Dispatching and Pricing in Two-Sided Spatial Queues

Ang Xu[1] and Chiwei Yan[1]
[1]Department of Industrial Engineering and Operations Research, UC Berkeley
angxu@berkeley.edu, chiwei@berkeley.edu

Extended abstract submitted for presentation at the 12th Triennial Symposium on Transportation Analysis conference (TRISTAN XII)
June 22-27, 2025, Okinawa, Japan

February 27, 2025

Keywords: matching, pricing, ride-hailing platforms, spatial queues

1 Introduction

In recent years, ride-hailing services such as Uber, Lyft, and Didi have revolutionized urban transportation by providing a convenient and flexible alternative to traditional taxi services. The dispatching process, responsible for assigning available drivers to incoming ride requests, is crucial for the operational efficiency of ride-hailing services. In designing effective dispatching and pricing algorithms, platforms often aim to balance key objectives: keeping prices affordable, minimizing pickup distances, and reducing rider wait times to get a match (Yan *et al.*, 2020).

In this paper, we explore the joint dispatching and pricing problem in two-sided spatial queues where both sides (riders and drivers) can wait in the system. This extends prior literature on one-sided dispatching (e.g., Castillo *et al.* 2024, Xu *et al.* 2020 and Besbes *et al.* 2022) where either arriving riders are immediately matched to the closest available drivers or a newly available driver is immediately matched a waiting rider. The two-sided feature of our model is similar to that of Wang *et al.* (2024), who characterize optimal dispatching by allowing intentional idling of drivers, even when riders are waiting in the queue, using a fluid analysis. However, our approach directly optimizes a Markovian model, enabling a more detailed stochastic analysis and state-dependent pricing and dispatching. We summarize our contributions as follows.

- **Modeling.** We give a continuous-time Markovian model to depict the dynamics of the dispatching process of a ride-hailing platform. The platform adaptively sets prices and makes matching decisions. One distinguishing feature of our Markovian model is that the service time is state-dependent, whose rates depend on the current number of idle drivers and riders in the queue, capturing the spatial system we model. The platform aims to design dynamic policies that maximize the long-run average revenue rate while accounting for penalties associated with rider waiting time, both in the queue and during pickup.
- **Optimality Analysis.** We show that, under certain mild assumptions, the optimal dispatching policy has a closed-form solution. Inspired by this closed-form result, for general cases, we introduce a class of dispatching policy called zigzag policy, which enjoys a threshold-type structure and is tractable and easy to implement. We develop a dynamic programming algorithm that finds a high-quality zigzag policy efficiently.

2 Brief Overview of Model and Analysis

We give a brief overview of our model, analysis, and some key results. Consider a ride-hailing platform where riders are served over a fixed service region. The arrival of riders follows a

Poisson process with rate Λ whose origins are sampled from the service region. Upon arrival, each customer is offered with a price. If the customer decides to pay for the ride, she joins a queue waiting to be matched with a driver—the platform is *not* required to immediately dispatch a driver upon rider arrival. In the following, we introduce the state space, the action space, and the policy class for the platform.

State, action and policy. Let L be the total number of drivers in the system and $\mathcal{L} := \{0, 1, \cdots, L\}$ be the index set of drivers. The state space $\mathcal{S} := \{(l, m) : l \in \mathcal{L}, m \in \mathbb{Z}_{\geq 0}\}$ is defined as a set of ordered pairs (l, m), where l represents the number of drivers *in service* and m represents the number of riders in the queue. There are three events that can cause a state to change. First, when a driver completes his trip, he immediately becomes idle, and the state transitions from (l, m) to $(l - 1, m)$. Second, when a customer arrives and accepts the price to join the queue, the state transitions from (l, m) to $(l, m+1)$. Third, when a dispatching decision happens, the state transitions from (l, m) to $(l + 1, m - 1)$.

The action of the platform contains two parts: pricing and dispatching. At the moment of each state transition, the platform updates the price for incoming riders and makes its dispatching decisions. For pricing, each customer is offered with a price p upon arrival. We assume a linear relationship between the price p and the trip distance d, given by $p = p_0 + p_1 d$. Here, p_0 is a fixed base fare preset by the platform for each trip, while p_1 represents the per-mile charge which can be adjusted by the platform according to different states (aka surge multipliers). Riders' willingness to pay per mile for the ride is drawn from a cumulative distribution function (CDF) $F(\cdot)$. A customer would accept the price and join the queue if and only if their willingness to pay per mile is higher or equal to p_1. As a result, the *effective request rate* (or request conversion rate) can be written as $\lambda = \Lambda \bar{F}(p_1)$, where $\bar{F}(p_1) = 1 - F(p_1)$ is the tail CDF of customer's willingness to pay per mile. We assume λ has a one-to-one relationship to p_1. Hence, setting a price p_1 is equivalent to setting an effective rate λ. We use $p_1(\lambda)$ to represent the per-mile price under effective request rate λ. In the following, we express pricing-related decisions in terms of λ instead of p. We define $\boldsymbol{\lambda} := \{\lambda_{l,m} : l \in \mathcal{L}, m \in \mathcal{N}\}$ as a *pricing policy* where $\lambda_{l,m}$ is the effective request rate at state (l, m).

A *dispatching policy* $\boldsymbol{\phi} := \{\phi_{l,m} : \phi_{l,m} \in \{0, 1\}, l \in \mathcal{L}, m \in \mathbb{Z}_{\geq 0}\}$ determines whether the platform dispatches available drivers to riders at state (l, m). In specific, $\phi_{l,m} = 1$ represents the dispatching action, and $\phi_{l,m} = 0$ the holding action (no dispatching) under state (l, m). The system transitions from state (l, m) to state $(l + 1, m - 1)$ if $\phi_{l,m} = 1$. At the new state $(l + 1, m - 1)$, the platform considers again whether to dispatch and repeats this process until policy ϕ at the current state indicates a holding action.

Combining the pricing and dispatching policies introduced above, we now obtain a policy $(\boldsymbol{\lambda}, \boldsymbol{\phi})$: the pricing policy $\boldsymbol{\lambda}$ determines the rate of riders joining the queue, and the dispatching policy ϕ specifies whether to dispatch an idle driver at each state.

Trip completion rate. For each trip, the service time consists of two parts: "pick-up time" and "on-trip time". The pick-up time depends on the number of idle drivers as well as the number of riders in the queue. The more idle drivers or riders in the queue, the shorter the expected pick-up time will be as platform often matches the closest idle driver and rider pair. We assume that trip times/distances are drawn from some distribution. We approximate the service time of each driver in-service at state (l, m) as an exponential random variable with rate $\mu_{l,m}$ which increases with m and decreases with l. The trip completion rate at state (l, m) is thus $l\mu_{l,m}$.

Objective. Our objective balances two components: (1) maximizing the platform's expected long-run average revenue, and (2) minimizing penalties associated with rider waiting in queues and pickup delays. Let $\pi_{\boldsymbol{\lambda},\boldsymbol{\phi}}(l, m)$ be the stationary distribution of having l drivers in service and m riders in the queue under policy $(\boldsymbol{\lambda}, \boldsymbol{\phi})$. Without loss of generality, we can set the expected distance of each trip to be 1. Then the long-run average revenue can be written as $R(\boldsymbol{\lambda}, \boldsymbol{\phi}) := \sum_{(l,m)\in\mathcal{S}} \pi_{\boldsymbol{\lambda},\boldsymbol{\phi}}(l, m) \left(p_0 + p_1(\lambda_{l,m})\lambda_{l,m}\right)$. We consider two penalties: (1) the expected number of riders waiting to be picked up, and (2) the expected number of riders in the queue. It can be shown that, for penalty (1), replacing the expected number of riders waiting for pickup with the expected number of customers in service does not affect the optimal policy. Thus, our objective function can be written as

$$\tilde{R}(\boldsymbol{\lambda}, \boldsymbol{\phi}) := \sum_{(l,m)\in\mathcal{S}} \pi_{\boldsymbol{\lambda},\boldsymbol{\phi}}(m,l)\left(p_0 + \lambda_{l,m} p_1(\lambda_{l,m}) - c_1 l - c_2 m\right),$$

where c_1 is the cost per waiting rider for pickup and c_2 is the cost per waiting rider in the queue.

2.1 Optimality

Our first result shows that, under certain mild assumptions, the optimal dispatching policy has a closed form. We begin by introducing these assumptions.

Assumption 1 *We assume $\boldsymbol{\mu}$ satisfies the following conditions.*

1. *$\mu_{l,m}$ is bounded above, decreasing in l, and increasing in m;*
2. *$\mu_{l,m+1} - \mu_{l,m}$ is decreasing in m and increasing in l;*
3. *$\mu_{l,m} - \mu_{l+1,m}$ is decreasing in m and increasing in l.*

Assumption 1 states that the service rate is decreasing in the number of drivers in service and is increasing in the number of riders in the queue. Moreover, the increment in service rate with one more idle vehicle or one more customer in the queue is decreasing in l and m. These are often satisfied in spatial systems. To help characterize the optimal dispatching policy, we classify a state into either *type 1* or *type 2*, defined as follows.

Definition 1 *We say a state (l,m) a type 1 state if $l\mu_{l,m} > (l+1)\mu_{l+1,m-1}$ holds (we assume $\mu_{-1,m} = \mu_{l,-1} = 0$). In other words, (l,m) is a type 1 state if the total service rate at (l,m) is greater than one at $(l+1, m-1)$. Otherwise, we say (l,m) is a type 2 state.*

In short, a type 1 state has a higher total service rate compared to its lower left state (the state after dispatching exactly 1 driver) and conversely, a type 2 state has a lower total service rate compared to its lower left state. The intuition of classifying a state as such is to determine whether or not the total service rate will increase after a dispatching action. Using these definitions, under Assumption 1, we can characterize the optimal dispatching policy.

Theorem 1 (Optimal Dispatching Policy) *Suppose $\boldsymbol{\mu}$ satisfies Assumption 1,*

1. *If $c_1 = c_2$, there exists an optimal policy $(\boldsymbol{\lambda}^*, \boldsymbol{\phi}^*)$ such that $\phi^*_{l,m} = 0$ for all type 1 states and $\phi^*_{l,m} = 1$ for all type 2 states.*
2. *If $c_1 > c_2$, there exists an optimal policy $(\boldsymbol{\lambda}^*, \boldsymbol{\phi}^*)$ such that $\phi^*_{l,m} = 0$ for all type 1 states.*
3. *If $c_1 < c_2$, there exists an optimal policy $(\boldsymbol{\lambda}^*, \boldsymbol{\phi}^*)$ such that $\phi^*_{l,m} = 1$ for all type 2 states.*

Theorem 1 motivates a subclass of general dispatching policies, which we call *zigzag policy*. By restricting our attention to this subclass of policies, we greatly simplify the computation while we can still obtain a near-optimal solution. We begin with its definition.

Definition 2 (Zigzag Policy) *We say a dispatching policy is zigzag if:*

1. *For all l, $[\phi_{l,m} : m \in \mathbb{Z}_{\geq 0}]$ is a sequence of consecutive 0(s) followed by consecutive 1(s);*
2. *For all m, $[\phi_{l,m} : l \in \mathcal{L}]$ is a sequence of consecutive 1(s) followed by consecutive 0(s).*

The definition of zigzag policy is similar to the definition of a matrix in *row echelon form* if we consider m as the horizontal axis and l as the vertical axis. One advantage of the zigzag policy is that the induced Markov chain under is a birth-death process, with a birth rate $\lambda_{l,m}$ and a death rate $\mu_{l,m}$ at each recurrent state (l,m). Another advantage is that the policy

$l \backslash m$	0	1	2	3	4
0	0	1	1	1	1
1	0	0	1	1	1
2	0	0	0	0	1
3	0	0	0	0	0

Table 1 – *An example of a zigzag dispatching policy.*

has a simple structure to implement, as it specifies a dispatching threshold for the number of riders in the queue at each level of available drivers. The zigzag structure also motivates us to reframe the problem as a path-finding problem which separates the 0s and 1s. Motivated by this, we introduce a dynamic-programming-based algorithm that yields a zigzag policy by applying efficient dominance rules.

2.2 Numerical Experiments

We conduct numerical experiments to evaluate the performance of our algorithm. We set $(L, \Lambda) = (100, 40)$. Service rate $\mu_{l,m}$ is estimated from simulation data. We test various combinations of penalties (c_1, c_2), where $c_1, c_2 \in \{0.5, 0.75, 1\}$, resulting in a total of 9 different cases. The pricing function is defined with a base fare $p_0 = 5$, and the dynamic fare component is given by a linear inverse demand function $p_1(\lambda) = 2(1 - \lambda/\Lambda)$. Next, we evaluate the performance of our dynamic programming algorithm under both static pricing (with constant $\lambda_{l,m}$) and dynamic pricing. We benchmark our approach against a policy derived from a standard value iteration method, as well as a naive policy that dispatches drivers immediately if possible. We set a limit of 20 minutes in computation time. We observe that the dynamic programming algorithm significantly outperforms the benchmarks.

	Value Iteration		Naïve		Zigzag Dynamic		Zigzag Static	
c_1, c_2	$\tilde{\mathcal{R}}(\lambda, \phi)$	Time (s)	$\tilde{\mathcal{R}}(\lambda, \phi)$	Time (s)	$\tilde{\mathcal{R}}(\lambda, \phi)$	Time (s)	$\tilde{\mathcal{R}}_s(\lambda, \phi)$	Time (s)
0.5, 0.5	112.18	1,200.00	116.03	474.32	128.16	618.43	126.95	169.40
0.5, 0.75	101.53	1,200.00	111.28	955.00	125.80	563.34	124.38	168.92
0.5, 1.0	91.07	1,200.00	111.27	510.14	124.04	490.29	122.45	150.14
0.75, 0.5	87.94	1,200.00	91.03	1,200.00	105.43	643.61	104.69	169.97
0.75, 0.75	77.14	1,200.00	90.92	503.03	103.37	559.14	102.50	168.70
0.75, 1.0	66.55	1,200.00	90.92	257.14	101.82	491.76	100.86	152.63
1.0, 0.5	64.00	1,200.00	71.53	493.03	83.57	583.16	83.22	149.15
1.0, 0.75	52.91	1,200.00	71.53	237.00	81.82	519.07	81.41	150.65
1.0, 1.0	42.31	1,200.00	71.53	146.08	80.50	462.09	80.04	130.64

Table 2 – *Comparison of different algorithms under different penalty coefficients.*

References

Besbes, Omar, Castro, Francisco, & Lobel, Ilan. 2022. Spatial capacity planning. *Operations Research*, **70**(2), 1271–1291.

Castillo, Juan Camilo, Knoepfle, Dan, & Weyl, E Glen. 2024. Matching and pricing in ride hailing: Wild goose chases and how to solve them. *Management Science*.

Wang, Guangju, Zhang, Hailun, & Zhang, Jiheng. 2024. On-demand ride-matching in a spatial model with abandonment and cancellation. *Operations Research*, **72**(3), 1278–1297.

Xu, Zhengtian, Yin, Yafeng, & Ye, Jieping. 2020. On the supply curve of ride-hailing systems. *Transportation Research Part B: Methodological*, **132**, 29–43.

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