b)). As a result, stable arrival rates of customer requests are obtained throughout the horizon (cf. Fig. 3.5(c)), regardless of the demand magnitude. This allows the platform to avoid temporary shortages of drivers, meaning that actual customer requests have a higher likelihood of being served at any time.

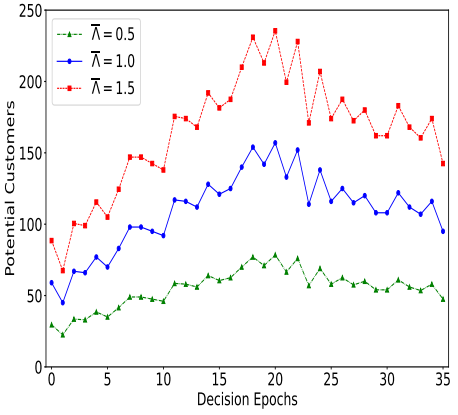
Looking closely at Fig. 3.5(b), it can be noticed that the 95% prediction price intervals (between brackets) grow with the arrival rates of potential customers (0.13 when $\overline{\Lambda} = 0.5$ vs. 0.29 when $\overline{\Lambda} = 1.5$, on average). This variation of prices between horizon runs suggests that the pricing decision may exert a *state-dependent role*. That is, prices are adapted as a function of the observed state (i.e., the number of available drivers) to fit demand with supply in real time. If the number of drivers in a location is lower than expected, then the price is raised to discourage customer requests, and conversely. As a result, the number of available drivers becomes more stable, protecting the platform against stochastic fluctuations. This effect gains in importance with high demand, as drivers become scarcer.

Repositioning Decision

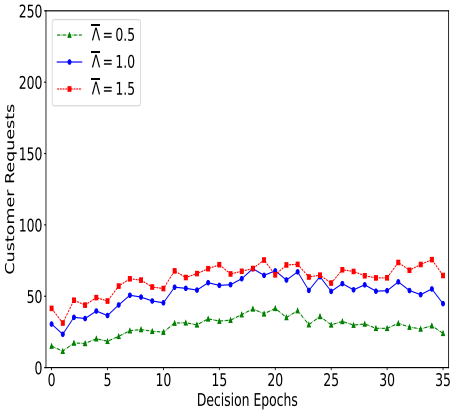
We investigate the repositioning decision by looking at Table 3.5, which shows, once again for the default instance, the average net flows of dispatched drivers, repositioned drivers, and self-relocating drivers for each location. We note that the net flows of dispatched drivers present significant imbalances, as illustrated by the absolute sum of net flows being equal to 1441. These imbalances, compa-



(a)

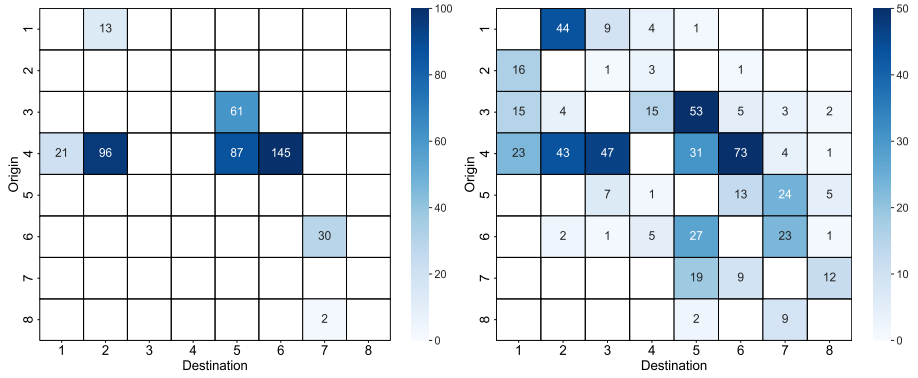


(b)



(c)

Figure 3.5: Evolution of the average prices (the brackets around the average prices represent 95% prediction intervals over 300 horizon runs) (a), the arrival rates of potential customers (b), and the arrival rates of customer requests (c), in the fourth location, for different arrival rates of potential customers.



(a) (b)

Figure 3.6: Average number of repositioned (a) and self-relocating drivers (b) over the time horizon for each origin-destination pair, for the default instance. All the values are rounded down to the nearest integer. Values rounded to 0 are omitted for conciseness.

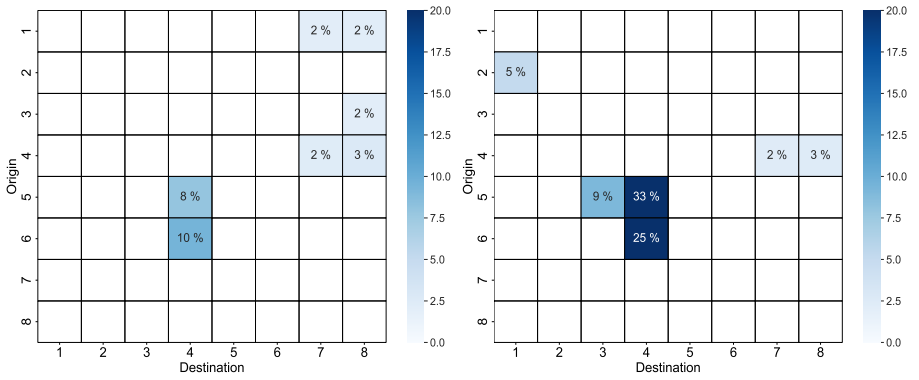
rable to those of customer requests (cf. Table 3.4), can cause excessive or scarce driver supply in some locations. Driver repositioning and driver self-relocating are crucial for rebalancing the network, sending drivers from locations with sufficient supply to locations with limited supply. For instance, in the fourth location, the number of dispatched drivers arriving exceeds the number of dispatched drivers departing by 583 (cf. Table 3.5). To offset this imbalance, our policy induces an average of 394 drivers to reposition and 194 drivers to self-relocate from this location. This permits the platform to satisfy 96% of the requests on average.

Interestingly, Table 3.5 reports that the net flows of repositioned drivers are often higher than those of self-relocating drivers. This result suggests that the repositioning decision is more effective in addressing network imbalances than driver self-relocating. We explain this result by the fact that the repositioning decision allows for a more immediate way to rebalance driver supply than driver self-relocation. Despite being costly, driver repositioning is directly controlled by the platform. In contrast, driver self-relocating is indirectly controlled by the pricing decision (which also affects customer demand) and depends on factors beyond the platform's control (e.g., the distance between locations).

Fig. 3.6 further elaborates the previous insight by plotting the average number of repositioned (a) and self-relocating drivers (b) for each pair of locations. In Fig. 3.6 (a), drivers are repositioned only for specific pairs of locations, from locations with an excess of supply (e.g., the third and fourth locations) to locations with a shortage of supply (e.g., the fifth and sixth locations). Conversely, in Fig. 3.6 (b), drivers self-relocate for many more origin-destination pairs, and even from locations with scarce supply to locations with excessive

	Loc. 1	Loc. 2	Loc. 3	Loc. 4	Loc. 5	Loc. 6	Loc. 7	Loc. 8	Abs. sum
Customer Requests	27	-187	136	631	-299	-225	-69	-15	1589
Admitted Cust. Req.	24	-181	135	612	-296	-208	-71	-16	1543

Table 3.6: Average net flows of customer requests, and admitted customer requests, in each location for the default instance. The last column gives the absolute sum of the average net flows over the network.



(a) (b)

Figure 3.7: Average percentage of rejected customers for each origin-destination pair of locations when $\overline{\Lambda} = 1.5$ (a), and when $D = 450$ (b). For conciseness, values equal to 0% are omitted.

supply. In total, more drivers (562) self-relocate than are repositioned (455). We thus conclude that the repositioning decision rebalances the network more accurately than driver self-relocating, while involving fewer drivers.

Admission Decision

To examine the admission decision, Table 3.6 displays the average net flows of customer requests and admitted customer requests in each location for the default instance. We observe that the admission decision reduces the absolute sum of net flows from 1589 to 1543 (a fall of about 3%). Overall, the admission decision has a limited impact on the customer net flows and demand imbalances, contrary to the pricing decision. The reluctance to reject customers arises from the fact that doing so generates both a cost and a loss of revenue. In contrast, higher prices prevent some customers from entering the system at no cost while increasing revenues from remaining customer requests.

When the number of requests is excessive, the admission decision can still be relevant since it allows for rejecting requests according to their origin and destination (contrary to prices that only depend on the customer origin). To illustrate this, Fig. 3.7 plots the average percentage of rejected customers for

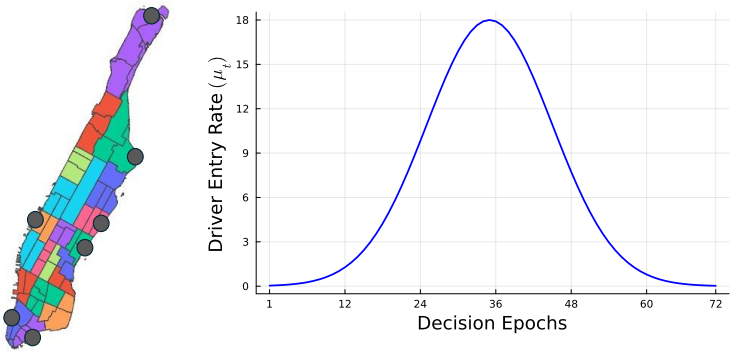
each pair of locations when customer demand is high ($\overline{\Lambda} = 1.5$) (a), or when driver supply is low ($D = 450$) (b). We see that the admission decision avoids sending drivers from locations with scarce supply (e.g., the fifth location) to locations with excessive supply (e.g., the fourth location), permitting finer control than prices. Because drivers are busy serving customers most of the time (e.g., the driver utilization rate is equal to 93% when $D = 450$), few drivers are available for repositioning. Thus, the admission decision acts as a substitute for driver repositioning, rejecting customers going to undesirable locations.

3.5.4 Extension to Large-Scale Applications

In this section, we examine the performance of our approach in a larger setting. We do so by making four modifications to the base experimental settings of Section 3.5.1. First, the ride-hailing network is extended to 20 locations, illustrated in Fig. 3.8(a). Second, the horizon is broadened to 72 decision epochs, occurring every five minutes from 6 a.m. to 12 p.m. Third, we relax the assumption that there is a constant number of drivers in the network. Instead, only half of the drivers are operating at the beginning of the horizon, and the rest of the drivers enter the network according to a non-stationary Poisson process with rate μ_t, as shown in Fig. 3.8(b). The random number of drivers in the network at epoch t is denoted by D_t. Assuming that drivers come mainly from other boroughs than Manhattan, drivers are only allowed to enter the network from seven entry points (cf. the gray dots in Fig. 3.8(a)), which correspond to bridges linking Manhattan to the remaining of the city. Fourth, since real-world applications are typically characterized by variable customer arrival rates, our approach first uses training data from 20 weekdays of February 2019 to learn a policy. Then, test data from 20 weekdays of March 2019 are used to evaluate the policy performance with new arrival rates that are unknown at the time of training. For each test day, 50 horizon runs are simulated with random customer requests generated from the corresponding arrival rates to evaluate the performance of the learned policy.

The analysis consists of comparing our reinforcement learning approach with the best rolling horizon (RH) strategy from Section 3.5.2 (with $T^{\text{RH}} = 1$, $Q^{\text{RH}} = 12$). Given the extensive size of the problem, the PPO algorithm does not reach sufficient performance without pretraining and is excluded from the comparison.[6] To quantify the benefits of our approach, we rely on three metrics. The first metric is the total profit of the platform over the whole horizon, $\text{TP} = \sum_t R_t(\mathbf{s_t}, \mathbf{a_t})$. The second metric is the driver utilization rate $\text{UR} = \frac{1}{T}\sum_t \sum_i \sum_j (d^{\text{dis}}_{ij,t}/D_t \cdot \tau_{ij})$, which evaluates the proportion of time that drivers are busy serving customers. The third metric is the request fulfillment rate $\text{FR} = \frac{1}{T}\sum_t (\sum_i \sum_j d^{\text{dis}}_{ij,t})/(\sum_i \sum_j c^{\text{req}}_{ij,t})$, which assesses the proportion of requests that are effectively satisfied.

[6] The PPO algorithm was implemented with the same set of hyperparameters as in Section 3.5.2 (and $\alpha^{\text{PPO}} = 10^{-4}$), leading to an average profit 53% lower after a training time more than twice as long, in comparison with our approach.



(a) (b)

Figure 3.8: Partition of Manhattan into 20 locations with driver entry points described by gray dots (a). Aggregate rate μ_t of drivers entering the network at each decision epoch (b).

Fig. 3.9 displays the total profit for our approach and the RH strategy for each test day. The Whisker plots summarize the percentiles and average values obtained from the 50 horizon runs that evaluate the learned policy. We observe that our approach generates substantially higher profits than the RH strategy, increasing the profit by 6% on average. This result demonstrates the scalability of our approach, as the latter can be applied to larger settings with novel forms of system dynamics. These profit gains are generally more significant than those reported in Section 3.5.2, where the same parameters were used for training and testing. One probable reason for this improved performance lies in the ability of our approach to adapt to new situations. By leveraging DRL, our approach learns a policy that generalizes to new states not encountered in the training data (e.g., days with higher demand). In contrast, the RH strategy relies solely on expected values from historical data to make decisions. When test data significantly differ from data previously observed, this dependence causes the RH strategy to make inappropriate decisions, resulting in profit losses.

Fig. 3.10 complements the analysis by plotting the average driver utilization rate and request fulfillment rate for our approach and the RH strategy for each test day. Fig. 3.10 shows that our approach can benefit drivers and customers besides maximizing profit. On the one hand, driver utilization rates are consistently higher with our approach than with the RH strategy (5% on average). This result is desirable as it reveals that drivers spend less time idle, improving their earnings. It also suggests that more customers are served in total. On the other hand, the request fulfillment rate is raised, compared to the RH strategy (2% on average). That is, customer requests are more likely to be satisfied with our approach. The only cases where our approach has a lower fulfillment rate than the RH strategy (i.e., days 11, 12, 16) occur during low-demand days. However, for these days, the fulfillment rate is already high, and the difference

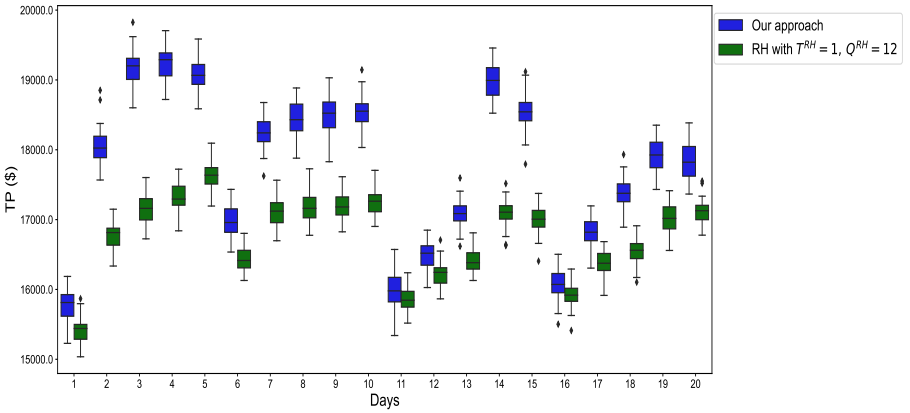


Figure 3.9: Comparison of the total profit (TP) between our approach and the RH strategy ($T^{\mathrm{RH}} = 1,\ Q^{\mathrm{RH}} = 12$) over 20 test days.

between our approach and the RH strategy remains marginal.

3.6 Conclusion

This chapter considers a ride-hailing platform that balances driver supply with customer demand over multiple locations. Impatient customers arrive with distinct willingness to pay and decide whether to request services based on prices. Drivers can elect to self-relocate between locations according to their preferences regarding prices and transit times. In this broad setting, the platform makes pricing, customer admission, and driver repositioning decisions.

We formulate the problem as a Markov decision process that determines the price in each location, and the number of repositioned drivers and admitted customers between locations. Given the problem complexity, we develop a novel approach that combines mathematical optimization with deep reinforcement learning. Our approach first designs a rolling horizon (RH) strategy that makes decisions based on an expected-value formulation of the problem. Then, two neural networks are pretrained to reproduce the decisions prescribed by the RH strategy, and approximate the value function of the associated policy. Finally, the Proximal Policy Optimization (PPO) algorithm is applied to further improve the policy.

Using data from New York City, we demonstrate that our approach leads to a stable learning process characterized by improved performance. On average, our approach finds a policy that increases the platform's profit by 16.7% while reducing the total training time by $42\% - 65\%$ in comparison with the PPO algorithm without pretraining. Moreover, our approach outperforms RH strategies by 4% at least, while the latter are known to work well in practice. Our approach is also shown to generalize to settings with many locations, many

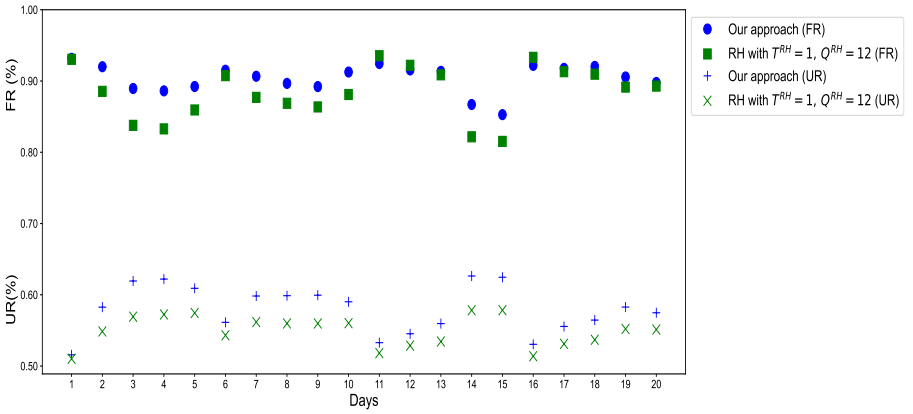


Figure 3.10: Comparison of the average driver utilization rate (UR), and request fulfillment rate (FR) between our approach and the RH strategy ($T^{\mathrm{RH}} = 1$, $Q^{\mathrm{RH}} = 12$) over 20 test days.

decision epochs, and changing demand parameters. Overall, this chapter contributes to the literature by successfully applying a pretraining procedure to a complex problem. We hope to inspire more research that leverages mathematical optimization with DRL in the future.

By analyzing the resulting policy, we explore managerial insights on pricing, driver repositioning, and customer admission. The main insights are summarized as follows:

(i) The pricing decision plays three roles in balancing customer demand and driver supply. First, it adjusts prices spatially to balance the flows of customer requests between locations. Second, it adjusts prices temporally to stabilize the number of customer requests in each location over time. Third, to a lesser extent, it adjusts prices as a function of the system state to match requests with drivers in real time.

(ii) The repositioning decision can effectively mitigate supply-demand imbalances. Furthermore, it appears more effective than driver self-relocating because it allows the platform to control the driver rebalancing fully. In contrast, driver self-relocating is indirectly induced by the pricing decision and depends on the transit times between locations.

(iii) In general, the use of the admission decision is limited. Nevertheless, when driver supply is scarce, customer admission becomes relevant. On the one hand, it balances the network flows more finely than the pricing decision. On the other hand, it serves as a way to indirectly reposition drivers when the latter are too busy to be directly repositioned.

The present chapter leaves several avenues for further research. First, for the sake of tractability, we made several assumptions about the problem, such

as drivers always complying with the platform's repositioning decision and transit times that follow exponential distributions. It would be interesting (yet challenging) to relax these assumptions and explore their related implications. Second, our model only makes repositioning and pricing decisions, leaving additional decisions to other optimization models. In practice, however, decisions with different timings interact with each other. A challenge to undertake would be to develop a model integrating decisions occurring over distinct time scales. Finally, this chapter applies our approach to a ride-hailing network. Nevertheless, we believe the approach is sufficiently general for other contexts where heuristics are available (e.g., inventory management and vehicle routing). It would be worth adapting our approach to settings where DRL algorithms can be pretrained from existing heuristics.

Appendix 3.A Hyperparameters

Hyperparameters	Values
Neural networks	
Hidden layers	4
Nodes per hidden layer	128
Activation function	tanh
Proximal Policy Optimization	
Simulation horizon	2400
Mini-batch size	400
Learning rate (α^{PPO})	5×10^{-6}
Clipping range (ε)	0.2
Training epochs	10
Iterations between evaluations	50
Horizon runs during evaluation	500
Minimum number of iterations	2000
Maximum number of iterations	6000
Rolling horizon strategy	
Optimization interval (T^{RH})	4
Optimization horizon (Q^{RH})	12
Cardinality of the state-decision set ($\lvert\mathcal{D}^{\text{RH}}\rvert$)	10^5
Actor pretraining	
Mini-batch size	400
Learning rate (α^{AP})	10^{-4}
Training epochs (E^{AP})	100
Critic pretraining	
Simulation horizon (T^{CP})	2400
Mini-batch size	400
Learning rate (α^{CP})	2×10^{-3}
Training epochs (E^{CP})	10
Number of iterations (I^{CP})	150

Table 3.7: Values of the hyperparameters of the reinforcement learning approach.

4 Ride-Hailing Dispatching with Public Transit Integration

De Munck, T., Tancrez, J.-S., and Vera, J. On the benefits of coordinating ride-hailing platforms with public transit systems. Working paper, 2025 (to be submitted).

4.1 Introduction

Since their foundation, ride-hailing platforms (e.g., Uber, Lyft) have experienced remarkable growth and are now essential components of urban mobility. At the core of their success, these platforms offer convenient services, allowing customers to request rides from their current location to any destination of their choice. Yet, despite their widespread adoption, ride-hailing platforms have also faced strong criticism. Namely, one issue concerns their substitutive effect on public transit (e.g., Shi et al., 2021, Erhardt et al., 2022). The resulting decline in public transit ridership can have significant implications for urban mobility, including increased traffic congestion (Agarwal et al., 2023) and financial struggles for public transit agencies (Lee, 2023).

Conversely, some studies indicate that ride-hailing platforms and public transit can, in fact, complement each other (see, e.g., Hall et al., 2018, Babar & Burtch, 2020). In practice, there exist several illustrations of integrations between ride-hailing platforms and public transit systems. For instance, Uber and Lyft offer discounts to customers going to public transit stations in several cities worldwide (see Uber, 2019, Lyft, 2024). In the same vein, Via and Uber cooperate with local authorities to replace fixed bus lines with on-demand services in areas characterized by low density (see Via, 2018, Uber, 2024a). There are several reasons to justify such collaborations. On the one hand, the platform can serve as a first-mile and last-mile connector to public transit, hence attracting additional transit users from remote areas. On the other hand, the public transit system can improve the platform's operations by providing more flexibility in the matching process. Customers can also benefit from a more accessible and affordable service. Furthermore, the higher incentives for travelers to use public transit can alleviate traffic congestion and carbon emissions.

Although it may be valuable, the integration of ride-hailing platforms and public transit systems remains a challenge in practice. At the strategic level,

it is often unclear when such an integration is beneficial. Similar to other on-demand transport services (e.g., on-demand bus systems, Vansteenwegen et al., 2022), the magnitude of the benefits can be affected by factors such as the public transit configuration and the spatio-temporal pattern of supply and demand. Therefore, a careful evaluation of the potential benefits is highly valuable prior to any implementation. At the operational level, efficient decision-making tools are indispensable to support the integration between the two entities (H. Wang & Yang, 2019). However, these tools are typically hard to develop. Since demand is uncertain and impatient, decisions must be made over short periods of time. At a city scale, thousands (if not millions) of decisions are required daily, further complicating the problem.

In this paper, we propose an operational tool to adapt the dispatching process on a ride-hailing platform to existing public transit. We take the perspective of a ride-hailing platform operating in a network characterized by multiple locations and public transit lines. The transit lines are fixed and operate with a periodic timetable that is known to the platform. To optimize profit and customer travel time, the platform offers three types of services to its customers: door-to-door, first-mile, and last-mile services. Door-to-door service transports customers from origin to destination, without resorting to public transit. First-mile service brings customers from their origin to a transit station, leaving the rest of the trip to public transit. Last-mile service refers to the opposite case, where customers first use public transit and then receive service from a transit station to their destination. Customers are assumed to accept the service recommended by the platform if the service satisfies several quality constraints with respect to travel times, and service fees. In this setting, the platform decides in real time (i) which services to provide to the customer requests, (ii) which transit stations to use for first/last-mile connection, and (iii) which drivers to dispatch.

Given the dynamic and stochastic nature of our problem, we develop a lookahead approach based on stochastic optimization to guide the platform's decisions. At regular time intervals, the approach solves a two-stage stochastic program that accounts for the current system state, and future (uncertain) demand. To improve the approach tractability, we exploit the structure of our problem in two ways. First, a preprocessing procedure is developed to reduce the model size. Second, the L-shaped method (Birge & Louveaux, 2011) is adapted to enhance the model resolution. In numerical experiments, we demonstrate the relevance of our approach by showing how solution quality and resolution time can be balanced, and by comparing it with deterministic lookahead and myopic rolling horizon approaches.

To evaluate the value of integrating public transit transfers into ride-hailing dispatching, the proposed approach is applied to a broad range of synthetic instances. We explore the tradeoff between two common ride-hailing objectives: maximizing profit and minimizing customer travel time. The analysis reveals that these objectives lead to distinct solution structures and can have a significant impact on customer service. Additionally, the effects of demand, supply,

and public transit changes are examined to assess the benefits of coordination. Lastly, a case study based on real-world data from New York City is considered. The integrated system is found to improve profit by up to 7.3%, reduce customer travel time by up to 5.5%, and increase the percentage of fulfilled requests by up to 17.9% compared to the setting without integration.

To summarize, this research makes three main contributions:

- From a modeling perspective, we study an online problem in which a ride-hailing platform takes advantage of the existing public transit system to improve its dispatching process, at the scale of a city. The problem involves demand uncertainty, first-mile and last-mile services, and multiple transit stations and lines.

- From a methodological perspective, we develop a lookahead approach to support the dispatching process. The approach relies on stochastic optimization to make decisions in a dynamic and proactive way, rather than in a static or reactive way. The approach can dispatch hundreds of drivers to thousands of customer requests within a reasonable computation time (i.e., less than 3 minutes).

- From a managerial perspective, we perform extensive numerical experiments to collect meaningful insights on the benefits of an integrated system. In particular, we apply our approach both to synthetic and real-world data in order to evaluate the benefits under profit maximization and travel time minimization.

The rest of this paper is structured as follows. Section 4.2 reviews relevant literature. Section 4.3 introduces the online dispatching problem of a ride-hailing platform integrating public transit transfers. Section 4.4 studies its offline counterpart, formulating it as an integer program. Section 4.5 develops the lookahead approach. Section 4.6 conducts numerical experiments to study the approach performance and evaluate the potential benefits of the integration. Finally, Section 4.7 concludes.

4.2 Literature Review

Our work is related to the literature on ride-hailing platforms, and the literature on the integration of on-demand transport and public transit systems. In this section, we review these two literature streams, underlining our distinct research features.

Reviewed by H. Wang and Yang (2019), the recent literature on ride-hailing platforms addresses various operational problems. Several studies use stylized models to explore the implications of pricing on supply and demand (e.g., Cachon et al., 2017, Bai et al., 2019), while others examine its spatial and temporal role in balancing supply across networks (e.g., C. Chen et al., 2021,

Lei & Ukkusuri, 2023, De Munck et al., 2024). Besides pricing, driver repositioning is also considered when demand and supply are spatially distributed. Some research examines driver repositioning as a centralized process, entirely controlled by the platform (e.g., Braverman et al., 2019, Hosseini et al., 2024, and Beojone et al., 2024) while others consider repositioning as a decentralized process in which drivers choose where to relocate (e.g., Shou & Di, 2020, Zhu, Ke, & Wang, 2021, and Beojone & Geroliminis, 2023).

Closer to our research, the problem of dispatching drivers to customer requests has also garnered significant attention in the literature on ride-hailing platforms. Several papers analytically explore the structure of their specific problem to obtain dispatching policies that are provably near-optimal under simplified settings (e.g., Özkan & Ward, 2020 and M. Hu & Zhou, 2022). To tackle more realistic problems, multiple studies resort to reinforcement learning algorithms (e.g., Al-Kanj et al., 2020, Mao et al., 2020, Turan et al., 2020, and Azagirre et al., 2024; see Qin et al., 2022 for a review). Similar to us, some works use approaches from mathematical programming to dispatch drivers dynamically over time. Namely, Alonso-Mora et al. (2017) and Simonetto et al. (2019) repeatedly solve a reactive integer model to match drivers with pooled customers in real time. In a related setting, Bertsimas et al. (2019) incorporate time windows to customer requests, formulating the resulting problem as a mixed-integer program and solving it using a backbone algorithm. Lowalekar et al. (2018) capture future demand uncertainty through a multistage stochastic program. All these papers consider a ride-hailing platform that operates in isolation, providing only door-to-door services to customers. Our work complements this research by providing an approach to integrate public transit transfers into the dispatching process of a ride-hailing platform.

Several papers examine the relationship between ride-hailing and public transit systems. A recurring question concerns whether ride-hailing platforms complement or substitute public transit systems. To answer this question, multiple studies empirically analyze the effects of Uber on transit ridership. For instance, Hall et al. (2018) find that Uber complements public transit in large cities with limited transit networks but acts as a substitute in smaller cities. Babar and Burtch (2020) report that Uber reduces bus ridership while increasing rail ridership. Erhardt et al. (2022) contrast these results by showing a negative effect on rail ridership in mid-sized cities. Zhu, Xu, et al. (2021) and Ke et al. (2021) take a theoretical perspective by investigating the question through equilibrium models. These studies highlight the complementarity between ride-hailing and public transit under certain conditions. While all these studies focus on strategic interactions, our work provides an operational tool to support the coordination between ride-hailing and public transit at the platform level. By applying this tool across many configurations, we gain novel insights into the impact of coordination via key metrics such as the request fulfillment rate and the average customer travel time.

An extensive number of papers develop optimization models to integrate on-demand and public transit systems. The on-demand transport system can

take many forms, including ride-sharing (Stiglic et al., 2018, Kumar & Khani, 2021), bus-sharing (X. Li & Quadrifoglio, 2010, Montenegro et al., 2021), bike-sharing (Tang et al., 2018, X. Luo et al., 2021), or other vehicle-sharing systems (Friedrich & Noekel, 2017, Q. Luo et al., 2021). A variety of problems are considered, ranging from selecting the service area (X. Yan et al., 2019), assigning customers to transport modes (Fielbaum et al., 2024), sizing the vehicle fleet (Shehadeh et al., 2021), to determining service prices (Y. Chen & Wang, 2018).

Within this literature, several works address the problem of dispatching drivers to customers to/from public transit stations. Some research considers static dispatching problems where customer requests are fully known in advance (e.g., H. Wang, 2019). In the latter case, the problem can be formulated as a deterministic mathematical program, and solved before the start of the planning horizon. Other papers, as in the present article, examine more dynamic settings where requests are progressively revealed during the planning horizon. Among these, Agussurja et al. (2019) propose a value iteration algorithm with state-aggregation to solve a last-mile problem where customers arrive by batch at the station. S. Chen et al. (2020) dispatch vehicles for first-mile services using a rolling horizon approach based on mixed-integer programming. (X. Wang et al., 2024) propose a bi-level optimization framework to coordinate ride-hailing and public transit services in real time during periods of excessive demand. B. Sun et al. (2025) optimize both first-mile and last-mile trips to a transit station with a rolling-horizon column-generation algorithm. Most of these papers study many-to-one systems, where a limited number of customers start from or end at a single transit station. Often, they implement reactive rolling-horizon approaches that only account for currently observed demand. In contrast, our work applies to many-to-many systems, where customers can be picked up and dropped off across a large network of locations connected with multiple public transit stations. To capture future demand uncertainty, we optimize the system using an lookahead approach based on historical data.

Closest to our research, three studies develop operational models to integrate public transit and ride-hailing at the scale of a city (i.e., in many-to-many systems, with thousands of requests, and dozens of transit stations), using demand anticipative approaches. T.-Y. Ma et al. (2019) determine repositioning and dispatching decisions for drivers who connect customers to public transit. To make these decisions, they devise an insertion-based algorithm that relies on queueing approximations of future demand. Feng et al. (2022) and Y. Hu et al. (2025) both use reinforcement learning algorithms to dispatch drivers to customers. Similar to us, Feng et al. (2022) consider single-request trips, while Y. Hu et al. (2025) study a ride-pooling setting. Our work differs from these prior studies in several ways. In terms of modeling, unlike Feng et al. (2022) and Y. Hu et al. (2025), we use a multi-objective stochastic program that optimizes both first-mile and last-mile services. In contrast to T.-Y. Ma et al. (2019), we assign drivers over time intervals rather than at each customer arrival to improve the dispatching process (cf. Uber, 2025). In terms of methodology, all these studies capture demand uncertainty through value

function approximations. We differentiate by capturing demand uncertainty through scenarios, as part of a stochastic program. In terms of insights, our analysis evaluates the conditions that favor an integrated system, and implications for the stakeholders, topics barely touched in the mentioned studies, which mainly provided algorithmic contributions.

4.3 Definition of the Online Problem

In this section, we introduce the online problem of a platform that integrates public transit transfers into its dispatching process. We first describe the problem and define general notations. Then, we detail the two platform's decisions: customer directing and driver dispatching. Finally, we formulate the problem as a Markov decision process (MDP). The main notations introduced in this section are summarized in Table 4.1.

4.3.1 General Setting

We take the point of view of a ride-hailing platform that operates in a region (e.g., a city). The platform dispatches drivers to customers to maximize its profit while minimizing customer travel time. Both objectives are frequently used by ride-hailing platforms, so as to balance immediate profit with service quality, a proxy for long-term profit (see, e.g., Azagirre et al., 2024). To improve the dispatching process, the platform leverages the existing public transit system (e.g., the subway), whose main characteristics (e.g., travel times, and frequencies) are assumed to be known by the platform. Rather than always providing door-to-door service, the platform can recommend some customers to receive first-mile service up to a transit station, or last-mile service from a transit station. Customers then complete the remaining leg of their trip by public transit and walking in exchange for a monetary discount. By doing so, the platform can increase its service flexibility, both temporally and spatially. From a temporal perspective, service times can be adjusted, enabling drivers to better cope with the variable flow of customer requests. From a spatial perspective, the service pickup and dropoff locations can be modified, thereby bringing drivers closer to future customers.

The platform's service area is divided into a set $\mathcal{I} \equiv \{1, \ldots, n_I\}$ of disjoint locations (e.g., neighborhoods). A subset $\mathcal{I}^{\mathrm{PT}} \subseteq \mathcal{I}$ of the locations are connected to public transit, which allows customers to reach any other location where public transit is also available. To serve customers, the platform operates N drivers. For simplicity, the number of drivers is assumed to be constant, even though the assumption can be relaxed (see Section 4.5.2). The platform makes decisions over a planning horizon $\mathcal{T} \equiv \{1, 2, \ldots, T\}$ of discrete epochs, with epochs separated by small, fixed intervals of time (e.g., every 5 minutes). At each epoch $t \in \mathcal{T}$, the platform considers customers who have requested service over the last time interval. Each customer is indexed

Indices and sets	
$i \in \mathcal{I}$	for locations.
$j \in \mathcal{J}$	for customer origin-destination (OD) pairs.
$o_j \in \mathcal{I}$	for origin of OD pair j.
$d_j \in \mathcal{I}$	for destination of OD pair j.
$k \in \mathcal{J}$	for service pickup-dropoff (PD) pairs.
$t \in \mathcal{T}$	for decision epochs.
$\mathcal{I}^{\mathrm{PT}} \subseteq \mathcal{I}$	Set of locations connected to public transit.
$\mathcal{J}^t \subseteq \mathcal{J}$	Set of OD pairs for requests collected at epoch t.
$\mathcal{F}_j^t \subseteq \mathcal{I}^{\mathrm{PT}}$	Set of feasible dropoff locations for OD pair j directed to first-mile service at epoch t.
$\mathcal{L}_j^t \subseteq \mathcal{I}^{\mathrm{PT}}$	Set of feasible pickup locations for OD pair j directed to last-mile service at epoch t.
$\mathcal{K}^t \subseteq \mathcal{J}$	Set of possible PD pairs to which drivers can be dispatched at epoch t.
$\mathcal{D}_i^t \subseteq \mathcal{K}^t$	Set of possible PD pairs with location i as dropoff at epoch t.
$\mathcal{O}_k^t \subseteq \mathcal{I}$	Set of feasible origins for drivers dispatched to PD pair k at epoch t.
Parameters	
D_j^t	Observed customer demand for OD pair j at epoch t.
FT_{ji}	Travel time for OD pair j of the first leg by public transit ending in pickup location i.
ST_{ji}	Travel time for OD pair j of the second leg by public transit starting in dropoff location i.
TT_j	Travel time for OD pair j by walking and using public transit.
FC_{ji}	Cost for OD pair j of the first leg by public transit ending in pickup location i.
SC_{ji}	Cost for OD pair j of the second leg by public transit starting in dropoff location i.
DT_{ik}^t	Travel time from driver origin i to pickup and then dropoff of PD pair k.
DR_{ik}^t	Revenue for dispatching a driver in location i to PD pair k at epoch t.
N_i	Number of drivers available in location i at the first epoch.
$\beta_{ji}^{t,t'}$	1 if customer for OD pair j completes first leg in pickup location i at epoch t', 0 otherwise.
$\gamma_{ik}^{t,t'}$	1 if driver dispatched in location i to PD pair k completes service at epoch t', 0 otherwise.
Decision variables	
w_j^t	Number of unsatisfied customers for OD pair j at epoch t.
x_j^t	Number of customers for OD pair j directed to door-to-door service at epoch t.
y_{ji}^t	Number of customers for OD pair j directed to first-mile service with dropoff i at epoch t.
z_{ji}^t	Number of customers for OD pair j directed to last-mile service with pickup i at epoch t.
d_{ik}^t	Number of drivers dispatched in location i to PD pair k at epoch t.

Table 4.1: Summary of the main notations.

by an origin-destination (OD) pair j that belongs to the set of location pairs $\mathcal{J} \equiv \{(1,1), \ldots, (o_j, d_j), \ldots, (n_I, n_I)\}$ where o_j and d_j denote the customer origin and destination. The number of customers for any OD pair j at epoch t is random, and the realized number of customers making a request is given by D_j^t. For each customer, the platform makes a *directing decision*, determining which service to offer to the customer and a *dispatching decision*, determining from which location to assign a driver to the resulting service. Next, we present these two interdependent decisions and their associated notations.

4.3.2 Customer Directing Decision

The customer directing decision is illustrated in Fig. 4.1(a). At epoch t, the set of OD pairs for which at least one customer has made a request is defined by $\mathcal{J}^t$ (with $\mathcal{J}^t \subseteq \mathcal{J}$). For each OD pair $j \in \mathcal{J}^t$, the platform determines whether to

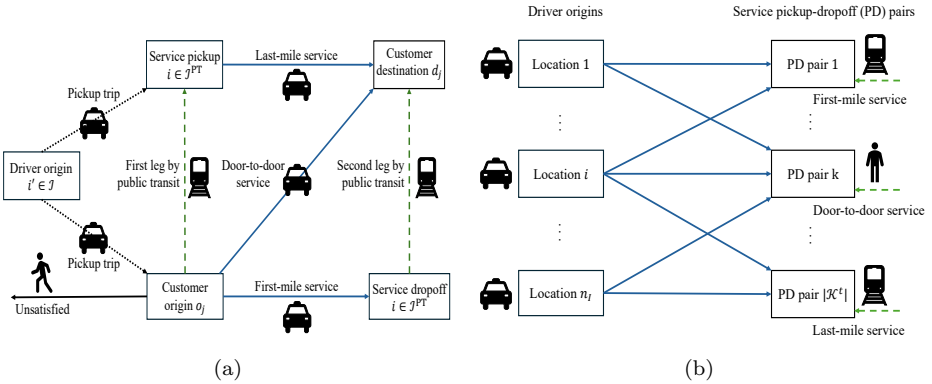


Figure 4.1: (a) The directing decision for a customer at epoch t. (b) The dispatching decision for drivers at epoch t.

satisfy the customer requests and, if so, whether to provide door-to-door, first-mile, or last-mile service. *Door-to-door service* consists of serving a customer from her origin o_j to her destination d_j without passing by public transit. The integer variable x_j^t denotes the number of customers directed to door-to-door service for OD pair j at epoch t. *First-mile service* refers to serving a customer from her origin o_j to a dropoff location $i \in \mathcal{I}^{\text{PT}}$, which is selected by the platform. Then, the customer completes the second leg of the trip from the dropoff location i to her destination d_j by public transit and walking. The integer variable y_{ji}^t denotes the number of customers with OD pair j who are directed to first-mile service at epoch t, and will be dropped off at location i. *Last-mile service* corresponds to the reverse case, where a customer receives service from a selected pickup location $i \in \mathcal{I}^{\text{PT}}$ to her destination d_j. Before this, the customer completes the first leg of the trip from o_j to pickup location i by walking and public transit. The integer variable z_{ji}^t denotes the number of customers with OD pair j who are directed to last-mile service at epoch t, and will be picked up at location i. Lastly, an unsatisfied customer for OD pair j is assumed to walk and possibly use public transit. The integer variable w_j^t corresponds to the number of unsatisfied customers for OD pair j at epoch t. We never allow customers to receive both first-mile and last-mile services, as it may result in cumbersome transfers.

When a customer receives first-mile service, the travel time of the second leg by public transit, from dropoff location i to customer destination d_j, is given by ST_{ji}. When a customer receives last-mile service, the travel time of the first leg by public transit, from customer origin o_j to pickup location i, is given by FT_{ji}. Associated with the latter notation, the indicator function $\beta_{ji}^{t,t'}$ takes the value 1 if a customer with OD pair j directed to last-mile service at epoch t completes the first leg to pickup location i at epoch t', and 0 otherwise. Unsatisfied customers with OD pair j incur a travel time TT_j from origin to

destination, walking and possibly using public transit. These parameters are assumed to be deterministic, and can be deduced beforehand, e.g., using a shortest-path algorithm. Whenever the platform assigns a customer with OD pair j to first-mile or last-mile service, it incurs a cost SC_{ji} for the second leg or a cost FC_{ji} for the first leg, which depends on the dropoff or pickup location i. These costs can reflect prices of transit tickets, or losses of brand image, as customers may endure longer or less convenient journeys than with door-to-door services. The lost sale costs for unsatisfied customers are not modeled, as our problem already includes revenues for serving customers.

For simplicity, we do not explicitly model customer adoption. Instead, we consider that a given percentage δ of the customers (e.g., 25%) indicates beforehand that they are willing to receive first-mile and last-mile services if several quality constraints are satisfied. Throughout the paper, these constraints are fixed in advance and are identical for all customers. However, the solution approach presented in Section 4.5 remains applicable when customers specify these constraints themselves (cf. Section 4.5.2). In this setting, customers are eligible (and supposed to accept) first-mile or last-mile service only if the following conditions are satisfied:

(i) their origin (for last-mile service) or destination (for first-mile service) is in the same location as a transit station;

(ii) their initial door-to-door service time is superior to a threshold $\bar{t}_1$ (e.g., 10 minutes);

(iii) the actual travel time (i.e., walking time, public transit time, plus service time) does not increase the door-to-door service time by more than some percentage $\bar{t}_2$ (e.g., 30%);

(iv) the actual price is decreased, compared to the door-to-door price, by at least a certain amount $\bar{t}_3$ (e.g., 2.5€).

In view of these constraints, let $\mathcal{F}_j^t$ (with $\mathcal{F}_j^t \subseteq \mathcal{I}^{\mathrm{PT}}$) denote the set of feasible dropoff locations for a customer with origin-destination (OD) pair j who is directed to first-mile service at epoch t. Similarly, let $\mathcal{L}_j^t$ (with $\mathcal{L}_j^t \subseteq \mathcal{I}^{\mathrm{PT}}$) represent the set of feasible pickup locations for a customer with OD pair j directed to last-mile service at epoch t.

4.3.3 Driver Dispatching Decision

In parallel with the directing decision, the platform dispatches drivers to services. Each service is characterized by a pickup-dropoff (PD) pair, indexed by $k \in \mathcal{J}$, which describes where a customer is picked up and dropped off. We use different indexes for service PD pairs than for customer OD pairs to clearly distinguish the customer origin and destination from the service pickup and dropoff. If a customer with OD pair j receives door-to-door service, the service pickup and dropoff coincide with the customer origin and destination (i.e.,

$k \equiv (o_j, d_j)\big)$. For first-mile service, the service pickup is also the customer origin, but the dropoff is changed to $i \in \mathcal{I}^{\mathrm{PT}}$ $\big($i.e., $k \equiv (o_j, i)\big)$. For last-mile service, the service dropoff remains the customer destination, but the pickup occurs at $i \in \mathcal{I}^{\mathrm{PT}}$ $\big($i.e., $k \equiv (i, d_j)\big)$. For each customer directed to first-mile or last-mile service, the platform must select one transit station among multiple options. Therefore, each OD pair $j \in \mathcal{J}^t$ may be associated with several PD pairs. It follows that the set of possible PD pairs $\mathcal{K}^t$ at epoch t often largely exceeds the set of OD pairs $\mathcal{J}^t$, such that $|\mathcal{K}^t| \supseteq |\mathcal{J}^t|$ and $|\mathcal{K}^t| \gg |\mathcal{J}^t|$.

The driver dispatching decision is depicted in Fig. 4.1(b). At each epoch t, this decision involves dispatching available drivers from the set of locations $\mathcal{I}$ to services specified by the set of possible PD pairs $\mathcal{K}^t$. Therefore, the decision can be modeled as an assignment on the bipartite graph where a vertex $i \in \mathcal{I}$ represents a driver origin location, and a vertex $k \in \mathcal{K}^t$ represents a PD pair. Each edge (i, k) between two vertices represents the possibility to dispatch drivers from a location i to a PD pair k. The capacity of an edge (i, k) is defined as the minimum between the number of drivers available in the location i and the number of services to provide for the PD pair k. Associated with every edge (i, k), the integer variable d_{ik}^t denotes the number of drivers dispatched from location i to PD pair k at epoch t.

The (deterministic) travel time for a driver from origin location i to pickup and complete a service with PD pair k is expressed as DT_{ik}^t. Accordingly, let the indicator function $\gamma_{ik}^{t,\,t'}$ take the value 1 if a driver at origin i dispatched to service with PD pair k at epoch t completes her service at epoch t', and 0 otherwise. Each driver dispatched at origin i to PD pair k at epoch t generates a revenue DR_{ik}^t. This term can capture the gross revenue from the service pickup to the service dropoff, and the operational cost from the driver origin to the service dropoff. After completing service, drivers become available again at the dropoff location. The set of PD pairs with dropoff location i for drivers dispatched at epoch t is written as $\mathcal{D}_i^t \subseteq \mathcal{K}^t$. Like for the customer directing decision, we impose a quality constraint to prevent excessive customer waiting. That is, a driver can only be dispatched to a service if the resulting pickup time (i.e., the time from the driver origin to the service pickup) is inferior to a fixed threshold (e.g., 10 minutes). Consequently, the set of feasible driver origin locations to serve a PD pair k, with drivers dispatched at epoch t is given by $\mathcal{O}_k^t \subseteq \mathcal{I}$.

4.3.4 MDP Formulation

We now formalize the problem as a Markov decision process (MDP), defining states, decisions, transitions, and rewards.

States. At epoch $t \in \mathcal{T}$, the system state is characterized by the following components: (i) the demand D_j^t for each OD pair j at the current epoch, (ii) the number of customers $L_k^{t,\,t'}$ receiving last-mile service for PD pair k that completed their first leg via public transit at the current epoch ($t' =$

t), or who will complete their first leg subsequently ($t' > t$), and (iii) the number of drivers $N_i^{t,\,t'}$ in each location i that are available at the current epoch ($t' = t$) or who will become available subsequently ($t' > t$). Therefore, the state at epoch t can be defined as the tuple $\boldsymbol{s}^t = (\boldsymbol{D}^t, \boldsymbol{L}^t, \boldsymbol{N}^t)$, with vector $\boldsymbol{D}^t = (D^t_{(1,\,1)}, \ldots, D^t_{(n_I,\,n_I)})$, and matrices $\boldsymbol{L}^t = (L^{t,\,t}_{(1,\,1)}, \ldots, L^{t,\,T}_{(n_I,\,n_I)})$, and $\boldsymbol{N}^t = (N_1^{t,\,t}, \ldots, N_{n_I}^{t,\,T})$.

Decisions. A decision at epoch t consists of a tuple $\boldsymbol{a}^t \equiv (\boldsymbol{w}^t, \boldsymbol{x}^t, \boldsymbol{y}^t, \boldsymbol{z}^t, \boldsymbol{d}^t)$, where (i) $\boldsymbol{w}^t \equiv (w_j^t)_{j \in \mathcal{J}^t}$ denotes unsatisfied customers, (ii) $\boldsymbol{x}^t \equiv (x_j^t)_{j \in \mathcal{J}^t}$ is associated with customers receiving door-to-door services, (iii) $\boldsymbol{y}^t \equiv (y_{ji}^t)_{j \in \mathcal{J}^t,\, i \in \mathcal{F}_j^t}$ corresponds to customers directed to first-mile services, (iv) $\boldsymbol{z}^t \equiv (z_{ji}^t)_{j \in \mathcal{J}^t,\, i \in \mathcal{L}_j^t}$ refers to customers directed to last-mile services, and (v) $\boldsymbol{d}^t \equiv (d_{ik}^t)_{i \in \mathcal{O}_k^t,\, k \in \mathcal{K}^t}$ represents drivers dispatched to complete the resulting services.

Transitions. Between decision epochs t and $t+1$, the system transitions from the old state $\mathbf{s}^t$ to a new state $\mathbf{s}^{t+1}$. The demand vector $\mathbf{D}^t$ changes stochastically, based on customer requests collected between the epochs t and $t+1$. The matrices $\mathbf{L}^t$ and $\mathbf{N}^t$ evolve deterministically depending on the decisions made at epoch t such that

$$L_k^{t+1,\,t'} = L_k^{t,\,t'} + \sum_{j,\,i\,:\,k=(i,\,d_j)} \beta_{ji}^{t,\,t'} \cdot z_{ji}^t \quad \forall\, t' = t+1, \ldots, T,\ k \in \mathcal{J}, \tag{4.1}$$

$$N_i^{t+1,\,t+1} = N_i^{t,\,t} - \sum_{k \in \mathcal{K}^t} d_{ik}^t + N_i^{t,\,t+1} + \sum_{l \in \mathcal{I}} \sum_{k \in \mathcal{D}_i^t} \gamma_{lk}^{t,\,t+1} \cdot d_{lk}^t \quad \forall\, i \in \mathcal{I}, \tag{4.2}$$

$$N_i^{t+1,\,t'} = N_i^{t,\,t'} + \sum_{l \in \mathcal{I}} \sum_{k \in \mathcal{D}_i^t} \gamma_{lk}^{t,\,t'} \cdot d_{lk}^t \quad \forall\, t' = t+2, \ldots, T,\ i \in \mathcal{I}. \tag{4.3}$$

Eq. (4.1) tracks last-mile customers who complete the first leg of their trip at epochs $t' > t$, incorporating those directed to last-mile service at epoch t. Eq. (4.2) updates the number of available drivers in location i at epoch $t+1$, based on the number of drivers available at epoch t, the number of drivers dispatched at epoch t, and those completing service between epochs t and $t+1$. Eq. (4.3) captures drivers dispatched at epoch t who become available at epochs $t' > t+1$.

Rewards. Along with state transitions, the platform collects rewards, which depend on the action taken at each epoch. As the platform aims to optimize profit and customer travel time, the following reward functions are defined:

$$\begin{aligned} C^{Profit}(\mathbf{s}^t,\, \mathbf{a}^t) = &\sum_{i \in \mathcal{O}_k^t} \sum_{k \in \mathcal{K}^t} DR_{ik}^t \cdot d_{ik}^t \\ &- \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} SC_{ji} \cdot y_{ji}^t - \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{L}_j^t} FC_{ji} \cdot z_{ji}^t \\ C^{Time}(\mathbf{s}^t,\, \mathbf{a}^t) = &\sum_{i \in \mathcal{O}_k^t} \sum_{k \in \mathcal{K}^t} DT_{ik} \cdot d_{ik}^t \end{aligned} \tag{4.4}$$

$$+ \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} ST_{ji} \cdot y_{ji}^t + \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{L}_j^t} FT_{ji} \cdot z_{ji}^t + \sum_{j \in \mathcal{J}^t} TT_j \cdot w_j^t \tag{4.5}$$

In Eq. (4.4), the reward function deduces the platform's profit at epoch t, and includes revenues for dispatching drivers to customers, and costs for directing customers to first-mile and last-mile services and completing second and first legs by public transit. In Eq. (4.5), the reward function computes the customers' travel time at epoch t, and includes the time spent waiting for pickup and being served, the time spent in the second leg and first leg, and the time spent in public transit and walking if unsatisfied. The platform objectives are to maximize the expected total profit, i.e., $\max \mathbb{E}\big[\sum_{t \in \mathcal{T}} C^{Profit}(\mathbf{s}^t, \mathbf{a}^t)\big]$, and to minimize the expected total travel time, i.e., $\min \mathbb{E}\big[\sum_{t \in \mathcal{T}} C^{Time}(\mathbf{s}^t, \mathbf{a}^t)\big]$. To consider both objectives, our solution approach will optimize their weighted sum, as will be explained in Section 4.5.

4.4 The Offline Problem

If we suppose that the customer requests are known in advance, the problem becomes *offline* (or static). In the offline problem, there is no uncertainty about future customers, so the problem can be solved before the start of the planning horizon. This section presents a deterministic integer program to solve the offline problem. The purpose of this section is twofold. First, it allows us to derive an upper bound (lower bound) on the optimal profit objective (travel time objective), which helps evaluate the performance of any approach applied to the online problem (Section 4.6.2). This is achieved by solving the deterministic program with the demand revealed at the end of the planning horizon. Second, it provides intuition on the problem structure, laying the ground for the approach developed in Section 4.5.

The offline problem can be formulated as the following integer program:

$$\max\ Profit^{off} = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{O}_k^t} \sum_{k \in \mathcal{K}^t} DR_{ik}^t \cdot d_{ik}^t$$

$$- \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} SC_{ji} \cdot y_{ji}^t - \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{L}_j^t} FC_{ji} \cdot z_{ji}^t \tag{4.6a}$$

$$\min\ Time^{off} = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{O}_k^t} \sum_{k \in \mathcal{K}^t} DT_{ik} \cdot d_{ik}^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} ST_{ji} \cdot y_{ji}^t$$

$$+ \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{L}_j^t} FT_{ji} \cdot z_{ji}^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} TT_j \cdot w_j^t \tag{4.6b}$$

$$\text{s.t. } w_j^t + x_j^t + \sum_{i \in \mathcal{F}_j^t} y_{ji}^t + \sum_{i \in \mathcal{L}_j^t} z_{ji}^t = D_j^t \quad \forall\, t \in \mathcal{T},\, j \in \mathcal{J}^t, \tag{4.6c}$$

$$\sum_{i \in \mathcal{O}_k^t} d_{ik}^t = x_k^t + \sum_{j,\, i\,:\, k=(o_j,\, i)} y_{ji}^t + \sum_{u=1}^{t} \sum_{j,\, i\,:\, k=(i,\, d_j)} \beta_{ji}^{u,\, t} \cdot z_{ji}^u \quad \forall\, t \in \mathcal{T},\, k \in \mathcal{K}^t, \tag{4.6d}$$

$$\sum_{k \in \mathcal{K}^1} d_{ik}^1 \leq N_i \quad \forall\, i \in \mathcal{I}, \tag{4.6e}$$

$$\sum_{u=1}^{t} \sum_{k \in \mathcal{K}^u} d_{ik}^u \leq N_i + \sum_{u=1}^{t} \sum_{v=u+1}^{t} \sum_{l \in \mathcal{I}} \sum_{k \in \mathcal{D}_i^u} \gamma_{lk}^{u,v} \cdot d_{lk}^u \quad \forall\, t \in \mathcal{T} \setminus \{1\},\, i \in \mathcal{I}, \tag{4.6f}$$

$$w_j^t,\, x_j^t,\, y_{ji}^t,\, z_{jl}^t \in \mathbb{Z}_0^+ \quad \forall\, t \in \mathcal{T},\, j \in \mathcal{J}^t,\, i \in \mathcal{F}_j^t,\, l \in \mathcal{L}_j^t \tag{4.6g}$$

$$d_{ik}^t \in \mathbb{Z}_0^+ \quad \forall\, t \in \mathcal{T},\, k \in \mathcal{K}^t,\, i \in \mathcal{O}_k^t. \tag{4.6h}$$

Objective function (4.6a) represents the total platform's profit. Objective function (4.6b) denotes the total customers' travel time. Constraints (4.6c) enforce that each customer request is either unsatisfied or receives door-to-door, first-mile, or last-mile service for each origin-destination pair at any epoch. Constraints (4.6d) state that the number of drivers dispatched to customers must equal the final demand for each pickup-dropoff pair at any epoch. On the right-hand side, the first term stands for customers who start receiving door-to-door services at the current epoch, the second term for customers who start receiving first-mile services at the current epoch, and the third term for customers who were directed to last-mile services at previous epochs and start being served at the current epoch. Constraints (4.6e), (4.6f) are "flow constraints", and enforce that the number of dispatched drivers is always inferior to the number of available drivers in each location, at any epoch. More precisely, in constraints (4.6f), the term on the left-hand side is the number of drivers who have been dispatched until the current epoch. On the right-hand side, the first term denotes the number of drivers available in a location at the first epoch, and the second term denotes the number of drivers who have finished serving customers in the location until the current epoch. Lastly, constraints (4.6g), (4.6h) are integrality constraints.

In the above formulation, the extreme points of the convex hull are integral under some conditions, and it suffices to solve the linear relaxation to obtain optimal solutions to the offline problem. Proposition 4.1 defines these conditions.

Proposition 4.1. *Let*

(i) $z_{ji}^t = 0$ *for all* $j \in \mathcal{J}^t$, $i \in \mathcal{L}_j^t$, $t \in \mathcal{T}$ *(customers never receive last-mile service),*

(ii) $\gamma_{ik}^{t,t'} = \gamma_{lk}^{t,t'}$ *for all* $k \in \mathcal{K}^t$, $i, l \in \mathcal{O}_k^t : i \neq l$, *and* $t, t' \in \mathcal{T} : t < t'$ *(pickup time is fixed for each PD pair* k*),*

(iii) $DR_{ik}^t = DR_i^t + DR_k^t$ *for all* $i \in \mathcal{I}$, $k \in \mathcal{K}^t$ *(the dispatching reward is separable).*

Then the optimal solution of the linear relaxation of the offline problem (4.6a)-(4.6h) is integral.

The proof of Proposition 4.1 is given in Appendix 4.A.1. In short, it demonstrates that the linear relaxation of the offline problem can be reduced to a

min-cost flow problem with integral optimal solutions. The first condition lets us remove the variables for last-mile services from the above formulation. The second condition allows us to convert constraints (4.6e), (4.6f) to simpler flow constraints. The third condition is used to differentiate driver dispatching from customer service, and introduce additional variables to track the number of customers served at each epoch.

Proposition 4.1 states that, under certain conditions, the optimal solution is integral, even when considering the linear relaxation of the offline problem. For the general case, we have no guarantee that the linear relaxation results in an integral optimal solution. However, in our experiments, we observed that most variables in the optimal solutions of the relaxed problems are integral (about 90% on average). One possible explanation is that the offline problem is often close to satisfying the conditions of Proposition 4.1. For the first condition, only a minority of customers receive last-mile service (about 10% on average). For the second condition, drivers are mainly dispatched to customers nearby, with slight variations in pickup times. For the third condition, it fits many pricing schemes encountered in practice, e.g., with revenues only depending on the distance between pickup and dropoff. Thus, even when the conditions do not hold, the offline problem can be solved efficiently, and can be used for evaluating the performance of any solution to the online problem. As shown in the next section, this result is also exploited to reduce computational effort.

4.5 Stochastic Lookahead Approach

In this section, we develop a solution approach based on stochastic dynamic optimization to tackle the online problem. Section 4.5.1 gives an overview of the approach. Section 4.5.2 defines the two-stage stochastic program supporting the approach. Section 4.5.3 details two procedures to reduce the complexity and solve the stochastic program efficiently.

4.5.1 Approach Overview

The central idea behind our solution approach, illustrated in Fig. 4.2, is to re-optimize a two-stage stochastic program at every epoch based on newly revealed information. The approach accounts for the real-time evolution of the system, adjusting decisions *dynamically*. At the same time, it anticipates future demand patterns, making decisions *proactively*.

To make dynamic decisions, the approach follows a rolling-horizon strategy that re-optimizes decisions at each epoch t. For each optimization, the two-stage stochastic program receives the system state $\mathbf{s}^t = (\mathbf{D}^t, \mathbf{L}^t, \mathbf{N}^t)$ as an input, influencing the solution space and resulting decisions.

To make proactive decisions, the approach optimizes a two-stage stochastic program characterized by a set $\mathcal{S} \equiv \{1, \ldots, n_S\}$ of demand scenarios. Each scenario $s \in \mathcal{S}$ corresponds to a realization of the customer demand over the next

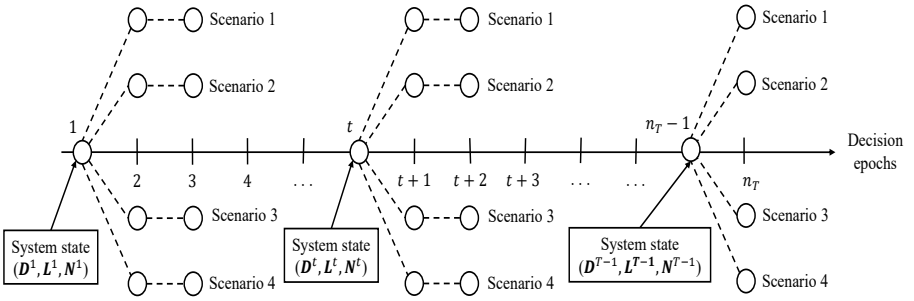


Figure 4.2: The rolling horizon approach, used with $n_S = 4$ and $H = 2$ over decision epochs $\mathcal{T} = \{1, 2, \ldots, n_T\}$.

n_H epochs, and is derived, e.g., from historical data. Accordingly, in the two-stage stochastic program, the decision variables fall into two categories. The first-stage variables represent "here-and-now" decisions made at the current epoch t before realization of a demand scenario. The second-stage variables represent "recourse" decisions made at future epochs $h = t+1, \ldots, t+n_H$ after the realization of a demand scenario. Only first-stage variables are implemented, while second-stage variables are used to anticipate future decisions.

As shown in Fig. 4.2, at the first epoch (left-hand side of Fig. 4.2), the approach receives the system state as an input and optimizes the two-stage stochastic program over $n_H + 1$ epochs ($h = 1, \ldots, n_H + 1$), with n_S demand scenarios. The first-stage solutions are implemented as decisions, and the system state is updated accordingly. At every epoch $t \leq n_T - n_H$ (center of Fig. 4.2), the process is repeated, implementing decisions and updating states. At epoch $t = n_T - n_H + 1$ (right-hand side of Fig. 4.2), the approach optimizes a two-stage stochastic program over n_H epochs ($h = n_T - n_H + 1, \ldots, n_T$), still with n_S demand scenarios. The process repeats until epoch $t = n_T$, gradually decreasing the number of epochs in the two-stage stochastic program from n_H to 2. Finally, at the last epoch $t = n_T$, a deterministic program over only one epoch is optimized, completing the approach.

4.5.2 Two-Stage Stochastic Program

To formulate the two-stage stochastic program, we first introduce some additional notation. Let $\widehat{D}^h_{j,s}$ be the number of customers with OD pair j at epoch $h = t+1, \ldots, t+n_H$ in scenario s. Let $\widehat{\mathcal{J}}^{t'}_s$ and $\widehat{\mathcal{K}}^{t'}_s$ denote the corresponding sets of OD and PD pairs. The first-stage variables use the same notations as before and are given by w^t_j, x^t_j, y^t_{ji}, z^t_{ji}, and d^t_{ik} at epoch t. The second-stage variables extend the notations and are denoted by $w^h_{j,s}$, $x^h_{j,s}$, $y^h_{ji,s}$, $z^h_{ji,s}$, and $d^h_{ik,s}$ at subsequent epochs $h = t+1, \ldots, t+n_H$, for each scenario $s \in \mathcal{S}$.

With these notations, the two-stage stochastic program at each epoch t

(with $t \leq n_T - n_H$) can be expressed as

$$\max \ Profit^{on} =$$

$$\sum_{k\in\mathcal{K}^t}\sum_{i\in\mathcal{O}_k^t} DR_{ik}^t \cdot d_{ik}^t - \sum_{j\in\mathcal{J}^t}\Big[\sum_{i\in\mathcal{F}_j^t} FC_{ji}\cdot y_{ji}^t - \sum_{i\in\mathcal{L}_j^t} LC_{ji}\cdot z_{ji}^t\Big]$$

$$+\frac{1}{n_S}\sum_{s\in\mathcal{S}}\sum_{h=t+1}^{t+n_H}\Big(\sum_{k\in\widehat{\mathcal{K}}_s^h}\sum_{i\in\mathcal{O}_k^h} DR_{ik}^h\cdot d_{ik,s}^h - \sum_{j\in\widehat{\mathcal{J}}_s^h}\Big[\sum_{i\in\mathcal{F}_j^h} FC_{ji}\cdot y_{ji,s}^h - \sum_{i\in\mathcal{L}_j^h} LC_{ji}\cdot z_{ji,s}^h\Big]\Big) \tag{4.7a}$$

$$\min \ Time^{on} = \sum_{k\in\mathcal{K}^t}\sum_{i\in\mathcal{O}_k^t} DT_{ik}^t\cdot d_{ik}^t + \sum_{j\in\mathcal{J}^t}\Big[\sum_{i\in\mathcal{F}_j^t} FT_{ji}\cdot y_{ji}^t + \sum_{i\in\mathcal{L}_j^t} LT_{ji}\cdot z_{ji}^t + TT_j^t\cdot w_j^t\Big]$$

$$+\frac{1}{n_S}\sum_{s\in\mathcal{S}}\sum_{h=t+1}^{t+n_H}\Big(\sum_{k\in\widehat{\mathcal{K}}_s^h}\sum_{i\in\mathcal{O}_k^h} DT_{ik}^h\cdot d_{ik,s}^h$$

$$+\sum_{j\in\widehat{\mathcal{J}}_s^h}\Big[\sum_{i\in\mathcal{F}_j^h} FT_{ji}\cdot y_{ji,s}^h + \sum_{i\in\mathcal{L}_j^h} LT_{ji}\cdot z_{ji,s}^h + TT_j^h\cdot w_{j,s}^h\Big]\Big) \tag{4.7b}$$

$$\text{s.t. } w_j^t + x_j^t + \sum_{i\in\mathcal{F}_j^t} y_{ji}^t + \sum_{i\in\mathcal{L}_j^t} z_{ji}^t = D_j^t \quad \forall\, j\in\mathcal{J}^t, \tag{4.7c}$$

$$w_{j,s}^h + x_{j,s}^h + \sum_{i\in\mathcal{F}_j^h} y_{ji,s}^h + \sum_{i\in\mathcal{L}_j^h} z_{ji,s}^h = D_{j,s}^h \quad \forall\, s\in\mathcal{S},\, h=t+1,\ldots,t+n_H,\, j\in\widehat{\mathcal{J}}_s^h, \tag{4.7d}$$

$$\sum_{i\in\mathcal{O}_k^t} d_{ik}^t = L_k^{t,\,t} + x_k^t + \sum_{j,\,i\,:\,k=(o_j,\,i)} y_{ji}^t \quad \forall\, k\in\mathcal{K}^t, \tag{4.7e}$$

$$\sum_{i\in\mathcal{O}_k^h} d_{ik,s}^h = L_k^{t,\,h} + x_{k,s}^h + \sum_{j,\,i\,:\,k=(o_j,\,i)} y_{ji,s}^h + \sum_{j,\,i\,:\,k=(i,\,d_j)} \beta_{ji}^{t,\,h}\cdot z_{ji}^t$$

$$+\sum_{u=t+1}^{h}\sum_{j,\,i\,:\,k=(i,\,d_j)} \beta_{ji}^{u,\,h}\cdot z_{ji,s}^u \quad \forall\, s\in\mathcal{S},\, h=t+1,\ldots,t+n_H,\, k\in\widehat{\mathcal{K}}_s^h, \tag{4.7f}$$

$$\sum_{k\in\mathcal{K}^t} d_{ik}^t \leq N_i^{t,\,t} \quad \forall\, i\in\mathcal{I}, \tag{4.7g}$$

$$\sum_{k\in\mathcal{K}^t} d_{ik}^t + \sum_{u=t+1}^{h}\sum_{k\in\widehat{\mathcal{K}}_s^u} d_{ik,s}^u \leq \sum_{u=t}^{h} N_i^{t,\,u} + \sum_{v=t+1}^{h}\sum_{k\in\mathcal{D}_i^t}\sum_{l\in\mathcal{I}} \gamma_{lk}^{t,\,v}\cdot d_{lk}^t$$

$$+\sum_{u=t+1}^{h}\sum_{v=u+1}^{h}\sum_{k\in\mathcal{D}_i^u}\sum_{l\in\mathcal{I}} \gamma_{lk}^{u,\,v}\cdot d_{lk,s}^u \forall\, s\in\mathcal{S},\, h=t+1,\,\ldots,\,t+n_H,\, i\in\mathcal{I}, \tag{4.7h}$$

$$w_j^t,\, x_j^t,\, y_{ji}^t,\, z_{jl}^t \in \mathbb{Z}_0^+ \quad \forall j\in\mathcal{J}^t,\, i\in\mathcal{F}_j^t,\, l\in\mathcal{L}_j^t, \tag{4.7i}$$

$$d_{ik}^t \in \mathbb{Z}_0^+ \quad \forall\, k\in\mathcal{K}^t, i\in\mathcal{O}_k^t, \tag{4.7j}$$

$$w_{j,s}^h,\, x_{j,s}^h,\, y_{ji,s}^h,\, z_{jl,s}^h \in \mathbb{Z}_0^+$$

$$\forall\, s\in\mathcal{S},\, h=t+1,\ldots,t+n_H,\, j\in\widehat{\mathcal{J}}_s^h,\, i\in\mathcal{F}_j^h,\, l\in\mathcal{L}_j^h, \tag{4.7k}$$

$$d_{ik,s}^h \in \mathbb{Z}_0^+ \quad \forall\, s\in\mathcal{S},\, h=t+1,\ldots,t+n_H,\, k\in\widehat{\mathcal{K}}_s^h,\, i\in\mathcal{O}_k^h. \tag{4.7l}$$

This formulation can be viewed as the stochastic version of the offline problem (4.6a)-(4.6h). Objective function (4.7a) maximizes the platform's profit,

while objective function (4.7b) minimizes the total travel time of customers. Constraints (4.7c) and (4.7d) ensure that each customer is either unsatisfied or directed to door-to-door, first-mile, or last-mile service. In constraint (4.7c), the right-hand side is given by the first component of the system state, i.e., the current demand, while in (4.7d), it represents potential future demand. Constraints (4.7e) and (4.7f) ensure that a driver is dispatched to each customer receiving service. In constraints (4.7e) and (4.7f), the first term on the right-hand side is derived from the second component of the system state and corresponds to customers directed to last-mile service who complete the first leg of their trip at the decision epoch. Constraints (4.7g) and (4.7h) ensure that the number of drivers dispatched does not exceed the number of drivers available. The right-hand side term comes from the third component of the system state and reflects drivers available at the current epoch, or who become available at subsequent epochs. Finally, constraints (4.7i)–(4.7l) are integrality constraints. To optimize both objectives (i.e., the platform's profit and the customers' total travel time), we formulate the two-stage stochastic program (4.7a)-(4.7l) as a single-objective program that consists of a weighted sum of the two objectives. The objective is then formulated as $\max Obj^{on} = \alpha \cdot Profit^{on} - (1-\alpha) \cdot Time^{on}$ where $\alpha \in [0,1]$ is the parameter trading off profit maximization ($\alpha = 1$) with travel time minimization ($\alpha = 0$).

Although our solution approach is tailored to our problem setting, it readily applies to more complex environments. First, customers are assumed to wait no more than a single decision epoch. However, our approach can handle waiting by re-generating requests for unsatisfied customers across multiple epochs. Second, while drivers currently move only when serving requests, repositioning can be included by modeling fictitious requests at each epoch. Third, though the number of drivers is supposed to be constant, a variable number of drivers can be incorporated in the system state and scenarios. Last, the quality constraints for feasible first-mile and last-mile services (cf. Section 4.3.2) are assumed fixed across all customers. Yet, our formulation allows customers to specify these constraints individually when submitting a request. In such cases, the sets $\mathcal{F}_j^t$ and $\mathcal{L}_j^t$ reflect information shared by customers for OD pair j at epoch t.

While Proposition 4.1 states that the offline problem can be solved efficiently under certain conditions, our two-stage stochastic program can never be solved in polynomial time with more than one scenario (i.e., $S > 1$). Indeed, it can be shown that the model reduces to the model of Lowalekar et al. (2018) when customers have only access to door-to-door service over two epochs, and the platform only maximizes profit. Since their model is NP-hard (see Proposition 3 in their article), our model is NP-hard as well. Given this result, it is not surprising that our stochastic program becomes intractable for realistic instances. The issue is further aggravated by the need to compute solutions within a reduced time. Therefore, in the following section, we introduce two procedures to improve the model tractability.

4.5.3 Scaling the Approach

In this section, we first present a preprocessing procedure to select decision variables and reduce the complexity of the stochastic program. We then adapt the L-shaped method to our model so as to reduce its resolution time. For exposition purposes, we only consider the profit-maximizing two-stage stochastic program in this section, although the two procedures can be applied with any objective function.

Preprocessing Procedure

One major limitation of our stochastic program is its large number of variables and constraints. Specifically, our formulation is characterized by $\mathcal{O}(n_s \cdot n_H \cdot (n_I)^3)$ variables and $\mathcal{O}(n_S \cdot n_H \cdot (n_I)^2)$ constraints.[1] Many variables can be removed by enforcing the quality constraints described in Section 4.3. In addition, constraints (4.7c), (4.7d) are only defined over the sets $\mathcal{J}^t$ and $\widehat{\mathcal{J}}_s^{t'}$, and constraints (4.7e), (4.7f) over the sets $\mathcal{K}^t$ and $\mathcal{K}_s^{t'}$, rather than over the set $\mathcal{J}$. Despite these reductions, the tractability of the model remains limited due to the large number of OD pairs and PD pairs.

A natural way to reduce model complexity is to limit the number of decision variables. In our case, most of the decision variables represent drivers dispatched from their origin location $i \in \mathcal{I}$ to a PD pair $k \in \mathcal{K}^t$. To simplify the model while preserving solution quality, we devise a preprocessing procedure, summarized by Algorithm 3. For each PD pair $k \in \mathcal{K}^t$ and each PD pair $k \in \widehat{\mathcal{K}}_s^h$, the algorithm retains only the $\overline{R}$ variables d_{ik}^t and $d_{ik,s}^h$, characterized by the lowest pickup times and discards all the remaining variables. The stochastic program is then formulated and solved over the restricted sets of selected driver origins $\mathcal{O}_k^t(\overline{R})$ and $\mathcal{O}_k^h(\overline{R})$, instead of the initial sets $\mathcal{O}_k^t$ and $\mathcal{O}_k^h$, for each $k \in \mathcal{K}^t$ or $k \in \widehat{\mathcal{K}}_s^h$. Intuitively, the procedure is expected to select variables that are optimal or near-optimal, as optimal solutions typically prevent drivers from driving empty for long periods. At the same time, the resulting stochastic program is characterized by fewer decision variables, which simplifies its resolution.

The only issue with our preprocessing procedure is that customers previously directed to last-mile services are not guaranteed to receive service at the current epoch. In some extreme cases, no driver is available in any of the $\overline{R}$ selected locations to complete a last-mile service, resulting in an infeasible model. To ensure feasibility, we allow customer waiting and introduce variables q_k^t and $q_{k,s}^h$ on the left-hand side of constraints (4.7e), (4.7f), which represent customers whose services have been delayed for PD pair k. To track the number of customers waiting, Eq. (4.1) is modified to include q_k^t. High waiting costs WC_k^t and WC_k^h are also imposed in the profit objective function to limit customer waiting as much as possible. We acknowledge that waiting can be inconvenient

[1] A variable can be created for each scenario, epoch, location, PD pair, and a constraint for each scenario, epoch, and PD pair.

Algorithm 1: Preprocessing Procedure

Input:
The selection parameter $\overline{R}$, and the sets $\mathcal{O}_k^t$, $\forall\, k \in \mathcal{K}^t$ and $\mathcal{O}_k^h$, $\forall\, h = t+1, \ldots, t+n_H$, $k \in \mathcal{K}_s^h$, $s \in \mathcal{S}$
for $k \in \mathcal{K}^t$ **do**
  Sort the driver origins of the set $\mathcal{O}_k^t$ in increasing order of pickup time to reach PD pair k at epoch t.
  Keep the $\overline{R}$ first driver origins of the ordered set $\mathcal{O}_k^t$, and remove the others.
  Define $\mathcal{O}_k^t(\overline{R}) \leftarrow \mathcal{O}_k^t$
for $s \in \mathcal{S}$ **do**
  for $h = t+1, \ldots, t+n_H$ **do**
    for $k \in \widehat{\mathcal{K}}_s^h$ **do**
      Sort the driver origins of the set $\mathcal{O}_k^h$ in increasing order of pickup time to reach PD pair k at epoch h.
      Keep the $\overline{R}$ first driver origins of the ordered set $\mathcal{O}_k^h$, and remove the others.
      Define $\mathcal{O}_k^h(\overline{R}) \leftarrow \mathcal{O}_k^h$
Output: The sets $\mathcal{O}_k^t(\overline{R})$, $\forall\, k \in \mathcal{K}^t$ and $\mathcal{O}_k^h(\overline{R})$, $\forall\, h = t+1, \ldots, t+n_H$, $k \in \mathcal{K}_s^h$, $s \in \mathcal{S}$.

for customers. However, our numerical experiments indicate that the impact is minimal, with less than 0.1% of last-mile services not being served upon arrival at their pickup location. It is also true that the variables q_k^t and $q_{k,s}^h$ increase complexity. Yet, the preprocessing procedure remains relevant as the number of removed variables largely outweighs the number of added variables.[2]

L-Shaped Method

While the previous procedure improves the model tractability, computation time can remain high for some instances. To address this, we next adapt the (multicut) L-shaped method to our model.

We refer to Birge and Louveaux (2011) for a detailed introduction. In short, the L-shaped method uses Benders decomposition (1962) to decompose a two-stage program into a *master problem* and *subproblems*. The master problem includes first-stage variables and constraints, along with variables θ_s to approximate second-stage costs or profits under each scenario. Each subproblem includes the second-stage variables and constraints of a scenario.

Based on this decomposition, the L-shaped method proceeds by iteration to reach the optimal solution. At each iteration, the master problem is first solved

[2] With customer waiting, the number of removed variables is equal to $|\mathcal{K}^t| \cdot (|\mathcal{O}_k^t| - \overline{R} + 1)^+ + \sum_{s\in\mathcal{S}} \sum_{h=t+1}^{t+n_H} |\hat{\mathcal{K}}_s^h| \cdot (|\mathcal{O}_k^h| - \overline{R} + 1)^+$.

to find a candidate solution for the first-stage and approximation variables. The solution to the master problem provides an upper bound (if maximizing) on the optimal objective value. Given the candidate solution, the dual of each subproblem is then solved. From the optimal dual solutions, two categories of constraints can be added to the master problem: feasibility cuts, which ensure that first-stage solutions lead to feasible subproblems, and optimality cuts, which bound the approximation variables from above. From the subproblems, a lower bound on the optimal objective value is derived. Finally, the method checks the optimality gap, computed as the difference between the upper bound provided by the master problem and the lower bound obtained from the subproblems, divided by the lower bound. If the gap is inferior to some threshold (e.g., 1%), then the algorithm stops and returns the lower bound solution. Otherwise, it repeats with updated constraints.

Applied to our specific model, the master problem at the end of the M^{th} iterations of the L-shaped method is

$$\max \sum_{k\in\mathcal{K}^t}\sum_{i\in\mathcal{O}_k^t(\overline{R})} DR_{ik}^t\cdot d_{ik}^t - \sum_{j\in\mathcal{J}^t}\Big[\sum_{i\in\mathcal{F}_j^t} SC_{ji}\cdot y_{ji}^t - \sum_{i\in\mathcal{L}_j^t} FC_{ji}\cdot z_{ji}^t\Big] - \sum_{k\in\mathcal{K}^t} WC_k^t\cdot q_k^t + \sum_{s\in\mathcal{S}}\mathbb{P}_s\cdot\theta_s \tag{4.8a}$$

$$\text{s.t. } w_j^t + x_j^t + \sum_{i\in\mathcal{F}_j^t} y_{ji}^t + \sum_{i\in\mathcal{L}_j^t} z_{ji}^t = D_j^t \quad \forall j\in\mathcal{J}^t, \tag{4.8b}$$

$$\sum_{i\in\mathcal{O}_k^t(\overline{R})} d_{ik}^t + q_k^t = L_k^{t,\,t} + x_k^t + \sum_{j,\,i\,:\,k=(o_j,\,i)} y_{ji}^t \quad \forall k\in\mathcal{K}^t, \tag{4.8c}$$

$$\sum_{k\in\mathcal{K}^t} d_{ik}^t \le N_i^{t,t} \quad \forall i\in\mathcal{I}, \tag{4.8d}$$

$$w_j^t,\, x_j^t,\, y_{ji}^t,\, z_{jl}^t \in \mathbb{Z}_0^+ \quad \forall j\in\mathcal{J}^t,\, i\in\mathcal{F}_j^t,\, l\in\mathcal{L}_j^t, \tag{4.8e}$$

$$q_k^t,\, d_{ik}^t \in \mathbb{Z}_0^+ \quad \forall k\in\mathcal{K}^t, i\in\mathcal{O}_k^t(\overline{R}), \tag{4.8f}$$

$$\theta_s \le \sum_{h=t+1}^{t+n_H}\Bigg[\sum_{j\in\widehat{\mathcal{J}}_s^h}\widehat{D}_{j,s}^h\cdot\lambda_{j,s}^{h,m} + \sum_{k\in\widehat{\mathcal{K}}_s^h}\Big(L_k^{t,h} + \sum_{j,\,i\,:\,k=(i,\,d_j)}\beta_{ji}^{t,h}\cdot z_{ji}^t\Big)\cdot\mu_{k,s}^{h,m} + \sum_{i\in\mathcal{I}}\Big(\sum_{u=t+1}^{h} N_i^{t,u} + \sum_{v=t+1}^{h}\sum_{k\in\mathcal{K}^t}\sum_{l\in\mathcal{O}_k^t(\overline{R})}\gamma_{lk}^{t,v}\cdot d_{lk}^t - \sum_{k\in\mathcal{K}^t} d_{ik}^t\Big)\cdot\nu_{i,s}^{h,m}\Bigg] \quad \forall s\in\mathcal{S},\, m=1,\dots,M, \tag{4.8g}$$

$$\theta_s \ge 0 \quad \forall s\in\mathcal{S}. \tag{4.8h}$$

Objective function (4.8a) resembles objective function (4.7a), but includes cancellation costs and replaces second-stage terms with approximation variables. Constraints (4.8b)–(4.8f) are the first-stage constraints from the original formulation (4.7a)-(4.7l). Constraints (4.8g) enforce all optimality cuts from previous iterations, with dual variables $\lambda_{j,s}^{h,m}, \mu_{k,s}^{h,m}$, and $\nu_{i,s}^{h,m}$ fixed, and computed from previous dual subproblem resolutions. No feasibility cuts are to be included, as customer waiting (cf. Section 4.5.3) ensures that the subproblems are always feasible regardless of the master solution (i.e., they have relative

complete recourse). Finally, constraints (4.8h) enforce nonnegative approximation variables.

Given a candidate solution from the master problem, the subproblem for each scenario s is defined as

$$\mathcal{Q}_s(\mathbf{d}^t, \mathbf{z}^t) \equiv \max \sum_{h=t+1}^{t+n_H} \Bigg(\sum_{k \in \widehat{\mathcal{K}}_s^h} \sum_{i \in \mathcal{O}_k^h} DR_{ik}^h \cdot d_{ik,s}^h - \sum_{j \in \widehat{\mathcal{J}}_s^h} \Big[\sum_{i \in \mathcal{F}_j^h} FC_{ji} \cdot y_{ji,s}^h - \sum_{i \in \mathcal{L}_j^h} LC_{ji} \cdot z_{ji,s}^h \Big] - \sum_{k \in \widehat{\mathcal{K}}_s^h} \sum_{i \in \mathcal{O}_k^h} WC_k^h \cdot q_{k,s}^h \Bigg) \tag{4.9a}$$

$$\text{s.t. } w_{j,s}^h + x_{j,s}^h + \sum_{i \in \mathcal{F}_j^h} y_{ji,s}^h + \sum_{i \in \mathcal{L}_j^h} z_{ji,s}^h = D_{j,s}^h \quad \forall\, h = t+1, \ldots, t+n_H,\, j \in \widehat{\mathcal{J}}_s^h, \quad (\lambda_{j,s}^h) \tag{4.9b}$$

$$\sum_{i \in \mathcal{O}_k^h(\overline{R})} d_{ik,s}^h + q_{k,s}^h = L_k^{t,\,h} + x_{k,s}^h + \sum_{j,\, i\,:\, k=(o_j,\, i)} y_{ji,s}^h + \sum_{u=t+1}^{h} \sum_{j,\, i\,:\, k=(i,\, d_j)} \beta_{ji}^{u,\,h} \cdot z_{ji,s}^u + \sum_{j,\, i\,:\, k=(i,\, d_j)} \beta_{ji}^{t,\,h} \cdot z_{ji}^t \quad \forall\, h = t+1, \ldots, t+n_H,\, k \in \widehat{\mathcal{K}}_s^h, \quad (\mu_{k,s}^h) \tag{4.9c}$$

$$\sum_{u=t+1}^{h} \sum_{k \in \widehat{\mathcal{K}}_s^u} d_{ik,s}^u \le \sum_{u=t}^{h} N_i^{t,\,u} + \sum_{u=t+1}^{h} \sum_{v=u+1}^{h} \sum_{k \in \mathcal{D}_i^u} \sum_{l \in \mathcal{I}} \gamma_{lk}^{u,\,v} \cdot d_{lk,s}^u + \sum_{v=t+1}^{h} \sum_{k \in \mathcal{D}_i^t} \sum_{l \in \mathcal{I}} \gamma_{lk}^{t,\,v} \cdot d_{lk}^t - \sum_{k \in \mathcal{K}^t} d_{ik}^t \quad \forall\, h = t+1, \ldots, t+n_H,\, i \in \mathcal{I}, \quad (\nu_{i,s}^h) \tag{4.9d}$$

$$w_{j,s}^h,\, x_{j,s}^h,\, y_{ji,s}^h,\, z_{jl,s}^h \ge 0 \quad \forall\, h = t+1, \ldots, t+n_H,\, j \in \widehat{\mathcal{J}}_s^h,\, i \in \mathcal{F}_j^h,\, l \in \mathcal{L}_j^h, \tag{4.9e}$$

$$d_{ik,s}^h,\, q_{k,s}^h \ge 0 \quad \forall\, h = t+1, \ldots, t+n_H,\, k \in \widehat{\mathcal{K}}_s^h,\, i \in \mathcal{O}_k^h. \tag{4.9f}$$

Each subproblem $\mathcal{Q}_s$ depends on variables $\mathbf{d}^t$ and $\mathbf{z}^t$, denoting drivers dispatched and customers directed to last-mile services at epoch t. In the subproblem $\mathcal{Q}_s$, the objective function (4.9a) represents the second-stage term in the objective function of the initial formulation, under the scenario s. Constraints (4.9b)-(4.9e) are the corresponding second-stage constraints in the initial formulation, organized to highlight that d_{ik}^t and y_{ji}^t are fixed. These constraints are associated with dual variables $\lambda_{j,s}^h$, $\mu_{j,s}^h$, and $\nu_{j,s}^h$. Constraints (4.9e), (4.9f) relax constraints (4.7k), (4.7l), and ensure nonnegative rather than integral variables.

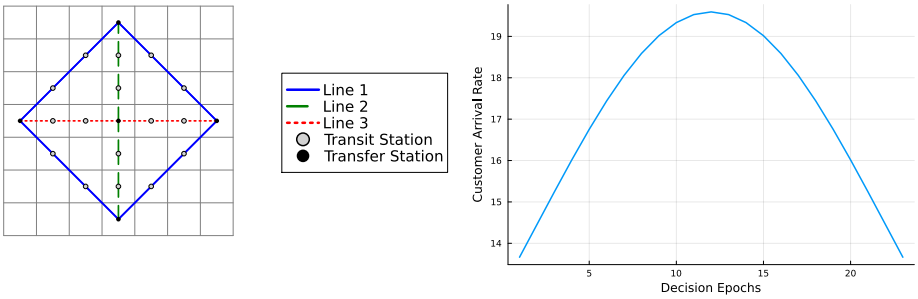
Our adaptation of the L-shaped method performs well for three main reasons. First, the candidate solutions generated by the master problem always lead to feasible subproblems. As a result, no feasibility cuts are needed, which reduces the number of iterations required before convergence. Second, each subproblem $\mathcal{Q}_s$ involves continuous rather than integer second-stage variables. This relaxation limits the computational effort per iteration, while the decisions at each epoch remain valid because the first-stage variables are still integral. Third, each subproblem $\mathcal{Q}_s$ can be viewed as an independent offline problem.

Thus, under conditions similar to those stated in Proposition 4.1 (see Appendix 4.A.2), these subproblems produce integral solutions. In our experiments, the subproblems returned optimal solutions that were mainly integral, resulting in strong optimality cuts and faster convergence.

4.6 Numerical Experiments

In this section, we conduct numerical experiments to evaluate our approach and derive managerial insights from the resulting decisions. Section 4.6.1 presents the base experimental setting. Section 4.6.2 analyzes the computational performance of the approach. Section 4.6.3 explores the potential benefits of an integrated system. Lastly, 4.6.4 applies our approach to a case study, based on real-world data from Manhattan, New York City.

4.6.1 Base Experimental Setting



(a) (b)

Figure 4.3: (a) The public transit network, and (b) the arrival rates of customer requests over time for the default instance.

To conduct our numerical experiments, we design a stylized routing network that allows easy manipulations of the model parameters. Table 4.2 summarizes the default instance, and the values used for the different parameters.

We consider a medium-sized urban area of 7 km by 7 km, discretized into 49 locations of 1 km by 1 km. Within this area, three lines operate between 21 public transit stations, including 5 transfer stations, as shown in Fig. 4.3(a). The planning horizon lasts two hours and is divided into 24 decision epochs with a time interval of five minutes in between. The speed of public transit is set to 30 km/h, the speed of customers walking to 5 km/h, and the speed of the drivers to 20 km/h, which corresponds to the speed of subways, pedestrians, and cars in many cities during peak hours (TomTom, 2023; Schwandl, 2024).

Public transit	
Number of lines	{1, 2, **3**}
Number of stations	{12, 13, **21**}
Network topology	{**kite**, cross, diamond}
Public transit speed	{20, **30**, 40, 60} km/hour
Public transit frequency	{1, **2**, 5, 10} minutes
Supply and demand	
Total number of drivers (N)	{25, **50**, 100}
Average total number of requests (D)	{200, **400**, 800}
Trip spatial pattern	{**uniform**, center to suburb, suburb to center}
Pickup fare for each trip	{0, 2.5, **4.5**}€/trip
Variable fare for each trip	{**2.7**, 3.1, 3.6}€/km
Constraints and objectives	
Minimum initial service time ($\bar{t}_1$)	{**10**, 15, 20} minutes
Maximum additional travel time ($\bar{t}_2$)	{0, 25, **50**} %
Minimum discount ($\bar{t}_3$)	{**0**, 5, 10}€
Objective weight parameter (w)	$\{0, 0.2, 0.4, 0.6, 0.8, \mathbf{1}\}$

Table 4.2: Characteristics of the instances. The default instance is written in bold.

When traveling by car, the travel times between locations are based on Manhattan distances (i.e., the sum of the absolute differences of their coordinates). When traveling by public transit, the travel times between stations are based on Euclidean distances. The transit frequency is equal to 2 minutes, similar to the subway frequency in cities such as Berlin (Civitatis, 2024), or Madrid (Madrid Tourism Office, 2024). The travel times using public transit reflect that customers endure an average walking and dwell time of 6 minutes to reach public transit, an average time of 1 minute waiting for transit, and an average transfer time of 4 minutes to change lines. On the network, the platform operates 50 drivers with randomly distributed initial locations. The origins and destinations of requests are uniformly drawn across the locations, leading to an average distance of roughly 5 km per trip. To replicate demand conditions during peak hours, customer requests arrive following a non-stationary Poisson process. The arrival rates are adjusted as shown in Fig. 4.3(b). These adjustments result in a maximum increase of about 50% of requests. On average, a total of 400 requests arrive over the horizon, which is roughly equal to the supply capacity.

In the default setting, the platform only maximizes profit and neglects customer travel time, i.e., $\alpha = 1$ in the weighted objective $\max Obj^{on} = \alpha \cdot Profit^{on} - (1-\alpha) \cdot Time^{on}$. The reward for serving each request is computed to approximate the fare structure employed by taxi companies in Europe. Taking an average among Western European cities, it consists of a pickup fare of 4.5 € and a variable fare of 2.7 €/km. Similarly, the first-mile and last-mile costs reflect subway ticket prices in Western Europe (Numbeo, 2024), and are set to 2.4 €. We assume that all customers are willing to be directed to public

transit, i.e., $\delta = 1$. Customers can be directed to first-mile and last-mile transit routes only if their door-to-door travel time exceeds 10 minutes. Every time a customer is directed to public transit, the increase in travel time must stay lower than 50 %. The new service price must be lower than the initial service price. Lastly, the maximum pickup time is equal to 10 minutes.

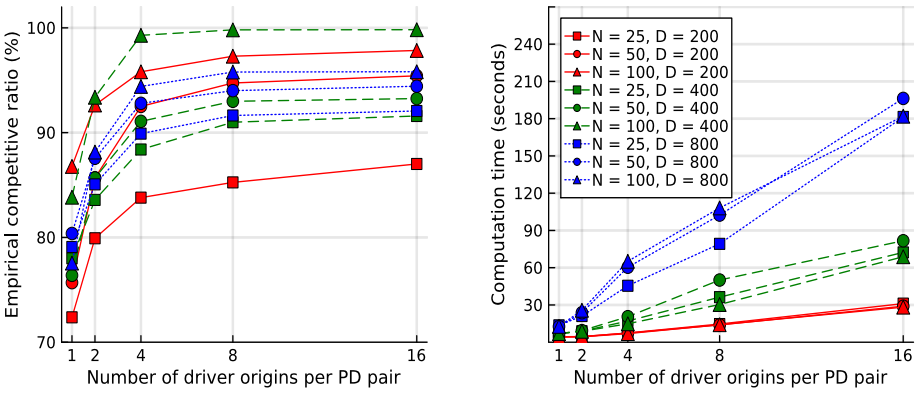
The computations are performed on Julia and rely on computers with 3.7 GHz AMD Epyc CPU cores and 32GB of RAM. The stochastic program is solved using the programming language Julia, the package StochasticPrograms.jl (Biel & Johansson, 2022), and the solver Gurobi. For each parameter configuration, we generate 30 test instances with random initial driver locations and customer requests to ensure the reliability of the results. Scenarios are generated independently of each instance, though they are generated using the same demand distribution. To simplify exposition, we omit the confidence intervals from subsequent results. The 95% confidence intervals typically range between 2% and 5% of the average values. We set the lookahead horizon of the two-stage program to 30 minutes (i.e., $n_H = 6$) by default as more than 90% of the trips are completed within this period.

4.6.2 Approach Performances

In this section, we evaluate the performance of our approach in terms of solution quality and computational effort. We first investigate the trade-off between high final performance and short computation time. Then, we demonstrate the relevance of our approach by comparing it with alternative baselines.

To evaluate performance, the analysis relies on two metrics. The first metric is the empirical competitive ratio, which is the average ratio between the objective value produced by our approach, and the objective value deduced from the offline problem, solved at the end of the planning horizon, once all requests have been revealed. The second metric is the average computation time, which refers to the average time taken by our approach to solve the optimization program at each epoch.

Fig. 4.4 displays the evolution of the empirical competitive ratio (a) and average computation time (b) for instances with various levels of supply and demand, as the number of driver origins $\overline{R}$ per PD pair varies (cf. Section 4.5.3). In Fig. 4.4(a), we observe that, for any level of demand and supply, the empirical competitive ratio is substantially lower when the number of selected origins is low (e.g., around 80% when $\overline{R} = 1$). As the number of driver origins increases, the empirical competitive ratio exhibits a similar behavior: it initially improves rapidly and then stabilizes once a sufficient number of driver origins are selected. On the other hand, Fig. 4.4(b) indicates an opposite tendency: the average computation time rises at an increasing rate as the number of selected driver origins grows. These two trends show the trade-off between solution quality and computational efficiency. Our approach only needs to consider a few driver origins for each PD pair (e.g., $\overline{R} = 8$) to find solutions close to those with unlimited driver origins, while reducing computational effort substantially.



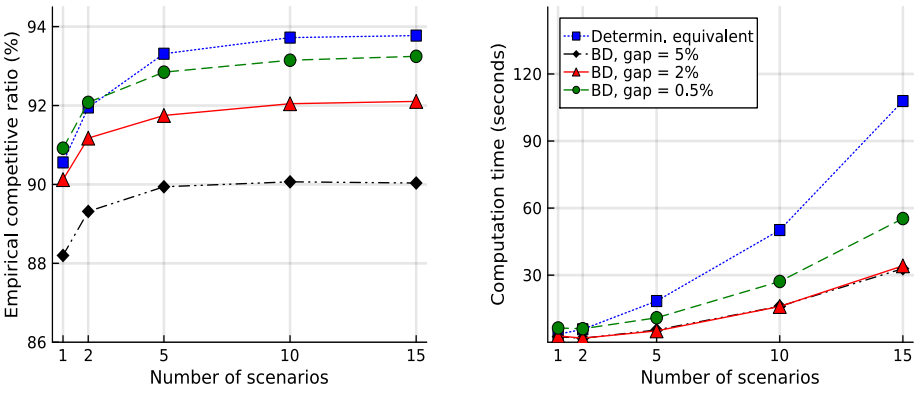
(a) (b)

Figure 4.4: Impact of changing the number of driver origins $\overline{R}$ per PD pair on the empirical competitive ratio (a) and computation time (b) for various levels of driver supply (N) and customer demand (D). The stochastic program is solved using the deterministic equivalent formulation with $n_S = 10$ scenarios.

Fig. 4.5 depicts the evolution of the empirical competitive ratio (a) and computation time (b) when the stochastic program is solved using the deterministic equivalent formulation, or the L-shaped method with various gap parameters. For each configuration, the performances are plotted as a function of the number of scenarios considered at each re-optimization.

We observe in Fig. 4.5(a) that the L-shaped method performs close to the deterministic equivalent, especially when the gap is low (e.g., less than 1% difference when gap = 0.5%). Also, only a limited number of scenarios appears to be necessary to capture uncertainty. In Fig. 4.5(b), the L-shaped method leads to lower computation times than the deterministic equivalent for any gap value, as soon as the number of scenarios becomes significant (e.g., when $S \geq 5$). The gains in computation times tend to be more pronounced as the number of scenarios increases. Overall, these results demonstrate that the L-shaped method allows for reducing computation times at the expense of a minor loss of solution quality. The L-shaped method has some practical value, as one can adjust the gap parameter depending on the setting, in order to balance computation time with solution quality.

Table 4.3 reports the empirical competitive ratio for our approach under three configurations, and for two alternative baselines, across various levels of driver supply and customer demand. The first configuration of our approach solves the deterministic equivalent formulation (DEF) with 15 scenarios, while the second and third use the L-shaped method (LS) with 15 and 10 scenarios, and a 0.5% optimality gap. The first baseline is a rolling horizon strategy that optimizes a single-stage model at each epoch, based on the observed demand.



(a) (b)

Figure 4.5: Comparison between the deterministic equivalent formulation, and the L-shaped method for several maximum optimality gaps, in terms of empirical competitive ratio (a) and computation time (b). The number of selected service routes is set to $k = 8$.

The second baseline is a lookahead approach that optimizes a multi-period program at each epoch, based on observed and expected future demand. Both baselines can be seen as special cases of our approach: the first corresponds to setting $H = 0$, and the second to solving a program with one expected value scenario, i.e., $n_S = 1$.

Compared to the rolling horizon strategy, our approach almost always achieves improved performance, often by more than 20 percentage points. The most significant gains arise when supply is limited. In the latter case, the rolling horizon strategy performs poorly because it fails to anticipate future requests and quickly exhausts its available drivers. In contrast, our approach proactively dispatches drivers while considering future customers. As expected, the rolling horizon strategy performs reasonably well when supply is abundant, and a myopic approach may even be more relevant in these situations (see, e.g., when $D = 200$ and $S = 100$).

Compared to the expected-value lookahead approach, our approach improves performance by only small margins on average (a difference of 1.1% to 1.8%). This result shows the value of incorporating uncertainty into our model, though it appears less critical than anticipating future customers. We note that the benefits are more pronounced under low demand and supply density (e.g., 4 more percentage points when $D = 200$ and $N = 25$), or when supply is limited (e.g., 2.9 more percentage points when $D = 800$ and $N = 25$). From these results, we recommend using an expected-value lookahead approach in high-density environments, especially when supply and demand are balanced. Multiple scenarios become relevant in low-density settings, when supply is scarce.

		Our approach - Stochastic LH			Baselines	
Demand	Supply	DEF, $n_S = 15$	LS, $n_S = 15$	LS, $n_S = 10$	Myopic	Expected-value
200	25	85.6%	**86.1**%	86.0%	70.3%	82.1%
	50	94.7%	**96.5**%	96.4%	90.9%	94.1%
	100	97.0%	98.0%	98.2%	**99.9**%	97.1%
400	25	**91.0%**	89.4%	89.2%	62.6%	88.1%
	50	**93.3%**	92.2%	91.8%	67.2%	90.8%
	100	**99.8%**	**99.8%**	**99,8%**	92.0%	99.7%
800	25	**91.7%**	90.0%	89.9%	57.2%	88.8%
	50	**94.2%**	92.9%	92.9%	60.4%	92.3%
	100	**95.8%**	95.0%	95.0%	65.7%	94.3%
Average		93.8%	93.2%	93.1%	74.3%	92.0%

Table 4.3: Comparison of our stochastic lookahead horizon approach with myopic rolling-horizon and expected-value lookahead approaches in terms of empirical competitive ratio, for different demand and supply levels. The best result for each supply and demand level is written in bold.

4.6.3 Value of the Integrated System

This section investigates the value of integrating public transit into ride-hailing dispatching. We first analyze the benefits as a function of the platform's objective, looking at the Pareto frontier between profit maximization and travel time minimization. Then, we explore various parameter configurations, quantifying the benefits for multiple instances.

In our analysis, we benchmark our initial setting (FM-LM), with three alternative settings that isolate first-mile, last-mile, and door-to-door services. The first setting (FM) only allows the platform to offer first-mile and door-to-door services to customers. The second setting (LM) considers the opposite case, where last-mile and door-to-door services are provided to customers. Lastly, the third setting (NPT) only allows the platform to deliver door-to-door services to customers, without any public transit integration. We analyze the decisions obtained by our approach, using the L-shaped method with a gap of 0.5%, a number of scenarios $n_S = 10$, and a number of selected driver origins $\overline{R} = 8$.

Analysis of Pareto Frontiers

To evaluate the benefits of an integrated system, we first analyze the trade-off between profit maximization and travel time minimization, focusing on the default instance. The analysis is carried out by adjusting the weight parameter α in the objective function $\max Obj^{on} = \alpha \cdot Profit^{on} - (1 - \alpha) \cdot Time^{on}$. By doing this, we generate a set of efficient solutions, forming a Pareto frontier between profit and travel time for each setting.

Fig. 4.6 displays the objective values along the Pareto frontier for each of

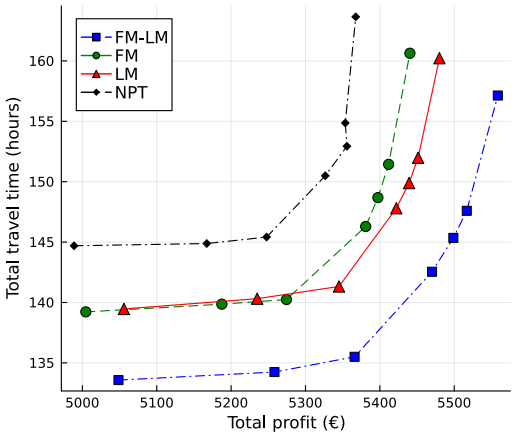


Figure 4.6: Trade-off between travel time minimization and profit maximization for the four settings. Each point represents a different value for the weight α.

the four settings, as we vary the weight parameter α from 0 to 1. The results for each point are averaged over 30 simulations of the default instance.

From Fig. 4.6, we can make three observations. First, as expected, FM-LM consistently improves the objective value compared to all alternative settings, for any weight parameter. However, comparing FM-LM with NPT, we observe a significant decrease in total travel time (from 7.6% when $\alpha = 0$ to 3.9% when $\alpha = 1$), while the increase in total profit is more modest (from 1.2% when $\alpha = 0$ to 3.6% when $\alpha = 1$). This result suggests that an integrated system can benefit both the platform and its customers, although customers are likely to be the primary beneficiaries.

Second, looking at the slope of the Pareto frontier for FM-LM, we note that the trade-off between profits and travel times occurs at intermediate values of α. On the contrary, when α is low or high, one objective can be improved without substantially affecting the other. This result suggests that low travel time and high profit can be jointly achieved. Rather than optimizing a single objective, a ride-hailing platform should seek a compromise between the two objectives.

Third, comparing FM with LM, we observe that LM always achieves higher profits than FM (1.2% higher on average). This result happens because customers are impatient. The platform therefore uses last-mile services to postpone some customer requests to epochs with more drivers. In contrast, first-mile service requires customers to be served at the current epoch. In our case, the effect is amplified due to our assumption that customers leave if they are not served at the current epoch. In practice, the effect may still occur as customers can exhibit impatient behaviors.

Table 4.4 complements Fig. 4.6 by reporting additional results across the

	$\alpha = 0$				$\alpha = 0.2$			
	FM-LM	FM	LM	NPT	FM-LM	FM	LM	NPT
Service rate (%)	83.6	75.9	76	67.5	85.4	78.8	78.4	70.6
Transit rate (%)	34.6	19.8	19.4	0	30.4	17.9	18.9	0
Avg. cust. travel time (minutes)	19.5	20.4	20.5	21.1	20	20.7	20.8	21.4
Avg. cust. service time (minutes)	8.4	8.7	8.9	9.2	9.1	9.3	9.5	9.7
Avg. cust. transit time (minutes)	6.6	6.2	6.4	5.9	5.8	5.5	5.3	5.1
Avg. cust. walking time (minutes)	1.8	2.9	2.7	3.7	2.2	3	3.3	3.9
Price per service (€)	14.8	16.1	16.3	18	15.3	16.5	16.6	18.3
Driver utilization rate (%)	54.6	57.2	57.8	60.4	58.8	61.1	61.5	63.9
	$\alpha = 0.6$				$\alpha = 1$			
	FM-LM	FM	LM	NPT	FM-LM	FM	LM	NPT
Service rate (%)	83.1	77.2	78.1	70.4	81	76	77	70.2
Transit rate (%)	29.6	16.6	19.4	0	27.7	15.6	18.1	0
Avg. cust. travel time (minutes)	21.6	22.3	22.2	22.7	23.1	23.6	23.6	24.1
Avg. cust. service time (minutes)	9.6	9.7	9.8	10	9.7	9.8	9.8	10
Avg. cust. transit time (minutes)	5.3	5.1	4.9	4.5	5.3	5	4.8	4.4
Avg. cust. walking time (minutes)	3.6	4.3	4.6	5.3	4.5	5.3	5.4	6.2
Price per service (€)	16.2	17	17.1	18.5	16.6	17.3	17.3	18.6
Driver utilization rate (%)	62.3	63.4	63.5	65.1	62.9	63.6	63.7	65.5

Table 4.4: Performance metrics for the four settings, under different weight values.

four settings for varying weight values. Specifically, the service rate measures the percentage of customers served by the platform. The transit rate is the proportion of served customers who receive first-mile or last-mile services. The average customer travel time refers to the mean time that customers take to reach their destination, regardless of whether the platform serves them. The average service time refers to the time spent in vehicles, the average transit time to the time spent in public transit, and the average walking time to the time spent walking during the journey. The price per service captures the average fee paid by a customer. Finally, the utilization rate is the proportion of time that drivers are busy transporting requests.

Table 4.4 confirms the benefits of an integrated system for customers. First, FM-LM achieves a substantially higher service rate than any alternative setting. The improvement, ranging between 10.8 and 16.1 percentage points compared to NPT, is due to the ability of FM-LM to direct a significant share of the requests to public transit. Second, FM-LM significantly reduces customer travel time (between 0.9 to 1.7 minutes) in comparison with the alternative setting. This reduction is also obtained by shifting customers toward public transit (as reflected by the increased transit time), which leads to lower service and walking times. Third, the price paid by the customers decreases, with a reduction from 2 € to 3.2 €. All in all, these results show that customers enjoy more accessible, faster, and more affordable services.

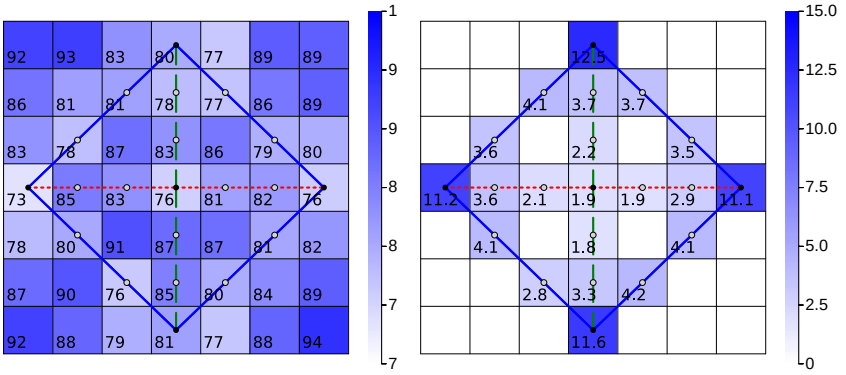
It can also be noticed that FM-LM decreases driver utilization by 2.6 to 5.8 percentage points compared to NPT. This decrease in utilization may seem surprising at first, as FM-LM serves more customers than NPT. The reason is that many satisfied requests are directed to public transit for part of the trip, resulting in shorter service times. As fewer drivers are required to serve more customers, our results indicate that FM-LM has the potential to reduce car usage, and thus congestion and carbon emissions. At the same time, this result can also reveal that the integration could negatively impact the earnings of drivers, since their earnings are directly proportional to their utilization. This will largely depend on the business model set up by the platform. If the platform manages drivers as employees, then the integration would be beneficial for drivers. If the drivers are independent contractors, then drivers would benefit from the integration only if the platform adopts compensation schemes that fairly share the additional profits.

To better understand the trade-off between profit and travel time, we next examine the platform's decisions. Fig. 4.7 plots the service rate as a function of the origin location, and the proportion of customers dropped off or picked up in each location when receiving first-mile or last-mile service, for both objectives.

Fig. 4.7 reveals important structural differences in the platform's decisions. On the one hand, when travel time is minimized, the platform mainly acts as a feeder system. The platform tends to provide first-mile and last-mile service to customers starting or finishing their journey far from transit stations, while customers near transit stations are less likely to be served (cf. Fig. 4.7(a)). In that case, customers who receive first-mile and last-mile services are frequently picked up and dropped off at peripheral transit stations, and complete the remaining leg via public transit (cf. Fig. 4.7(b)). On the other hand, when profit is maximized, the platform leverages the existing public transit network to improve its dispatching decisions. The platform favors customers in central locations, already close to transit stations (cf. Fig. 4.7(c)). Customers who receive first-mile and last-mile services tend to be dropped off and picked up in the transit stations at the center, as it allows for concentrating supply and demand in the same locations, improving the platform's profit (cf Fig. 4.7(d)).

Analysis of Parameter Variations

This section aims to evaluate factors that impact the integration benefits. To this end, we alter various parameters of the default instance. The modifications concern (i) the quality constraints of first-mile and last-mile services, (ii) supply and demand parameters, and (iii) public transit parameters. For each alteration, the benefits of the setting with integration (FM-LM) are measured relative to the setting without integration (NPT), both in terms of profit and travel time. For clarity, the analysis only considers the extreme cases where the platform minimizes travel time, or maximizes profit. The service rate and transit rate in the integrated setting are also reported to provide additional information.



(a) min $Time^{on}(\alpha = 0)$ (b) min $Time^{on}(\alpha = 0)$

(c) max $Profit^{on}(\alpha = 1)$ (d) max $Profit^{on}(\alpha = 1)$

Figure 4.7: Service rate as a function of the customer origin (a,c) and percentage of all the customers receiving first-mile and last-mile services that are dropped off or picked up (b,d) in each location for the different objectives.

Impact of Quality Constraints

We first evaluate the influence of quality constraints, introduced in Section 4.3.2, on the benefits of the integrated system. To this end, Table 4.5 summarizes results for instances with increasingly restrictive quality constraints.

We observe that the integrated setting continues to bring benefits, although the improvements decrease as quality constraints get stricter. In particular, significant reductions in travel time persist under tight constraints, while profit gains become limited. From these results, a decision-maker may conclude that strict quality constraints can hinder the profit gains of public transit integration for the platform. In such cases, the main motivation for the platform to implement an integrated system would be to improve the service quality of customers, while remaining profit-neutral.

	min $Time^{on}(\alpha = 0)$			max $Profit^{on}(\alpha = 1)$		
Min. initial service time $\bar{t}_1$ (minutes)	10	15	20	10	15	20
Profit change vs. NPT (%)	**1.2**	0	-0.7	**3.6**	2.5	1.2
Travel time change vs. NPT (%)	**-7.6**	-7.2	-4.1	**-3.9**	-3.2	0
Service rate (%)	**83.6**	80.8	73.3	**81**	78.2	72.5
Transit rate (%)	**34.6**	28.4	13.4	**27.7**	21	8.8
Max. additional travel time $\bar{t}_2$ (%)	50	25	0	50	25	0
Profit change vs. NPT (%)	**1.2**	-0.1	-0.7	**3.6**	2.5	0.6
Travel time change vs. NPT (%)	**-7.6**	-7.1	-3.1	**-3.9**	-4.9	-1.8
Service rate (%)	**83.6**	81.1	72	**81**	78.4	72.3
Transit rate (%)	**34.6**	29.6	10.8	**27.7**	19.3	4.6
Min. discount $\bar{t}_3$ (€)	2.4	5	10	2.4	5	10
Profit change vs. NPT (%)	**1.2**	0.4	-0.8	**3.6**	1.8	0.9
Travel time change vs. NPT (%)	**-7.6**	-7.5	-6.4	**-3.9**	-5.6	-4.8
Service rate (%)	**83.6**	83.2	80.7	**81**	80.6	77.8
Transit rate (%)	**34.6**	33.7	26.5	**27.7**	22.7	14

Table 4.5: Performance for various service quality constraints. Results of the default instance are written in bold.

Impact of Demand and Supply Parameters

Next, we examine the impact of demand and supply parameters on the benefits of the integration. We start the analysis with Table 4.6, which depicts results for nine different levels of demand and supply under the two objectives.

Table 4.6 shows that the benefits of the integration can vary depending on the levels of supply and demand. When demand is low relative to supply, the integrated setting brings negligible benefits, both in terms of profit and travel time. In these cases, all customers can be served with door-to-door services, and there is no need to direct them to public transit. As the demand rises, the profit gains increase under both objective functions, largely due to the pickup fare (€4.5/service), which favors serving more, shorter trips. In contrast, the travel time benefits first increase, but then decline as demand becomes abundant. This effect is due to a saturation of the supply capacity. When demand is excessive, drivers are fully utilized, and most requests remain unsatisfied even with public transit integration.

We can draw three managerial insights from Table 4.6. First, the integration may not always be beneficial. When demand is low, the implementation of an integrated system is not justified because all customers can receive door-to-door services, which are more profitable and faster. Second, when demand is high, the integration leads to important profit gains for the platform. In these cases, the platform could use an integrated system to serve more requests. Third, when demand is intermediate, the integration generates high travel time gains for customers. An integrated system could be employed for customers to enjoy reduced travel time during periods with moderate demand.

	min $Time^{on}(\alpha = 0)$			max $Profit^{on}(\alpha = 1)$		
Demand (D)	200					
Supply (N)	25	50	100	25	50	100
Profit change vs. NPT (%)	1.1	−8	−7.1	2.8	0.2	0
Travel time change vs. NPT (%)	−6.3	−2.9	−0.1	−2	0.2	0
Service rate (%)	75.8	99.3	100	72.8	99.8	100
Transit rate (%)	35.9	17.1	10.1	30.9	4.6	0
Demand (D)	400					
Supply (N)	25	50	100	25	50	100
Profit change vs. NPT (%)	4.1	**1.2**	−0.8	5.4	**3.6**	0.1
Travel time change vs. NPT (%)	−4	**-7.6**	−0.8	−2.3	**-3.9**	0
Service rate (%)	49.4	**83.6**	99.8	51.3	**81**	100
Transit rate (%)	46.5	**34.6**	13	44.3	**27.7**	0.6
Demand (D)	800					
Supply (N)	25	50	100	25	50	100
Profit change vs. NPT (%)	4.1	5.0	1.0	8.1	6.5	3.4
Travel time change vs. NPT (%)	−2.2	−5.1	−8.3	−1.5	−4.2	−5.6
Service rate (%)	30.2	53.6	90.5	35.2	58.3	88
Transit rate (%)	55.7	46.1	32.6	55.7	45.3	27.4

Table 4.6: Performance for various levels of demand and supply. Results of the default instance are written in bold.

To further investigate the impact of revenue parameters, Table 4.7 displays results for instances with different base and variable fares under profit maximization. The pricing scheme significantly affects profit gains in the integrated setting. As the pickup fare decreases, profit benefits drop from 3.6% to 1.2%, showing that most of the additional profit is due to the collected pickup fares. The platform still achieves a slight improvement with no pickup fare, but this requires directing many customers to public transit (cf. the transit rate of 22.9%) while the service rate increases only marginally (cf. the service rate of 70.2% under NPT in Table 4.5). We therefore conclude that, in the absence of a pickup fare, the platform has little financial incentive to adopt the integration, as the profit gains are likely to be limited.

Impact of Public Transit Parameters

Table 4.8 presents results on the impact of varying the transit speed, and frequency. Both factors significantly influence the benefits of an integrated system. As transit becomes faster and more frequent, integration generates greater benefits, suggesting that transportation means such as subways or buses are particularly suited for integration with ride-hailing. Moreover, frequency appears to have a stronger effect than speed. For instance, under travel time minimization, the transit rate drops from 35.6% to 18.5% as frequency decreases, compared to a decline from 39% to 28.9% when speed decreases. This is be-

	$\max Profit^{on}(\alpha = 1)$		
Pricing scheme (€)	$4.5 + 2.7\,/\,\text{km}$	$2.5 + 3.1\,/\,\text{km}$	$0 + 3.6\,/\,\text{km}$
Profit change vs. NPT (%)	**3.6**	2.4	1.3
Travel time change vs. NPT (%)	**-3.9**	−3.3	−1.3
Service rate (%)	**81**	76.6	71
Transit rate (%)	**27.7**	25.8	22.9

Table 4.7: Performance for various pricing schemes and first-mile and last-mile costs prices under profit maximization. Results of the default instance are written in bold.

	$\min Time^{on}(\alpha = 0)$				$\max Profit^{on}(\alpha = 1)$			
Frequency (minutes)	1	2	5	10	1	2	5	10
Profit change vs. NPT (%)	1	**1.2**	1.4	0.9	3.3	**3.6**	2.7	1.6
Travel time change vs. NPT (%)	−8.5	**−7.6**	−5.4	−3	−4.8	**−3.9**	−3.2	−1.8
Service rate (%)	84	**83.6**	81.4	76.5	81.2	**81**	79.6	76.3
Transit rate (%)	35.6	**34.6**	29.5	18.5	29.2	**27.7**	26.4	17.2
Speed (km/h)	20	30	40	60	20	30	40	60
Profit change vs. NPT (%)	1.3	**1.2**	1.2	1.6	2.5	**3.6**	3.7	4.1
Travel time change vs. NPT (%)	−5	**−7.6**	−9	−10.7	−3	**−3.9**	−5	−4.6
Service rate (%)	81	**83.6**	84.5	85.2	78.8	**81**	81.7	82
Transit rate (%)	28.9	**34.6**	37	39	26.1	**27.7**	29.7	31.6

Table 4.8: Performance for various public transit speeds and frequencies. Results for the default instance are in bold.

cause first-mile and last-mile services typically cover short distances, making waiting time a relatively large portion of the total time spent in public transit. For customers directed to public transit, a high transit frequency is thus key to reducing travel times. Based on this finding, a decision-maker might infer that integration is more likely to be relevant when the public transit system operates at high frequency rather than high speed.

Results for different network topologies are also reported in Table 4.9. The first topology corresponds to the default instance. The second only includes Line 1 (cf. Fig. 4.3(a)), and is referred to as the "diamond" topology. The third, named the "cross" topology, only keeps Line 2 and Line 3. The fourth, the "web" topology, adds three lines: a circular line circling around the service area, and two diagonal lines crossing at the center.

By comparing the diamond and cross topologies with the default topology, we see that the profit and travel time benefits decrease. As expected, the lower presence of public transit reduces the reach of an integrated system, as the platform can direct fewer customers to first-mile and last-mile services. For example, under travel time minimization, the average transit rate of the diamond and cross topology falls to 22.3% and 23.5%, respectively. By contrast, the average transit rate of the default topology is equal to 34.6%.

By comparing the web topology with the default topology, we observe that

	min $Time^{on}(\alpha = 0)$			
Network topology	Default	Diamond	Cross	Connected
Profit change vs. NPT (%)	**1.2**	1.1	1.5	3.5
Travel time change vs. NPT (%)	**−7.6**	−4.9	−4.8	−5.3
Service rate (%)	**83.6**	78.9	77.9	97
Transit rate (%)	**34.6**	22.3	23.5	47.5
	max $Profit^{on}(\alpha = 1)$			
Network topology	Default	Diamond	Cross	Connected
Profit change vs. NPT (%)	**3.6**	1.8	1.9	8.8
Travel time change vs. NPT (%)	**−3.9**	−3.1	−2.6	0.1
Service rate (%)	**81**	76.7	76.7	93.1
Transit rate (%)	**27.7**	17.8	21.2	48.1

Table 4.9: Performance for various public transit topologies. Results for the default instance are in bold.

the profit gains rise remarkably (e.g., from 3.6% to 8.8% under profit maximization). This result demonstrates the profitability of integration when there is wide public transit access. Looking at the service and transit rates, we note that the platform achieves this result by serving almost all customer requests, with many of them directed to public transit for part of the trip. In contrast, travel time benefits are lower for the web topology (e.g., from −7.6% to −5.3% under travel time minimization). When transit access is limited for some customers (as in the default topology), ride-hailing services are essential to prevent long travel times for these customers, making integration highly valuable. When transit access is high for all customers (as in the web topology), even unsatisfied requests incur short travel times, reducing the travel time benefits of integration.

Three managerial insights emerge from Table 4.9. First, an integrated system only becomes beneficial if there is sufficient access to public transit in the service area. Second, the profit benefits are highest when the whole service area has widespread access to public transit, as many more requests can be served. Third, the travel time benefits are highest when the service area is characterized by unequal access to public transit, as the integrated system connects remote customers to public transit. In view of the last two insights, a platform aiming to maximize profit should prioritize integration in areas with high access to public transit, while a platform aiming to reduce customer travel time should prioritize integration in areas with unequal access to public transit.

4.6.4 Case Study in New York City

In this section, we demonstrate the practicality of our approach by applying it to real-world data from New York City (NYC). Before presenting numerical results, we first describe the data supporting the study and associated parameters.

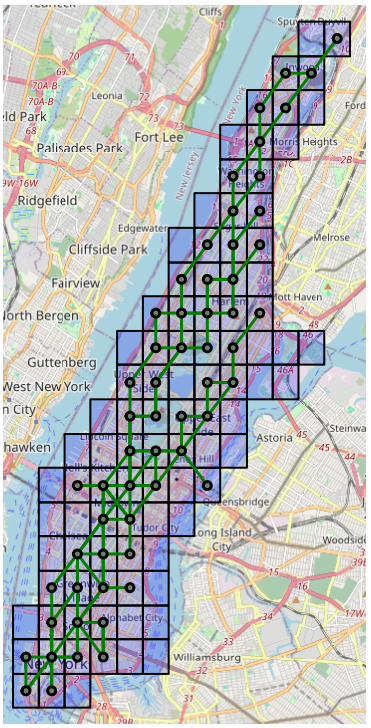


Figure 4.8: Partition of Manhattan in 94 locations with public transit lines connecting 57 transit stations.

Data and Parameters

The study uses publicly available data from the NYC Taxi and Limousine Commission (2016), which reports NYC taxi trips that occurred in June 2016. For each trip, the dataset includes start and end times, pickup and drop-off points, distance, and fare paid by the customer. We curate the dataset by removing trips with missing or inconsistent values and trips that start or end outside Manhattan. Furthermore, to limit the scope of the study, only trips for the first 16 weekdays during morning peaks from 7 a.m. to 9 a.m. are retained. The first 15 days are employed for scenario construction, while the remaining day is used for approach evaluation. The final dataset contains $156,564$ trips.

As shown in Fig. 4.8, Manhattan is modeled as a routing network of 94 locations, where each location represents an area of 1 square km. To fit trips within the network, origins and destinations are projected to the center of their nearest location. The planning horizon is divided into 40 decision epochs, with intervals of 3 minutes. Revenues are computed from fares recorded in the dataset, or using the fare model of NYC (Woodhouse, 2022), that is, a pickup fare of \$2.5 and a variable fare of \$1.5/mile. Reflecting Manhattan traffic

conditions (TomTom, 2023), driver travel times and customer walking times between locations are deduced by assuming average speeds of 20 km/h and 4 km/h, respectively. The number of drivers is balanced with customer demand, and set to about 800. Drivers are distributed across locations following the demand distribution during the first thirty minutes of the planning horizon.

The public transit system is built using open data from NYC Transit (2025), which provides GPS coordinates for all subway stations. We simplify the setting by allowing at most one station per location, keeping only the station closest to the location center (if any). The resulting transit network forms a graph with stations as nodes and subway lines as edges. Customer travel time in the transit network accounts for a walking time of 7.5 minutes to reach the station platform, a frequency of 3 minutes (MTA, 2025), and a speed of 30 km/h (Johnson, 2010). Service quality constraints are the same as in the base setting, but the adoption rate δ varies. First-mile and last-mile costs are not included, assuming that customers who are willing to get first-mile and last-mile services have transit subscriptions. This assumption is not far from reality, as a large share of customers in NYC benefit from automatic weekly subscriptions (OMNY, 2025).

Results and Analysis

We now apply our approach to the case study, and analyze numerical results. As the case study has high supply and demand densities,[3] we follow the recommendation made in Section 4.6.2 and use an expected-value lookahead model with a single scenario instead of a stochastic lookahead model with multiple scenarios. Multiple scenarios can be included, but yield little improvement over a single expected-value scenario, while requiring more computational effort.

Table 4.10 displays the profit and travel time improvements allowed by the integrated setting (FM-LM) compared to the setting without integration (NPT), as the adoption rate δ varies. In addition, Fig. 4.9 plots the service rate in the integrated setting (FM-LM) and the setting without integration (NPT). These results are given for two weight values to represent the cases where the platform favors profit over travel time ($\alpha = 0.6$), and conversely ($\alpha = 0.2$).

[3]In the test instance, more than 8000 requests are served by about 800 drivers.

	Objective weight $\alpha = 0.2$						
Adoption rate δ	0	0.2	0.4	0.5	0.6	0.8	1
Profit change vs. NPT (%)	0	3.7	4.7	**4.8**	4.9	5.2	5.2
Travel time change vs. NPT (%)	0	−2.9	−4.1	**−4.4**	−4.7	−5.2	−5.5
	Objective weight $\alpha = 0.6$						
Adoption rate δ	0	0.2	0.4	0.5	0.6	0.8	1
Profit change vs. NPT (%)	0	4.4	5.7	**6.2**	6.7	7.0	7.4
Travel time change vs. NPT (%)	0	−0.6	−0.9	**−0.8**	−0.7	−1.1	−0.9

Table 4.10: Performance for various adoption rates δ and weight values α in the objective function. Results for the default instance are in bold.

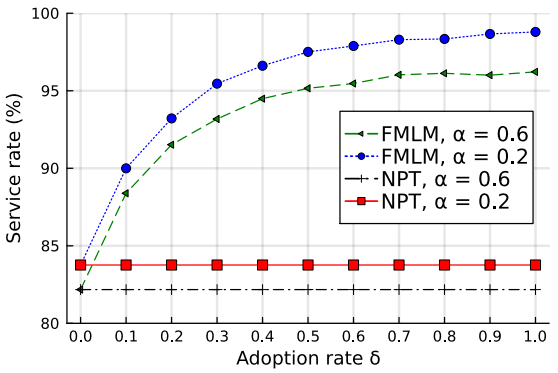
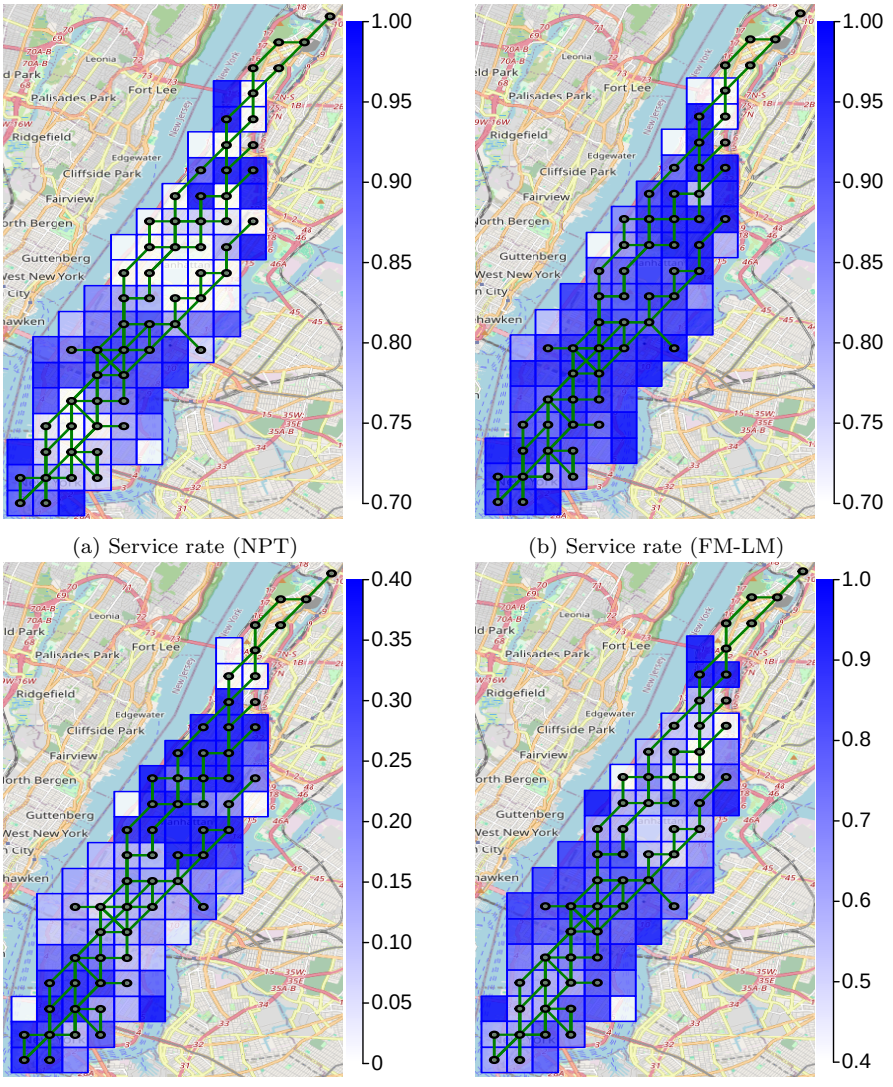


Figure 4.9: Service rate as a function of the customer adoption rate δ and weight value α for the integrated setting (FM-LM), and the setting without integration (NPT).

Table 4.10 and Fig. 4.9 reveal that an integrated system can lead to high benefits in practice, even with a low adoption rate. We observe in Table 4.10 that the integrated setting improves profit by up to 7.4% ($\alpha = 0.6$), compared to the setting without integration. In addition, travel times can be significantly reduced in the integrated setting, up to 5.5% ($\alpha = 0.2$). While profit benefits are obtained anyway, travel time benefits are only achieved when the platform is primarily concerned with travel time minimization (compare the travel time change when $\alpha = 0.2$ and when $\alpha = 0.6$). Finally, in Fig. 4.9, the service rate improves by up to 15 percentage points in the integrated setting, e.g., from 83% to 98% when the objective weight $\alpha = 0.2$.

Several factors can be invoked to explain these results. First, because we assume that customers already hold public transit subscriptions, the costs incurred by the platform for first-mile and last-mile services are absent from the problem. As a result, the increase in revenue translates more directly into higher profit, without being offset by additional costs. Furthermore, as discussed in Section 4.6.3, the profit improvement of integration tends to be substantial when (i) supply is limited relative to demand, (ii) the pickup fare



(a) Service rate (NPT)

(b) Service rate (FM-LM)

(c) Transit rate (FM-LM)

(d) Door-to-door rate (FM-LM)

Figure 4.10: Service rate (a,b) for the setting without (NPT) and with public transit integration (FM-LM), and transit rate (c) and door-to-door rate (d) for the setting with integration, in each location. Locations that are depicted are those where at least one customer request is observed (a, b), or served (c, d) during the planning horizon. The adoption rate and the objective weight are set to $\delta = 0.5$ and $\alpha = 0.6$.

is high relative to the variable fare, and (iii) the public transit system has a high frequency and a high coverage of the service area. Similarly, travel time benefits are expected to be high when (i) demand is moderately high, and (ii) the public transit system has high frequency. All these features are present in this case study, which explains the high benefits. From this discussion, we conclude that the insights from the previous section can be valuable to predict the magnitude of the integration benefits.

To further understand the value of the integrated setting, Fig. 4.10 plots, in each location, the service rate for the setting without integration (a), versus the service rate, transit rate, and door-to-door rate (i.e., the percentage of customers receiving door-to-door service) for the setting with integration. In Fig. 4.10(a), we observe that the platform mostly serves customers in business districts, while leaving many customers unsatisfied in remote locations. Nevertheless, in Fig. 4.10(b), almost all customers have access to ride-hailing services, regardless of their location. These improved results are achieved by offering first-mile and last-mile services to customers from residential areas (cf. Fig. 4.10(c)), and door-to-door services to customers in the business district that request trips over short distances (cf. Fig. 4.10(d)).

4.7 Conclusion

This article studies the online dispatching problem of a ride-hailing platform with integrated public transit transfers. Customer requests arrive dynamically, and indicate whether they are willing to receive first-mile and last-mile services. Customers are only eligible if several constraints are respected to ensure service quality. For each request, the platform decides (i) which service to offer between door-to-door, first-mile, and last-mile service; (ii) to which station a first-mile and last-mile customer should be directed; and (iii) which driver to dispatch to the resulting service.

To address this problem, we develop a lookahead approach based on stochastic optimization. The method is enhanced by a preprocessing procedure and an adaptation of the L-shaped method. It requires limited data and computational resources, making it applicable to instances of practical size, e.g., with thousands of requests per hour, and hundreds of drivers available. Our experiments show that our approach outperforms myopic methods by approximately 23 percentage points and expected-value lookahead methods by about 3 percentage points, while producing near-optimal decisions in a limited amount of time (i.e., less than 3 minutes on average).

In extensive numerical experiments, we examine the benefits of integrating public transit and ride-hailing services under various objectives and parameters. Using real-world data from New York City, we find that the integrated system can improve (i) the platform's profit by up to 7.3%, (ii) the percentage of fulfilled requests by up to 17.9%, and (iii) customers' travel time by up to 5.5%. Using synthetic data, we quantify the benefits of an integrated system

for the platform and its stakeholders, and the factors affecting the magnitude of benefits. The following insights are derived:

(i) Profit maximization and travel time minimization are compatible objectives that should both be optimized by the platform. In most cases, only a minor impact on platform profits would allow for significant reductions in customer travel times.

(ii) The platform's decisions present structural differences depending on the objective. When profit is the objective, customers with high transit access are more likely to receive first-mile and last-mile services. When travel time is the objective, customers with poor transit access are prioritized.

(iii) The platform may prefer last-mile services over first-mile services. First-mile services require immediate fulfillment, whereas last-mile services can be used for postponing requests to periods with more drivers available, which can increase the platform's profit.

(iv) An integrated system is particularly beneficial when supply is limited relative to demand (e.g., during peak hours). Although platform profit generally rises with demand, customer travel time may sometimes worsen, as the system becomes saturated and most requests are left unsatisfied.

(v) The magnitude of profit gains from integration depends strongly on the fare composition. In particular, with a high pickup fare per service, the platform can leverage public transit to satisfy more requests via shorter trips.

(vi) The integration of public transit is more effective when transit frequency is high than when transit speed is high. Because first-mile and last-mile customers must avoid spending too much time in public transit, waiting time at stations typically has a larger impact on total travel time than the speed of public transit.

(vii) Gains in profit are most significant when the public transit network is widely accessible across the service area, since the platform can exploit the transit system to improve dispatching. Gains in travel time are highest when the public transit network partially covers the area, because the platform then offers first-mile and last-mile service to remote customers.

This work opens several directions for further research. First, we assumed that customers indicate in advance whether they are willing to use public transit, and always accept the platform's suggested service if quality constraints are met. In practice, however, the customer's willingness depends on the service offered, and customers may cancel the service if it does not correspond to their expectations. A natural extension is therefore to model customer adoption more realistically, for example, using discrete choice models. Second, we supposed that customers receive either first-mile or last-mile services, and only

if their origin or destination is in the same location as a transit station. Future work could relax these assumptions, e.g., allowing customers to receive both first-mile and last-mile services, or to reach transit stations in other locations than their origin or destination. Third, our approach could also be applied to integrate public transit with other transportation means, such as bike-sharing, on-demand buses, and ride-pooling systems, to investigate the benefits of these alternative systems.

Appendix 4.A Proofs

4.A.1 Proof of Proposition 4.1

Proof. To prove that the optimal solution of the linear formulation is integral, it suffices to demonstrate that the linear formulation reduces to an instance of the min-cost flow problem. We first show it for profit maximization in three steps, and then adapt the proof for travel time minimization.

Profit maximization

Step 1. By the first condition of Proposition 4.1, we remove all variables z_{ji}^t from the linear formulation. The variables w_j^t are also removed, so that constraints (4.6c) are turned into inequality constraints.

Step 2. By the second condition in Proposition 4.1, we redefine the function $\gamma_k^{t,t'}$ to take value 1 if the customer service at PD pair k, starting at epoch t is exactly completed at epoch t', and 0 otherwise. Let the variable b_i^t be the number of drivers left idle in location i at epoch t. Then, we rewrite constraints (4.6e), (4.6f) as

$$b_i^1 + \sum_{k \in \mathcal{K}^1} d_{ik}^1 = N_i \qquad \forall\, i \in \mathcal{I}, \quad \text{and}$$

$$b_i^t + \sum_{k \in \mathcal{K}^t} d_{ik}^t = b_i^{t-1} + \sum_{u=1}^{t-1} \sum_{l \in \mathcal{I}} \sum_{k \in \mathcal{D}_i^u} \gamma_k^{u,t} \cdot d_{lk}^u \qquad \forall\, t \in \mathcal{T} \setminus \{1\},\, i \in \mathcal{I}.$$

Step 3. By the third condition, we define DR_{i,o_j}^t and DR_k^t to be revenue for picking up from driver origin i to customer origin o_j, and the revenue for serving a customer with PD pair k at epoch t. Further, let the variable c_k^t be the number of customers served for PD pair k at epoch t. With a slight abuse of notation, define d_{ij} as the number of drivers dispatched from origin i to customer OD pair j. The offline problem is reformulated as

$$\min \; Profit^{off} = -\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{O}_j^t} \sum_{j \in \mathcal{J}^t} DR_{i,\,o_j}^t \cdot d_{ij}^t - \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}^t} DR_k^t \cdot c_k^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} FC_{ji} \cdot y_{ji}^t \tag{4.10a}$$

$$\text{s.t.} \;\; N_i = b_i^1 + \sum_{j \in \mathcal{J}^1} d_{ij}^1 \quad \forall\, i \in \mathcal{I}, \tag{4.10b}$$

$$\sum_{i \in \mathcal{O}_j^t} d_{ij}^t = x_j^t + \sum_{u \in \mathcal{L}_j^t} y_{ji}^t \quad \forall t \in \mathcal{T},\, j \in \mathcal{J}^t, \tag{4.10c}$$

$$x_j^t + \sum_{j,\, i\,:\, k=(o_j,\, i)} y_{ji}^t = c_k^t \quad \forall t \in \mathcal{T},\, k \in \mathcal{K}^t, \tag{4.10d}$$

$$b_i^{t-1} + \sum_{u=1}^{t-1} \sum_{k \in \mathcal{D}_i^u} \gamma_k^{u,t} \cdot c_k^u = b_i^t + \sum_{j \in \mathcal{J}^t} d_{ij}^t \quad \forall t \in \mathcal{T} \setminus \{1\},\, i \in \mathcal{I}, \tag{4.10e}$$

$$\sum_{i \in \mathcal{I}} b_i^{n_T} + \sum_{u=1}^{n_T} \sum_{v=T}^{\infty} \sum_{k \in \mathcal{D}_i^u} \gamma_k^{u,v} \cdot c_k^u = \sum_{i \in \mathcal{I}} N_i \tag{4.10f}$$

$$x_j^t + \sum_{i \in \mathcal{L}_j^t} z_{ji}^t \leq D_j^t \quad \forall t \in \mathcal{T},\, j \in \mathcal{J}^t, \tag{4.10g}$$

$$x_j^t,\, y_{jl}^t,\, d_{ij}^t,\, b_i^t,\, c_k^t \geq 0 \quad \forall t \in \mathcal{T},\, j \in \mathcal{J}^t,\, l \in \mathcal{L}_j^t,\, i \in \mathcal{I},\, k \in \mathcal{K}^t \tag{4.10h}$$

Constraints (4.10b)-(4.10f) are flow constraints. For each of these constraints, the left-hand sides denote inflows, while the right-hand sides denote outflows. Constraints (4.10g) are (node) capacity constraints. As illustrated in Fig.4.11, the formulation is thus an instance of the min-cost flow problem.

Travel time minimization

The only difficulty is that the variable w_j^t, denoting unsatisfied customers, cannot be easily incorporated within a min-cost flow problem. Therefore, without loss of generality, we reformulate the travel time objective function (4.6b):

$$\min Time^{off} = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}^t} DT_{ik} \cdot d_{ik}^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{F}_j^t} (FT_{ji} - TT_j) \cdot y_{ji}^t$$
$$+ \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} \sum_{i \in \mathcal{L}_j^t} (LT_{ji} - TT_j) \cdot z_{ji}^t - \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} TT_j^t \cdot x_j^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}^t} TT_j^t \cdot D_j^t.$$

This objective function is similar to objective function (4.6b), but gets rid of variables w_j^t. It can be interpreted as the travel time reduction that is achieved by directing customers to first-mile, last-mile, and door-to-door service, assuming all customers initially incur a long travel time for not being served (cf. the last term). Since this objective function does not depend on variables w_j^t and the last term is constant, we can follow the same steps as for profit maximization, and conclude that the offline problem with this objective function is also an instance of the min-cost flow problem. □

4.A.2 Extension of Proposition 4.1 to Subproblem $\mathcal{Q}_s$

The following corollary extends Proposition 4.1, showing that the optimal solutions of each subproblem Q_s are integral under certain conditions.

Corollary 4.2. *If subproblem Q_s satisfies conditions (i)-(iii) of Proposition 4.1, and $q_{k,s}^h = 0$ for all $h = t+1, \ldots, t+n_H, k \in \widehat{K}_s^h$, then the optimal solution of Q_s is integral.*

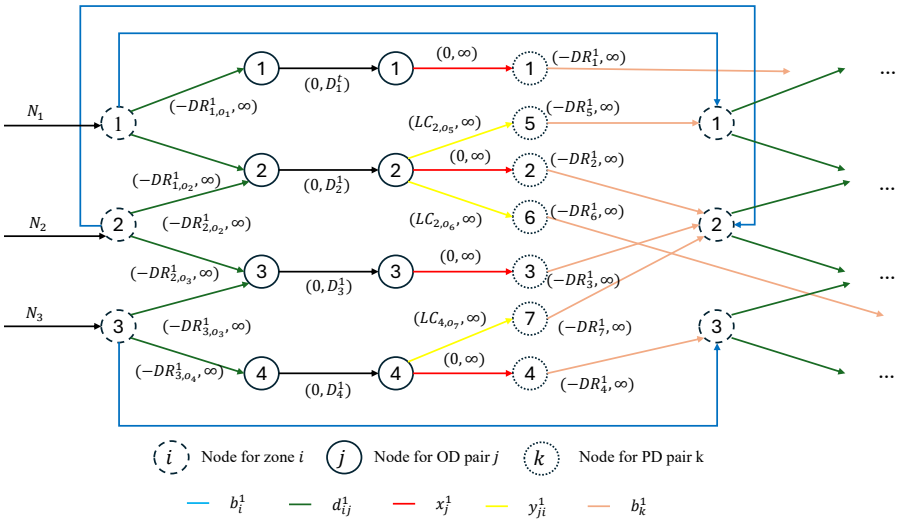


Figure 4.11: Illustrative min-cost flow network for the profit-oriented offline problem at the first epoch. The values in parentheses represent the costs and capacities associated with the arcs. When not specified, the cost and capacity of an arc are equal to $(0, \infty)$.

Sketch of the proof. The full proof, available upon request, proceeds in a similar fashion to the proof of Proposition 4.1, with two exceptions. In the second step, we include inflows in constraints (4.11d) to represent drivers completing service from previous epochs. Furthermore, a fourth step is added to include constraints (4.11g) to ensure that customer requests directed to last-mile services in previous epochs always receive door-to-door service. Since $q_{j,s}^h = 0, \forall h, j, s$, we know that there always exists a feasible solution that respects constraints (4.11g). After completing the four steps, the subproblem $\mathcal{Q}_s$ can be written as follows:

$$\mathcal{Q}_s(b_i^t, c_k^t, z_{ji}^t) \equiv \min \; - \sum_{h=t+1,\ldots,t+n_H} \Bigg[\sum_{i \in \mathcal{O}_j^h} \sum_{j \in \widehat{\mathcal{J}}_s^h} DR_{i,o_j}^h \cdot d_{ij,s}^h - \sum_{k \in \widehat{\mathcal{K}}_s^h} DR_k^h \cdot c_{k,s}^h + \sum_{j \in \widehat{\mathcal{J}}_s^h} \sum_{i \in \mathcal{F}_j^h} FC_{ji} \cdot y_{ji,s}^h \Bigg] \tag{4.11a}$$

$$\text{s.t.} \;\; N_i^{t,t+1} + \sum_{k \in \mathcal{D}_i^t} \gamma_k^{t,t+1} \cdot c_k^t + b_i^t = b_{i,s}^{t+1} + \sum_{j \in \widehat{\mathcal{J}}_s^{t+1} \cup \widetilde{\mathcal{J}}^{t+1}} d_{ij,s}^{t+1} \quad \forall i \in \mathcal{I}, \tag{4.11b}$$

$$\sum_{i \in \mathcal{O}_j^h} d_{ij,s}^h = x_{j,s}^h + \sum_{i \in \mathcal{F}_j^h} y_{ji,s}^h \quad \forall h = t+1, \ldots, t+n_H, \, j \in \widehat{\mathcal{J}}_s^h \cup \widetilde{\mathcal{J}}^h, \tag{4.11c}$$

$$x_{j,s}^h + \sum_{j,\, i \,:\, k=(o_j,\, i)} y_{ji,s}^h = c_{k,s}^h \quad \forall h = t+1, \ldots, t+n_H, \, k \in \widehat{\mathcal{K}}_s^h, \tag{4.11d}$$

$$N_i^{t,h} + \sum_{k \in \mathcal{D}_i^t} \gamma_k^{t,h} \cdot c_k^t + b_{i,s}^{h-1} + \sum_{u=t+1}^{h-1} \sum_{k \in \mathcal{D}_i^u} \gamma_k^{u,h} \cdot c_{k,s}^u = b_{i,s}^h + \sum_{j \in \widehat{\mathcal{J}}_s^h \cup \widehat{\mathcal{J}}^h} d_{ij,s}^h$$

$$\sum_{i \in \mathcal{I}} b_i^{n_H} + \sum_{u=1}^{n_H} \sum_{v=n_H}^{\infty} \sum_{k \in \mathcal{D}_i^u} \gamma_k^{u,v} \cdot c_k^u = \sum_{h=t+1}^{n_H} \sum_{i \in \mathcal{I}} N_i^{t,h} + \sum_{h=t+1}^{n_H} \sum_{k \in \mathcal{D}_i^t} \gamma_k^{t,h} \cdot c_k^t$$

$$\forall\, h = t+2, \ldots, t+n_H,\ i \in \mathcal{I}, \tag{4.11e}$$

$$x_{j,s}^h + \sum_{i \in \mathcal{L}_j^t} y_{ji,s}^h \le D_{j,s}^h \quad \forall\, h = t+1, \ldots, t+n_H,\ j \in \widehat{\mathcal{J}}_s^h, \tag{4.11f}$$

$$L_j^{t,h} + \sum_{j,\, i\,:\, k=(i,\, d_j)} \beta_{ji}^{t,\,h} \cdot y_{ji}^h = x_{j,s}^h \quad \forall\, h = t+1, \ldots, t+n_H,\ j \in \widetilde{\mathcal{K}}^h, \tag{4.11g}$$

$$x_{j,s}^h,\ y_{jl,s}^h,\ d_{ij,s}^h,\ b_{i,s}^h,\ c_{k,s}^h \ge 0 \quad \forall\, h = t+1, \ldots, t+n_H,\ j \in \widehat{\mathcal{J}}_s^h,\ l \in \mathcal{F}_j^h,\ i \in \mathcal{I},\ k \in \widehat{\mathcal{K}}_s^h \tag{4.11h}$$

Since constraints (4.11b)-(4.11e), (4.11g) are flow constraints, and constraints (4.10g) are (node) capacity constraints, the subproblem $\mathcal{Q}_s$ is an instance of the min-cost flow problem, and its optimal solution is integral.

Travel time minimization

As in the proof of Proposition 4.1, we can reformulate the time-oriented objective function of $\mathcal{Q}_s$ as follows:

$$\min \sum_{h=t+1,\ldots,\, t+n_H} \left[\sum_{k \in \widehat{\mathcal{K}}_s^h} \sum_{i \in \mathcal{O}_k^h} DT_{ik} \cdot d_{ik,s}^h + \sum_{j \in \widehat{\mathcal{J}}_s^h} \sum_{i \in \mathcal{F}_j^h} (FT_{ji} - TT_j) \cdot y_{ji,s}^h \right.$$

$$\left. + \sum_{j \in \widehat{\mathcal{J}}_s^h} \sum_{i \in \mathcal{L}_j^h} (LT_{ji} - TT_j) \cdot z_{ji,s}^h - \sum_{j \in \widehat{\mathcal{J}}_s^h} TT_j^t \cdot x_{j,s}^h + \sum_{j \in \widehat{\mathcal{J}}_s^h} TT_j^t \cdot D_{j,s}^h \right].$$

Then, repeating the same four steps as for the profit-oriented subproblem, we show that the time-oriented subproblem is an instance of the min-cost flow problem, and its optimal solution is integral.

5 Concluding Remarks

In this chapter, we provide some perspectives on the work presented in this thesis. We first outline some limitations of the research. Then, we discuss opportunities for future work. Since each preceding chapter already concludes with a detailed list of the main managerial insights, we omit such discussion here and instead focus on broader perspectives.

5.1 Research Limitations

In this thesis, stochastic models were developed to represent the operations of on-demand service platforms. As is always the case with mathematical models, a trade-off needed to be found between accurate modeling to describe complex phenomena and tractable modeling to derive practical solutions. To balance these two goals, we made some assumptions, which are next reviewed.

Probabilistic assumptions. A first category of assumptions relates to the approximation of uncertainty through simplified probabilistic distributions. In this thesis, the exponential distribution was extensively used to describe service times, customer inter-arrival times, and customer abandonment times (cf. Chapter 3 and 2). In some cases, some sources of uncertainty were completely ignored. For example, in Chapter 4, the travel time from one location to another was assumed constant. In Chapter 3 and 4, customers who were not receiving immediate service were supposed to abandon directly. Empirical observations sometimes justified these assumptions (e.g., the exponentially distributed inter-arrival times). Yet, for the most part, these assumptions were made for tractability purposes. Some attempts were carried out to evaluate their effects on the validity of decisions (cf. the extensions in Chapter 2), but further evaluation is required to test the robustness of the models when probabilistic assumptions do not hold.

Behavioral assumptions. A second category of assumptions (not exclusive to the first) concerns the modeling of strategic behaviors with respect to the platform's users. On-demand service platforms involve independent users on both

the supply and demand sides, making it essential to represent their behavior accurately. In Chapter 3, we tried to do so by modeling the driver's self-relocation decision using a multinomial logit model, and the customer's willingness to pay using a concave cumulative distribution function. In Chapter 2, we studied a model extension where drivers could self-schedule their working time after completing service. However, these only partially describe more complex decision processes. Moreover, in many models, several behavioral aspects were entirely omitted. For instance, it was assumed throughout the thesis that drivers fully comply with the platform's decisions, and that service cancellation never occurs after dispatching. In Chapter 4, customers are assumed to accept being directed to public transit if a series of customer-oriented conditions are satisfied. Clearly, these behavioral assumptions represent limitations of the present work. Nonetheless, such simplifications are also common in the literature. Currently, adequate modeling of strategic behaviors still constitutes an important challenge that deserves further attention.

Operational assumptions. A last category of assumptions refers to the operational characteristics of the problems studied. These were necessary to avoid being overly specific, so as to model operational problems rather than entire operational systems. For this reason, customers and drivers were aggregated into discrete zones. Decisions were made at fixed time intervals or discrete events. In line with most of the existing literature, we often did not represent interactions with other stakeholders than the platform's users (Chapter 4 being a notable exception). Finally, since on-demand service platforms are integrated systems with many interacting decisions, we often treated some decisions as exogenous while others were optimized. For example, in Chapter 3, the dispatching decision was made following a given heuristic policy: drivers were only dispatched to customers in the same location on a first-come, first-served basis. In Chapter 2 and 4, prices were given before optimizing customer admission and driver reservation. In practice, these decisions interact, and the output of one model can serve as input to another. For instance, the pricing parameters in Chapter 4 could be derived from the pricing decisions in Chapter 3, and the customer arrivals considered in Chapter 2 may be the result of the dispatching decisions modeled in Chapter 4.

Besides these modeling assumptions, some limitations of the thesis stem from the solution approaches employed to solve the models. Given the size and complexity of the problems, heuristic and approximate methods were often used to derive relevant decisions. As a result, we only found near-optimal solutions. In Chapter 3, we do not even have guarantees that the decisions made are close to optimality. We only know that they improve performance compared to decisions made by deterministic rolling horizon strategies and classical reinforcement learning methods. In Chapter 4, we derived optimality bounds on the solutions by solving the offline problem, yet the gap could be significant for a few specific instances. Moreover, the derived solutions could have been compared more extensively against alternative methods. For example, in Chapter 2, it may have been relevant to compare the optimal policy against

heuristic policies, in order to measure the gains allowed by our exact solution method. In Chapter 2 and Chapter 3, the solution approaches could also have been applied to various datasets (both from real-world and synthetic instances) to test the reproducibility of the results. Lastly, it is also worth mentioning that only expected quantities were maximized in this thesis. However, other paradigms may be considered to capture risk-aversion and worst-case scenarios. The objectives were generally defined over the short term (e.g., profit), while long-term objectives may have been considered (e.g., customer adoption).

5.2 Research Opportunities

This section explores five opportunities for future research. The first two would result in modeling contributions. The two others would lead to methodological contributions. The last one would bring a practical contribution.

5.2.1 Ride-Hailing with Walking Customers

In continuation of Chapter 4, it would be interesting to consider the case where customers are not directed to transit stations but can be suggested to walk to locations specified by the platform. Besides potential health benefits, it could improve dispatching, especially when some roads are difficult to access (e.g., congested streets during peak hours). From a more psychological perspective, walking could also be seen as less annoying than waiting still (Larson, 1987).

Nevertheless, the walking problem could be challenging to address. While only a fraction of customers can be directed to transit, most customers are eligible to walk to new pickup locations. As a result, the number of routes would be increased substantially, complicating the problem resolution. To deal with this issue, heuristic procedures could again be developed to select routes likely to be optimal. These procedures could, for instance, use the fact that customers should never walk toward congested locations, and would likely walk toward drivers currently available. Customers should also be advised to walk only if they are likely to incur long pickup times.

From a practical perspective, an important aspect is to model the customer acceptance decision rather than assuming customers are willing (or able) to walk. The speed of walking may also vary among customers, which can affect the platform's decisions. To model customer acceptance, the recent work by J. Yan et al. (2024) may be continued.

5.2.2 Ride-Hailing with Service Cancellation

While service cancellation is frequently observed in practice (e.g., about 10% of the requests are canceled by drivers, see Hazell, 2024), most literature neglects its impact on the operations. However, cancellation behavior is an important aspect of dispatching. Cancellation can be inconvenient for both sides:

customers have wasted time waiting; drivers have wasted time driving empty. Moreover, as explained in Chapter 1, dispatching involves a trade-off between customer abandonment (due to an excessive waiting time) and customer cancellation (due to an excessive pickup time). This trade-off is especially stringent in cities with multiple transit operators, where customers can easily switch to alternative transportation means if they wait too long. By ignoring cancellation, this trade-off cannot be captured. A few attempts have been made to model cancellation behaviors on the customer side (Liu et al., 2022; G. Wang et al., 2024), or the driver side (Meskar et al., 2023). However, to date, cancellation remains a significant modeling challenge, and its implications on driver dispatching are not fully understood. Many associated questions still need to be addressed. For instance, charging canceling users with a fee (as is done on several platforms, e.g., Uber) can improve dispatching efficiency. Yet, it also comes at the risk of damaging the platform's brand image.

5.2.3 Robust Optimization

In most of the literature, the platform's objective is defined so as to optimize some expected quantity (e.g., profit). This feature is generally motivated by the fact that platforms are rational, and that unexpected results do not significantly affect the optimized quantity. However, there are some cases in which the worst-case scenario can be highly detrimental to the platform, and its reputation. In such cases, risk-averse methods, like robust optimization, may be relevant. Next, we present two applications where robust optimization may be beneficial.

The first application lies in the integration of public transit and on-demand transport systems. The task of routing customers through multiple modes requires addressing a planning problem characterized by uncertain travel times and customer arrivals. These sources of uncertainty can sometimes render the solutions infeasible, leading to inconvenience for users. For instance, in Chapter 4, some customer services had to be canceled after taking public transit because no driver was available to complete the last-mile leg, contrary to the platform's expectation. To avoid such situations, platforms may be willing to adopt robust optimization methods to offer some guarantees to customers. When service quality is the objective, these methods can be applied to cope with uncertain travel times, which are particularly impactful in low-frequency public transit systems. When profit is the objective, these methods may effectively mitigate the effects of demand uncertainty, so as to avoid supply shortages.

The second application comes from food delivery platforms. In contrast to ride-hailing, these platforms tend to have more patient customers, often because these customers are less willing to turn to outside options. This feature can make food-delivery platforms more prone to excessive waiting times. Moreover, the delay can also result in meals that are not fresh anymore, potentially frustrating customers further (Ulmer et al., 2021). To hedge against worst-case scenarios, an adaptive robust optimization model could be developed, to ensure reasonable waiting times for all the served customers. Slightly more conserva-

tive decisions may significantly improve the platform's reputation in the long term, without compromising much of its revenues in the short term.

5.2.4 Hierarchical Markov Decision Processes

As shown throughout this thesis, on-demand service platforms encompass a variety of decisions, which are often optimized separately. While this simplifies the decision-making process, it can also lead to suboptimal decisions that fail to exploit potential synergies. One example is the interaction between pricing and driver dispatching. Pricing influences dispatching as it defines the revenues of possible driver-customer matches. Conversely, dispatching influences pricing because it affects the future driver distribution. By optimizing them jointly, the platform could improve profit and service quality.

Optimization of joint decisions is challenging due to differences in time scales: some decisions must be made within a few seconds (e.g., dispatching), while others must be made within a few minutes (e.g., pricing). A promising approach to address this challenge (and extend Chapter 3) is the use of hierarchical Markov decision processes (cf. Sutton et al., 1999). This framework, which has proven useful in operational contexts (e.g., inventory management, Goedhart et al., 2023), decomposes decisions into two levels. At the higher level, some decisions would be made periodically. At the lower level, other decisions would be made more frequently over sub-periods. To optimize such a hierarchical model, a mix of policy-based and value-based methods could be applied. By analyzing the resulting policy, we could assess the improvements achieved by integrated decisions, compared to independent decisions.

5.2.5 Open-Source Data

Currently, limited data is available to study the operations of on-demand service platforms. In the context of ride-hailing services, the most popular dataset, used in Chapter 3, is made publicly available by the NYC Taxi and Limousine Commission (2023). However, this dataset only provides trip records, and omits essential aspects, including censored demand, driver relocation between services, and service cancellation. Moreover, no driver IDs are associated with the recorded trips. As a result, much of the existing research has relied on synthetic data. When research used real-world data, it was often done on a limited number of datasets and protected by confidentiality agreements.

This lack of data and simulators seriously compromises the practical relevance and reproducibility of the models for on-demand service platforms (Qin et al., 2022). To address the issue, it would be valuable to develop realistic datasets and simulation environments that the whole community could use. These datasets and simulators would lead to more accurate representations of on-demand service platforms. They would also allow for reproducibility and systematic method benchmarking.

References

Afeche, P. (2013). Incentive-compatible revenue management in queueing systems: Optimal strategic delay. *Manufacturing & Service Operations Management*, *15*(3), 423–443.

Afèche, P., Liu, Z., & Maglaras, C. (2023). Ride-hailing networks with strategic drivers: The impact of platform control capabilities on performance. *Manufacturing & Service Operations Management*, *25*(5), 1890–1908.

Afeche, P., & Pavlin, J. M. (2016). Optimal price/lead-time menus for queues with customer choice: Segmentation, pooling, and strategic delay. *Management Science*, *62*(8), 2412–2436.

Agarwal, S., Mani, D., & Telang, R. (2023). The impact of ride-hailing services on congestion: Evidence from indian cities. *Manufacturing & Service Operations Management*, *25*(3), 862–883.

Agussurja, L., Cheng, S.-F., & Lau, H. C. (2019). A state aggregation approach for stochastic multiperiod last-mile ride-sharing problems. *Transportation Science*, *53*(1), 148–166.

Ahuja, K., Chandra, V., Lord, V., & Peens, C. (2021). Ordering in: The rapid evolution of food delivery. *McKinsey & Company*, *22*, 1–13.

Al-Kanj, L., Nascimento, J., & Powell, W. B. (2020). Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. *European Journal of Operational Research*, *284*(3), 1088–1106.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467.

Altman, E., Jiménez, T., & Koole, G. (2001). On optimal call admission control in resource-sharing system. *IEEE Transactions on Communications*, *49*(9), 1659–1668.

Aouad, A., & Saritaç, Ö. (2022). Dynamic stochastic matching under limited time. *Operations Research*, *70*(4), 2349–2383.

Atar, R., Giat, C., & Shimkin, N. (2010). The cμ/θ rule for many-server queues with abandonment. *Operations Research*, *58*(5), 1427–1439.

Aten, J. (2023). *This secret trick is the single most underrated feature in the Uber app.* Retrieved from `https://www.inc.com/jason-aten/uber-reserve-allows-you-to-set-forget-your-next-ride.html#google_vignette` (2025/08/18)

Azagirre, X., Balwally, A., Candeli, G., Chamandy, N., Han, B., King, A., ... others (2024). A better match for drivers and riders: Reinforcement learning at lyft. *INFORMS Journal on Applied Analytics*, *54*(1), 71–83.

Azriel, D., Feigin, P. D., & Mandelbaum, A. (2019). Erlang-s: A data-based model of servers in queueing networks. *Management Science*, *65*(10), 4607–4635.

Babar, Y., & Burtch, G. (2020). Examining the heterogeneous impact of ride-hailing services on public transit use. *Information Systems Research*, *31*(3), 820–834.

Bai, J., So, K. C., Tang, C. S., Chen, X., & Wang, H. (2019). Coordinating supply and demand on an on-demand service platform with impatient customers. *Manufacturing & Service Operations Management*, *21*(3), 556–570.

Banerjee, S., Freund, D., & Lykouris, T. (2022a). Pricing and optimization in shared vehicle systems: An approximation framework. *Operations Research*, *70*(3), 1783–1805.

Banerjee, S., Freund, D., & Lykouris, T. (2022b). Pricing and optimization in shared vehicle systems: An approximation framework. *Operations Research*, *70*(3), 1783–1805.

Banerjee, S., Johari, R., & Riquelme, C. (2016). Dynamic pricing in ridesharing platforms. *ACM SIGecom Exchanges*, *15*(1), 65–70.

Banerjee, S., Riquelme, C., & Johari, R. (2015). Pricing in ride-share platforms: A queueing-theoretic approach. *Working Paper, University of Columbia*.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*(5), 834–846.

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 679–684.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer. Math*, *4*(1), 238–252.

Benjaafar, S., El Hafsi, M., & Huang, T. (2010). Optimal control of a production-inventory system with both backorders and lost sales. *Naval Research Logistics (NRL)*, *57*(3), 252–265.

Benjaafar, S., & Hu, M. (2020). Operations management in the age of the sharing economy: What is old and what is new? *Manufacturing & Service Operations Management*, *22*(1), 93–101.

Beojone, C. V., & Geroliminis, N. (2023). Relocation incentives for ride-sourcing drivers with path-oriented revenue forecasting based on a markov chain model. *Transportation Research Part C: Emerging Technologies*, *157*, 104375.

Beojone, C. V., Zhu, P., Sirmatel, I. I., & Geroliminis, N. (2024). A hierarchical control framework for vehicle repositioning in ride-hailing systems. *Transportation Research Part C: Emerging Technologies*, 104717.

Bertsekas, D. (2019). *Reinforcement learning and optimal control* (Vol. 1). Athena Scientific.

Bertsimas, D., Jaillet, P., & Martin, S. (2019). Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, *67*(1), 143–162.

Biel, M., & Johansson, M. (2022). Efficient stochastic programming in julia. *INFORMS Journal on Computing*, *34*(4), 1885–1902.

Bimpikis, K., Candogan, O., & Saban, D. (2019). Spatial pricing in ride-sharing networks. *Operations Research*, *67*(3), 744–769.

Birge, J. R. (1997). State-of-the-art-survey—stochastic programming: Computation and applications. *INFORMS journal on computing*, *9*(2), 111–133.

Birge, J. R., & Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer Science & Business Media.

Birge, J. R., & Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, *34*(3), 384–392.

Boute, R. N., Gijsbrechts, J., Van Jaarsveld, W., & Vanvuchelen, N. (2022). Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research*, *298*(2), 401–412.

Braverman, A., Dai, J. G., Liu, X., & Ying, L. (2019). Empty-car routing in ridesharing systems. *Operations Research*, *67*(5), 1437–1452.

Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., & Zhao, L. (2005). Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, *100*(469), 36–50.

Cachon, G. P., Daniels, K. M., & Lobel, R. (2017). The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management*, *19*(3), 368–384.

Castillo, J. C., Knoepfle, D., & Weyl, G. (2017). Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation* (pp. 241–242).

Chen, C., Yao, F., Mo, D., Zhu, J., & Chen, X. M. (2021). Spatial-temporal pricing for ride-sourcing platform with reinforcement learning. *Transportation Research Part C: Emerging Technologies*, *130*, 103272.

Chen, Q., Lei, Y., & Jasin, S. (2023). Real-time spatial–intertemporal pricing and relocation in a ride-hailing network: Near-optimal policies and the value of dynamic pricing. *Operations Research*, *72*(5), 2097–2118.

Chen, S., Wang, H., & Meng, Q. (2020). Solving the first-mile ridesharing problem using autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, *35*(1), 45–60.

Chen, Y., & Wang, H. (2018). Pricing for a last-mile transportation system. *Transportation Research Part B: Methodological*, *107*, 57–69.

Civitatis. (2024). *Berlin metro (U-Bahn).* Retrieved from `https://www.introducingberlin.com/u-bahn#:~:text=Schedule%20and%20Frequency,at%20night%20every%2015%20minutes.` (2024/10/18)

Cox, D., & Smith, W. L. (1961). Queues, methuen & co. *Ltd., London*.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management science*, *1*(3-4), 197–206.

De Moor, B. J., Gijsbrechts, J., & Boute, R. N. (2022). Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. *European Journal of Operational Research*, *301*(2), 535–545.

De Munck, T., Chevalier, P., & Tancrez, J.-S. (2023). Managing priorities on on-demand service platforms with waiting time differentiation. *International Journal of Production Economics*, *266*, 109053.

De Munck, T., Tancrez, J.-S., & Chevalier, P. (2024). Pricing, customer admission, and driver repositioning in ride-hailing networks through transfer reinforcement learning. *Working Paper, Université catholique de Louvain*.

Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., & Xu, P. (2021). Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, *9*(3), 1–17.

Erhardt, G. D., Hoque, J. M., Goyal, V., Berrebi, S., Brakewood, C., & Watkins, K. E. (2022). Why has public transit ridership declined in the United States? *Transportation research part A: policy and practice*, *161*, 68–87.

Fedex. (2021). *Fedex overnight shipping.* Retrieved from `https://www.fedex.com/en-us/shipping/overnight.html` (2023/08/23)

Feng, S., Duan, P., Ke, J., & Yang, H. (2022). Coordinating ride-sourcing and public transport services with a reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *138*, 103611.

Fielbaum, A., Jara-Diaz, S., & Alonso-Mora, J. (2024). Beyond the last mile: different spatial strategies to integrate on-demand services into public transport in a simplified city. *Public Transport*, *16*(3), 855–892.

Friedrich, M., & Noekel, K. (2017). Modeling intermodal networks with public transport and vehicle sharing systems. *EURO Journal on Transportation and Logistics*, *6*(3), 271–288.

Garnett, O., Mandelbaum, A., & Reiman, M. (2002). Designing a call center with impatient customers. *Manufacturing & Service Operations Management*, *4*(3), 208–227.

Goedhart, J., Haijema, R., & Akkerman, R. (2023). Modelling the influence of returns for an omni-channel retailer. *European Journal of Operational Research*, *306*(3), 1248–1263.

Gómez-Lobo, A., Tirachini, A., & Gutierrez, I. (2022). Optimal prices for ridesourcing in the presence of taxi, public transport and car competition. *Transportation Research Part C: Emerging Technologies*, *137*, 103591.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1) (No. 2). MIT press Cambridge.

Gurvich, I., Lariviere, M., & Moreno, A. (2019). Operations in the on-demand economy: Staffing services with self-scheduling capacity. *Sharing Economy: Making Supply Meet Demand*, 249–278.

Ha, A. Y. (2000). Stock rationing in an $M/E_k/1$ make-to-stock queue. *Management Science*, *46*(1), 77–87.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning* (pp. 1861–1870).

Haliem, M., Mani, G., Aggarwal, V., & Bhargava, B. (2021). A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7931–7942.

Hall, J. D., Palsson, C., & Price, J. (2018). Is Uber a substitute or complement for public transit? *Journal of Urban Economics*, *108*, 36–50.

Hassin, R. (2016). *Rational queueing*. CRC press.

Haws, K. L., & Bearden, W. O. (2006). Dynamic pricing and consumer fairness perceptions. *Journal of Consumer Research*, *33*(3), 304–311.

Hazell, N. (2024). *What is the average cancellation rate for Uber drivers?* Retrieved from `https://www.ncesc.com/what-is-the-average-cancellation-rate-for-uber-drivers/` (2025/08/18)

Heitsch, H., & Römisch, W. (2009). Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, *118*, 371–406.

Hosseini, M., Milner, J., & Romero, G. (2024). Dynamic relocations in car-sharing networks. *Operations Research, forthcoming*.

Høyland, K., Kaut, M., & Wallace, S. W. (2003). A heuristic for moment-matching scenario generation. *Computational optimization and applications*, *24*, 169–185.

Hu, M., & Zhou, Y. (2022). Dynamic type matching. *Manufacturing & Service Operations Management*, *24*(1), 125–142.

Hu, Y., Dong, T., & Li, S. (2025). Coordinating ride-pooling with public transit using reward-guided conservative q-learning: An offline training and online fine-tuning reinforcement learning framework. *Transportation Research Part C: Emerging Technologies*, *174*, 105051.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, *50*(2), 1–35.

Iravani, S. M., Liu, T., & Simchi-Levi, D. (2012). Optimal production and admission policies in make-to-stock/make-to-order manufacturing systems. *Production and Operations Management*, *21*(2), 224–235.

Jacob, J., & Roet-Green, R. (2021). Ride solo or pool: Designing price-service menus for a ride-sharing platform. *European Journal of Operational Research*, *295*(3), 1008–1024.

Jayaswal, S., Jewkes, E., & Ray, S. (2011). Product differentiation and operations strategy in a capacitated environment. *European Journal of Operational Research*, *210*(3), 716–728.

Jiao, Y., Tang, X., Qin, Z. T., Li, S., Zhang, F., Zhu, H., & Ye, J. (2021). Real-world ride-hailing vehicle repositioning using deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, *130*, 103289.

Johnson, M. (2010). *Average schedule speed: How does metro compare?* Retrieved from https://web.archive.org/web/20151208111021/http://greatergreaterwashington.org/post/5183/average-schedule-speed-how-does-metro-compare/ (2025/07/30)

Kanoria, Y., & Qian, P. (2024). Blind dynamic resource allocation in closed networks via mirror backpressure. *Management Science*, *70*(8), 5445–5462.

Ke, J., Zhu, Z., Yang, H., & He, Q. (2021). Equilibrium analyses and operational designs of a coupled market with substitutive and complementary ride-sourcing services to public transits. *Transportation Research Part E: Logistics and Transportation Review*, *148*, 102236.

Kleywegt, A. J., Shapiro, A., & Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on optimization*, *12*(2), 479–502.

Koole, G. (1998). Structural results for the control of queueing systems using event-based dynamic programming. *Queueing Systems*, *30*(3), 323–339.

Koole, G. (2007). *Monotonicity in Markov reward and decision chains: Theory and applications* (Vol. 1). Now Publishers Inc.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Kullman, N. D., Cousineau, M., Goodson, J. C., & Mendoza, J. E. (2022). Dynamic ride-hailing with electric vehicles. *Transportation Science*, *56*(3), 775–794.

Kumar, P., & Khani, A. (2021). An algorithm for integrating peer-to-peer ridesharing and schedule-based transit system for first mile/last mile access. *Transportation Research Part C: Emerging Technologies*, *122*, 102891.

Larson, R. C. (1987). OR forum—perspectives on queues: Social justice and the psychology of queueing. *Operations research*, *35*(6), 895–905.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, J. (2023). *Here's why public transit keeps running out of money.* Retrieved from https://www.cnbc.com/2023/07/25/heres-why-public-transit-keeps-running-out-of-money.html (2025/03/31)

Lei, Z., & Ukkusuri, S. V. (2023). Scalable reinforcement learning approaches for dynamic pricing in ride-hailing systems. *Transportation Research Part B: Methodological*, *178*, 102848.

Lekach, S. (2020). *Pay a little more and Uber Eats will deliver faster.* Retrieved from `https://mashable.com/article/uber-eats-priority-delivery` (2023/08/23)

Li, S., Yang, H., Poolla, K., & Varaiya, P. (2021). Spatial pricing in ride-sourcing markets under a congestion charge. *Transportation Research Part B: Methodological*, *152*, 18–45.

Li, X., & Quadrifoglio, L. (2010). Feeder transit services: Choosing between fixed and demand responsive policy. *Transportation Research Part C: Emerging Technologies*, *18*(5), 770–780.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., & Tassa, Y. (2015). Continuous control with deep reinforcement learning. *Working Paper, Google Deepmind, London, United Kingdom*.

Lippman, S. A. (1975). Applying a new device in the optimization of exponential queuing systems. *Operations Research*, *23*(4), 687–710.

Liu, Y., Wu, F., Lyu, C., Li, S., Ye, J., & Qu, X. (2022). Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. *Transportation Research Part E: Logistics and Transportation Review*, *161*, 102694.

Lowalekar, M., Varakantham, P., & Jaillet, P. (2018). Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, *261*, 71–112.

Lu, A., Frazier, P. I., & Kislev, O. (2018). Surge pricing moves Uber's driver. In *Proceedings of the 2018 ACM Conference on Economics and Computation* (pp. 3–3).

Luo, Q., Li, S., & Hampshire, R. C. (2021). Optimal design of intermodal mobility networks under uncertainty: Connecting micromobility with mobility-on-demand transit. *EURO Journal on Transportation and Logistics*, *10*, 100045.

Luo, X., Gu, W., & Fan, W. (2021). Joint design of shared-bike and transit services in corridors. *Transportation Research Part C: Emerging Technologies*, *132*, 103366.

Lyft. (2021a). *Bonus zones - The Lyft Driver Blog.* Retrieved from `https://www.lyft.com/hub/posts/bonus-zones/` (2023/02/13)

Lyft. (2021b). *Lyft ride modes overview.* Retrieved from `https://help.lyft.com/hc/e/articles/115012927427-Lyft-ride-modes-overview` (2021/12/13)

Lyft. (2024). *Receiving airport FIFO pickup requests.* Retrieved from `https://help.lyft.com/hc/en-us/all/articles/115012922787-Receiving-Airport-FIFO-pickup-requests` (2024/02/13)

Ma, T.-Y., Rasulkhani, S., Chow, J. Y., & Klein, S. (2019). A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. *Transportation Research Part E: Logistics and Transportation Review*, *128*, 417–442.

Ma, W., Hekimoğlu, M., & Dekker, R. (2023). Admission control for a capacitated supply system with real-time replenishment information. *International Journal of Production Economics*, *266*, 109000.

Madrid Tourism Office. (2024). *Getting around Madrid by metro.* Retrieved from `https://www.esmadrid.com/en/madrid-metro.` (2024/10/18)

Mandelbaum, A., & Stolyar, A. L. (2004). Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized cμ-rule. *Operations Research*, *52*(6), 836–855.

Mao, C., Liu, Y., & Shen, Z.-J. M. (2020). Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *115*, 102626.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical programming*, *10*(1), 147–175.

Mendelson, H., & Whang, S. (1990). Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, *38*(5), 870–883.

Meskar, M., Aslani, S., & Modarres, M. (2023). Spatio-temporal pricing algorithm for ride-hailing platforms where drivers can decline ride requests. *Transportation Research Part C: Emerging Technologies*, *153*, 104200.

Montenegro, B. D. G., Sörensen, K., & Vansteenwegen, P. (2021). A large neighborhood search algorithm to optimize a demand-responsive feeder service. *Transportation Research Part C: Emerging Technologies*, *127*, 103102.

MTA. (2025). *Schedules.* Retrieved from `https://www.mta.info/schedules` (2025/07/30)

Naor, P. (1969). The regulation of queue size by levying tolls. *Econometrica: Journal of the Econometric Society*, 15–24.

Nourinejad, M., & Ramezani, M. (2020). Ride-sourcing modeling and pricing in non-equilibrium two-sided markets. *Transportation Research Part B: Methodological*, *132*, 340–357.

Numbeo. (2024). *Cost of living: Price rankings by country (transportation).* Retrieved from https://www.numbeo.com/cost-of-living/country_price_rankings_main (2024/10/18)

NYC Taxi and Limousine Commission. (2016). *TLC trip record data.* Retrieved from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page (2025/07/30)

NYC Taxi and Limousine Commission. (2023). *TLC trip record data.* Retrieved from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page (2023/09/03)

NYC Transit. (2025). *MTA subway stations.* Retrieved from https://data.ny.gov/Transportation/MTA-Subway-Stations/39hk-dx4f/about_data (2025/07/30)

OMNY. (2025). *Weekly fare cap.* Retrieved from https://omny.info/fares (2025/07/30)

Örmeci, L., Burnetas, A., & van der Wal, J. (2001). Admission policies for a two class loss system. *Stochastic Models*, *17*(4), 513–539.

Oroojlooyjadid, A., Nazari, M., Snyder, L. V., & Takáč, M. (2022). A deep Q-network for the Beer Game: Deep reinforcement learning for inventory optimization. *Manufacturing & Service Operations Management*, *24*(1), 285–304.

Özkan, E. (2020). Joint pricing and matching in ride-sharing systems. *European Journal of Operational Research*, *287*(3), 1149–1160.

Özkan, E., & Ward, A. R. (2020). Dynamic matching for real-time ride sharing. *Stochastic Systems*, *10*(1), 29–70.

Papadaki, K. P., & Powell, W. B. (2002). Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. *European Journal of Operational Research*, *142*(1), 108–127.

Pereira, M. V., & Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, *52*, 359–375.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (Vol. 703). John Wiley & Sons.

Powell, W. B. (2019). A unified framework for stochastic optimization. *European journal of operational research*, *275*(3), 795–821.

Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons.

Qian, X., Guo, S., & Aggarwal, V. (2022). Drop: Deep relocating option policy for optimal ride-hailing vehicle repositioning. *Transportation Research Part C: Emerging Technologies*, *145*, 103923.

Qin, Z. T., Zhu, H., & Ye, J. (2022). Reinforcement learning for ridesharing: An extended survey. *Transportation Research Part C: Emerging Technologies*, *144*, 103852.

Rochet, J.-C., & Tirole, J. (2006). Two-sided markets: a progress report. *The RAND journal of economics*, *37*(3), 645–667.

Rockafellar, R. T., & Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, *16*(1), 119–147.

Rogers, B. (2015). The social costs of Uber. *U. Chi. L. Rev. Dialogue*, *82*, 85.

Rosenblat, A. (2016). The truth about how Uber's app manages drivers. *Harvard Business Review*, *6*.

Ross, K. W., & Tsang, D. H. (1989). The stochastic knapsack problem. *IEEE Transactions on communications*, *37*(7), 740–747.

Savin, S. V., Cohen, M. A., Gans, N., & Katalan, Z. (2005). Capacity management in rental businesses with two customer bases. *Operations Research*, *53*(4), 617–631.

Scarf, H., Arrow, K., Karlin, S., & Suppes, P. (1960). The optimality of (S, s) policies in the dynamic inventory problem. *Optimal pricing, inflation, and the cost of price adjustment*, 49–56.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *Working Paper, University of California, Berkeley*.

Schwandl, R. (2024). *Urbanrail.net.* Retrieved from `https://www.urbanrail.net/` (2024/10/18)

Shapiro, A., Dentcheva, D., & Ruszczynski, A. (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shehadeh, K. S., Wang, H., & Zhang, P. (2021). Fleet sizing and allocation for on-demand last-mile transportation systems. *Transportation Research Part C: Emerging Technologies*, *132*, 103387.

Shi, K., Shao, R., De Vos, J., Cheng, L., & Witlox, F. (2021). The influence of ride-hailing on travel frequency and mode choice. *Transportation Research Part D: Transport and Environment*, *101*, 103125.

Shou, Z., & Di, X. (2020). Reward design for driver repositioning using multi-agent reinforcement learning. *Transportation research part C: emerging technologies*, *119*, 102738.

Shou, Z., Di, X., Ye, J., Zhu, H., Zhang, H., & Hampshire, R. (2020). Optimal passenger-seeking policies on e-hailing platforms using markov decision process and imitation learning. *Transportation Research Part C: Emerging Technologies*, *111*, 91–113.

Silver, D., Huang, A., Maddison, C. J., Guez, A., & Sifre, L. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Simonetto, A., Monteil, J., & Gambella, C. (2019). Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies*, *101*, 208–232.

Smith, W. E., et al. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, *3*(1-2), 59–66.

Stidham, S., & Weber, R. R. (1989). Monotonic and insensitive optimal policies for control of queues with undiscounted costs. *Operations Research*, *37*(4), 611–625.

Stiglic, M., Agatz, N., Savelsbergh, M., & Gradisar, M. (2018). Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers & Operations Research*, *90*, 12–21.

Subramanian, J., Stidham, S., & Lautenbacher, C. J. (1999). Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science*, *33*(2), 147–167.

Sun, B., Chen, S., & Meng, Q. (2025). Optimizing first-and-last-mile ridesharing services with a heterogeneous vehicle fleet and time-dependent travel times. *Transportation Research Part E: Logistics and Transportation Review*, *193*, 103847.

Sun, L., Teunter, R. H., Babai, M. Z., & Hua, G. (2019). Optimal pricing for ride-sourcing platforms. *European Journal of Operational Research*, *278*(3), 783–795.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.

Talluri, K. T., Van Ryzin, G., & Van Ryzin, G. (2004). *The Theory and Practice of Revenue Management* (Vol. 1). Springer.

Tang, G., Keshav, S., Golab, L., & Wu, K. (2018). Bikeshare pool sizing for bike-and-ride multimodal transit. *IEEE Transactions on Intelligent Transportation Systems*, *19*(7), 2279–2289.

Taylor, T. A. (2018). On-demand service platforms. *Manufacturing & Service Operations Management*, *20*(4), 704–720.

Thierer, A., Koopman, C., Hobson, A., & Kuiper, C. (2015). How the internet, the sharing economy, and reputational feedback mechanisms solve the lemons problem. *U. Miami l. Rev.*, *70*, 830.

TomTom. (2023). *Traffic index ranking.* Retrieved from `https://www.tomtom.com/traffic-index/ranking/.` (2024/10/18)

Tuncel, K., Koutsopoulos, H. N., & Ma, Z. (2023). An integrated ride-matching and vehicle-rebalancing model for shared mobility on-demand services. *Computers & Operations Research*, *159*, 106317.

Turan, B., Pedarsani, R., & Alizadeh, M. (2020). Dynamic pricing and fleet management for electric autonomous mobility on demand systems. *Transportation Research Part C: Emerging Technologies*, *121*, 102829.

Uber. (2019). *Get around your city for less.* Retrieved from `https://www.uber.com/blog/tampa-bay/uber-psta-dc/?uclick_id=fd481bae-6da1-4392-9ea5-fcb44d831f21` (2024/10/28)

Uber. (2021). *Priority pickups at airports: Benefit details.* Retrieved from `https://help.uber.com/riders/article/priority-pickups-at-airports-benefit-details?nodeId=53bd0a41-9df8-47ed-ba56-69688d78c92f.` (2021/12/13)

Uber. (2022). *What is surge pricing?* Retrieved from `https://help.uber.com/driving-and-delivering/article/what-is-surge?nodeId=e9375d5e-917b-4bc5-8142-23b89a440eec` (2024/02/13)

Uber. (2023). *Delivery heatmap: Finding busy areas.* Retrieved from `https://help.uber.com/en/driving-and-delivering/article/delivery-heatmap-finding-busy-areas?nodeId=88c4fc0e-278b-4d05-bd53-cc1ce7f0cc0b` (2024/02/13)

Uber. (2024a). *Canada's first ridesharing and transit partnership.* Retrieved from `https://www.uber.com/us/en/u/innisfil/?uclick_id=fd481bae-6da1-4392-9ea5-fcb44d831f21` (2024/10/28)

Uber. (2024b). *Uber announces results for first quarter 2024.* Retrieved from `https://investor.uber.com/news-events/news/press-release-details/2024/Uber-Announces-Results-for-First-Quarter-2024/default.aspx` (2025/05/11)

Uber. (2025). *How does uber match riders with drivers?* Retrieved from https://www.uber.com/be/en/marketplace/matching/ (2025/07/31)

Ulmer, M. W., Thomas, B. W., Campbell, A. M., & Woyak, N. (2021). The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, *55*(1), 75–100.

Van Mieghem, J. A. (1995). Dynamic scheduling with convex delay costs: The generalized c— mu rule. *The Annals of Applied Probability*, 809–833.

Van Slyke, R. M., & Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM journal on applied mathematics*, *17*(4), 638–663.

Vansteenwegen, P., Melis, L., Aktaş, D., Montenegro, B. D. G., Vieira, F. S., & Sörensen, K. (2022). A survey on demand-responsive public bus systems. *Transportation Research Part C: Emerging Technologies*, *137*, 103573.

Vercraene, S., Gayon, J.-P., & Karaesmen, F. (2018). Effects of system parameters on the optimal cost and policy in a class of multidimensional queueing control problems. *Operations Research*, *66*(1), 150–162.

Via. (2018). *ViaVan and BVG launch Berlkönig in Berlin.* Retrieved from https://ridewithvia.com/news/viavan-and-bvg-launch-berlkonig-in-berlin (2024/10/28)

Via. (2025). *What is Via Blue?* Retrieved from https://drivewithvia.com/push/37830-copy/#:~:text=What%20is%20Via%20Blue%3F,to%20drive%20with%20Blue%20mode. (2025/05/12)

Wang, G., Zhang, H., & Zhang, J. (2024). On-demand ride-matching in a spatial model with abandonment and cancellation. *Operations Research*, *72*(3), 1278–1297.

Wang, H. (2019). Routing and scheduling for a last-mile transportation system. *Transportation Science*, *53*(1), 131–147.

Wang, H., & Yang, H. (2019). Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, *129*, 122–155.

Wang, X., Chen, X. M., Xie, C., & Cheong, T. (2024). Coordinative dispatching of shared and public transportation under passenger flow outburst. *Transportation Research Part E: Logistics and Transportation Review*, *189*, 103655.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*, 279–292.

Williamson, O. E. (1981). The economics of organization: The transaction cost approach. *American journal of sociology*, *87*(3), 548–577.

Woodhouse, S. (2022). *Taxi fares are going up 23% in new york city.* Retrieved from `https://www.bloomberg.com/news/articles/2022-11-15/nyc-taxi-cab-fares-to-rise-23-in-first-increase-since-2012` (2024/10/18)

Wu, T., Zhang, M., Tian, X., Wang, S., & Hua, G. (2020). Spatial differentiation and network externality in pricing mechanism of online car hailing platform. *International Journal of Production Economics*, *219*, 275–283.

Yan, J., Martin, S., & Taylor, S. J. (2024). Trading flexibility for adoption: From dynamic to static walking in ride-sharing. *Management Science*.

Yan, P., Yu, K., Chao, X., & Chen, Z. (2023). An online reinforcement learning approach to charging and order-dispatching optimization for an e-hailing electric vehicle fleet. *European Journal of Operational Research*, *310*(3), 1218–1233.

Yan, X., Levine, J., & Zhao, X. (2019). Integrating ridesourcing services with public transit: An evaluation of traveler responses combining revealed and stated preference data. *Transportation Research Part C: Emerging Technologies*, *105*, 683–696.

Yu, X., Gao, S., Hu, X., & Park, H. (2019). A Markov decision process approach to vacant taxi routing with e-hailing. *Transportation Research Part B: Methodological*, *121*, 114–134.

Zhang, R., & Pavone, M. (2016). Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, *35*(1-3), 186–203.

Zhao, X., Stecke, K. E., & Prasad, A. (2012). Lead time and price quotation mode selection: Uniform or differentiated? *Production and Operations Management*, *21*(1), 177–193.

Zhong, Y., Pan, Q., Xie, W., Cheng, T., & Lin, X. (2020). Pricing and wage strategies for an on-demand service platform with heterogeneous congestion-sensitive customers. *International Journal of Production Economics*, *230*, 107-901.

Zhu, Z., Ke, J., & Wang, H. (2021). A mean-field Markov decision process model for spatial-temporal subsidies in ride-sourcing markets. *Transportation Research Part B: Methodological*, *150*, 540–565.

Zhu, Z., Xu, A., He, Q.-C., & Yang, H. (2021). Competition between the transportation network company and the government with subsidies to public transit riders. *Transportation Research Part E: Logistics and Transportation Review*, *152*, 102426.

Zohar, E., Mandelbaum, A., & Shimkin, N. (2002). Adaptive behavior of impatient customers in tele-queues: Theory and empirical support. *Management Science*, *48*(4), 566–583.