Submitted to *Omega*
manuscript OMEGA-D-24-01203

Fleet Size Planning in Crowdsourced Delivery: Balancing Service Level and Driver Utilization

Aliaa Alnaggar[a], Sahil Bhatt[a]
[a] Department of Mechanical, Industrial and Mechatronics Engineering, Toronto Metropolitan University, Toronto, ON, M5B 2K3, Canada

aliaa.alnaggar@torontomu.ca, sahil.bhatt@torontomu.ca

This paper addresses the fleet size planning problem for crowdsourced delivery platforms, focusing on optimizing the number of crowdsourced drivers to balance the platform's service level and driver utilization. This is motivated by recent regulatory measures that limit the number of vehicle licenses a platform can hold, which caps fleet size as a means to enhance driver working conditions and mitigate congestion and emissions caused by an excessive number of idle vehicles on the road. We propose a two-stage optimization model where the first stage involves tactical decisions for determining fleet sizes, while the second stage captures the operational dynamics of the platform by a Markov decision process (MDP) in which uncertain crowd driver arrival in the MDP depends on fleet size decisions in the first stage, introducing decision-dependent uncertainty. To efficiently solve the model, we employ a value function approximation (VFA) algorithm that iteratively determines fleet sizes using Boltzmann exploration then approximates the second-stage MDP using a rolling-horizon parametric cost function approximation. Extensive computational experiments confirm the effectiveness of the VFA algorithm, demonstrating its ability to meet both service level and driver utilization targets under various conditions. The results show that a platform can achieve high driver utilization and meet its service level target while only marginally reducing its profit, especially when driver behavior is more predictable and fleet sizes are temporally adjusted.

Key words: crowdsourced delivery, fleet size, Markov decision process, decision-dependent uncertainty, value function approximation

1. Introduction

Crowdsourced delivery has emerged as a novel last-mile delivery approach, where the *crowd*, i.e., freelance independent drivers, use their own vehicles to complete last-mile delivery tasks. This form of delivery falls under the emergent concept of the *sharing economy*, where individuals leverage underutilized assets to create value and generate supplementary income. By utilizing existing vehicles and engaging a broader pool of potential drivers, crowdsourced delivery has great potential for providing efficient and low-cost on-demand delivery that addresses the ever-increasing demand of the same-day delivery market. 45 Despite its significant potential, crowdsourced delivery and other sharing economy transportation services, such as ride-hailing, have recently faced increasing

scrutiny due to the limited protections they offer their workers. As drivers are compensated per-task or *gig*, they often experience extended periods of idle time while awaiting assignment to a delivery request (Benjaafar et al. 2022), which can result in earnings that fall below the minimum wage (Stanford 2022). Moreover, recent studies indicate that although these platforms aim to capitalize on underutilized vehicles, they often result in intensifying traffic congestion (Diao, Kong, and Zhao 2021).

To address these shortcomings, recent efforts have focused on regulating sharing economy transportation platforms to ensure fair compensation and prevent excessive fleet sizes that result in too many idle vehicles. These regulations include minimum wage and utilization standards, as seen in New York City (Parrott et al. 2019), Minnesota (Rothenberg 2024), and Ontario (Gray 2022), as well as capping the number of licenses granted to each platform (Fitzsimmons 2018), thus setting an upper limit on fleet size.

However, there are several challenges that make the establishment of these regulations not a straightforward task for policymakers, regulators, and platforms. First, the self-scheduling nature of drivers in crowdsourced delivery platforms compound the uncertainty of traditional delivery methods, since not only is demand uncertain, but also a platform's delivery capacity due to drivers varying availability is also uncertain. Second, these platforms often handle geographically dispersed pickup and delivery locations, which can lead to imbalanced service regions where drivers are not positioned near high-demand areas. Finally, demand on these platforms typically varies throughout the day, requiring dynamic adjustments to platform capacities to prevent both excess idle time and shortages of available drivers.

In this research, we investigate how a platform can leverage two key control levers, namely *fleet size planning* and *driver-order matching* to balance the trade-offs between maintaining a high service level (i.e., the proportion of received orders fulfilled by the delivery deadline) and ensuring sufficient driver utilization. We consider a setting where the platform provides delivery services across geographically dispersed pickup and drop-off locations with tight time constraints, such as those found in restaurant or grocery deliveries. The platform determines the fleet size at various periods over the planning horizon, which represents determining the set of drivers who may be available during specific times of the day, such as in the morning, afternoon, or the evening. However, as drivers are independent contractors, the platform lacks direct control over their actual availability on any given day. Once the fleet size is established, drivers independently decide whether or not to offer delivery services in a given day, introducing uncertainty into the system. At the operational level, the platform seeks to sequentially optimize the matching of drivers to orders to maximize profit, while maintaining high service levels and ensuring efficient driver utilization, considering uncertainty in both order and driver arrival.

Our contributions in this paper are as follows:

- We introduce a new problem faced by crowdsourced delivery platforms, namely determining the fleet size of crowdsourced drivers to balance service level and driver utilization at the operational level. While the literature considers fleet size management for contracted drivers in systems with a hybrid of contract and independent drivers, to the best of our knowledge, this work is the first to consider fleet size management of *crowdsourced drivers*, in an effort to improve their work conditions while ensuring high platform service reliability.
- We develop a novel two-stage optimization model that aims to determine the optimal driver fleet size that minimizes penalties incurred when expected service levels and driver utilization fall short of target thresholds. In the first stage, which represents tactical level planning, the platform determines time-varying driver fleet sizes. In the second-stage, a Markov Decision Process (MDP) model is proposed to capture the operational problem of the platform in which the platform dynamically matches drivers and order, where the uncertain driver arrival in the MDP is a function of the selected pool size in the first stage.
- To address the computational challenges associated with the proposed model, specifically, that fleet size decisions impose decision-dependent uncertainty in the second-stage large-scale MDP, we present a value function approximation (VFA) solution method that iteratively searches the solution space of the proposed model. The VFA method leverages Boltzmann exploration to determine fleet sizes and cost function approximation to efficiently approximate the underlying MDP for each candidate first-stage solution.
- We conduct an extensive computational study using synthetic and real-world data to evaluate the performance of the proposed solution approach and analyze the tradeoff between platform service level and driver fulfillment. The results demonstrate that platforms can balance service level and driver utilization, particularly with greater driver adherence to suggested work periods, which reduces the uncertainty in their behavior, and with temporally adjusted fleet sizes.

The remainder of this paper is structured as follows. Section 2 reviews relevant literature and positions our research contribution. Section 3 introduces the two-stage fleet size planning model and details the MDP model for assessing a candidate fleet size decision. Section 4 presents the value function approximation algorithm and details the Boltzmann exploration and parametric cost function approximation steps. Section 5 discusses the experimental setup and presents computational results and managerial insights. Finally, concluding remarks and future research directions are discussed in Section 6.

2. Literature Review

Our work investigates fleet size planning for crowdsourced delivery platforms, aiming to balance platform service levels with driver utilization. In what follows, we position our work within three

key streams of literature: *crowdsourced delivery*, *fleet size planning in crowdsourced delivery* and *Compensation and fair task allocation in crowdsourced delivery*.

2.1. Crowdsourced Delivery

In recent years, crowdsourced delivery has gained a lot of interest in the operations research literature as an alternative efficient method for last-mile delivery. For a comprehensive summary on the topic, we refer the readers to the reviews by Cleophas et al. (2019), Alnaggar, Gzara, and Bookbinder (2021), Savelsbergh and Ulmer (2022) and Kaspi, Raviv, and Ulmer (2022).

Most of the literature on crowdsourced delivery focuses on operational decisions faced by platforms, particularly under uncertainty regarding the arrival or availability of crowdsourced drivers. A significant body of this research examines optimizing *matching and routing* decisions, concurrently. This was pioneered by Archetti, Savelsbergh, and Speranza (2016), who introduced a variant of the capacitated vehicle routing problem (VRP) that considers a predetermined set of *occasional* or crowdsourced drivers. Other studies have explored various extensions of VRP assuming crowdsourced orders originate from a common depot (e.g., Dayarian and Savelsbergh (2020), Torres, Gendreau, and Rei (2022b), Mancini and Gansterer (2022), Pugliese et al. (2023), Silva, Pedroso, and Viana (2023), Wang, Xu, and Qin (2023)) as well as variants of the pickup and delivery problem (DPDP) (e.g., Arslan et al. (2019), Su et al. (2023), Mousavi et al. (2024)) which assumes geographically dispersed order pickup locations. Research has also investigated the interplay between routing and other decisions, such as joint routing and compensation (Le et al. 2021, Cerulli et al. 2024).

Other studies have examined *matching* decisions in crowdsourced delivery systems, including matching orders to distribution center employees acting as crowdsourced drivers (Boysen, Emde, and Schwerdfeger 2022), order bundling and matching (Simoni and Winkenbach 2023), joint matching and stochastic repositioning under short time windows (Alnaggar, Gzara, and Bookbinder 2024a), joint matching and pricing (Behrendt, Savelsbergh, and Wang 2024), matching considering drivers' accepting/rejecting an order match (Yildiz and Savelsbergh 2019, Ausseil, Ulmer, and Pazour 2024) and matching considering predetermined compensation guarantees (Alnaggar, Gzara, and Bookbinder 2024b). Further operational decisions studied in the literature include designing request menus from which drivers can select orders they wish to fulfill (Horner, Pazour, and Mitchell 2021, Ausseil, Pazour, and Ulmer 2022) as well as determining personalized driver incentives (Horner, Pazour, and Mitchell 2024).

Non-operational decision problems in crowdsourced delivery have received less attention in the literature. Examples include scheduling a subset of full-time drivers to manage uncertainty of the arrival of crowdsourced drivers (Ulmer and Savelsbergh 2020, Behrendt, Savelsbergh, and Wang

2023), selecting mobile depots to coordinate delivery (Mousavi, Bodur, and Roorda 2022), and the strategic sizing of operating regions to ensure adequate coverage and efficiency (Auad, Erera, and Savelsbergh 2023).

In contrast to the existing crowdsourced delivery literature, to the best of our knowledge, our work is the first to focus on fleet size planning for crowdsourced delivery drivers. This aspect has typically been overlooked, as crowdsourced drivers are paid per task or gig, leading platforms to incur no direct costs for maintaining a large pool of drivers. However, recent regulatory efforts aim at improving the working conditions of crowdsourced delivery workers have led some cities to impose minimum compensation and driver utilization standards (Stanford 2022) and in some cases a cap on the number of driver licenses a platform can hold (Fitzsimmons 2018, Casaletto and Nakhavoly 2024), limiting the maximum fleet size. To the best of our knowledge, this work is the first to consider fleet size planning of crowdsourced delivery drivers that enable sufficiently high driver utilization without compromising the platform service level and its ability to fulfill orders.

2.2. Fleet Size Planning in Crowdsourced Delivery

In the literature of crowdsourced delivery, fleet size planning is considered in hybrid systems that combine contracted and ad-hoc crowdsourced drivers. In this case, several models aim to determine the size and schedule of the contracted workforce considering varying availability of crowdsourced drivers. For example, Yildiz and Savelsbergh (2019) examine optimal service and capacity planning, in a hybrid delivery system involving both contracted and crowdsourced drivers, and aim to determine selection and pricing of service providers (restaurants/stored) to include in the service network, the number of full-time workers and compensation of crowdsourced drivers to incentivize their availability. Dai and Liu (2020) propose a deterministic mixed-integer programming model for fleet size selection across multiple depots, assuming known availability of crowdsourced drivers. Ulmer and Savelsbergh (2020) propose an optimization-simulation modelling framework for scheduling a fleet of full-time drivers under uncertainty in crowdsourced driver availability such that a minimum service level is maintained. Fatehi and Wagner (2022) investigate the optimal wages of crowdsourced drivers that incentivize their availability, as well as the optimal assignment of deliveries to both crowdsourced drivers and third-party logistics providers. They derive analytical expressions for delivery costs and system times considering guaranteed delivery windows. Furthermore, Goyal, Zhang, and Benjaafar (2023) develop a multi-stage stochastic programming problem where the first stage focuses on fleet sizing and location of contracted drivers, while subsequent stages handle order allocation and routing decisions of both contracted and crowdsourced drivers. Finally, Luy, Hiermann, and Schiffer (2024) develop a Markov decision process model to determine on the number of contracted drivers to hire, considering two types of crowdsourced

drivers—dedicated and opportunistic gig drivers—with varying order acceptance behaviors. The goal is to balance reliability and flexibility while ensuring a high platform service level.

In contrast to this stream of literature, to the best of our knowledge, our work is the first to consider fleet size planning of crowdsourced drivers rather than contracted drivers in a setting where the crowdsourced delivery platform depends solely on crowdsourced drivers. This aligns with platforms in practice such as UberEats and DoorDash that rely fully on crowdsourced capacity. This fleet size planning variant is not a trivial extension of existing works since crowdsourced drivers choose independently whether or not they want to offer service in a given day, and the platform does not directly control their scheduling. This creates decision-dependent or endogenous uncertainty, where the platform's crowdsourced driver fleet size decisions affect driver arrival at the operational level. Optimization problems with endogenous uncertainty are known to be notoriously difficult, as they typically result in nonlinear nonconvex formulations (Nohadani and Sharma 2018, Basciftci, Ahmed, and Shen 2021). Our purpose is to ensure that the fleet of crowdsourced drivers is selected to balance the trade-off between platform service level and driver utilization—a consideration motivated by recent efforts to improve working conditions for gig economy workers.

2.3. Compensation and Fair Task Allocation in Crowdsourced Delivery

In the crowdsourced delivery literature, driver compensation has been captured as a decision variable in the works of Qi et al. (2018), Le et al. (2021), and Barbosa, Pedroso, and Viana (2023). Other studies have considered how compensation influences a driver's likelihood of accepting a route or a match (Dahle et al. 2019, Horner, Pazour, and Mitchell 2021, Torres, Gendreau, and Rei 2022b). Benjaafar et al. (2022) studied the interaction of price, wage, labor supply, customer delay and demand by developing an equilibrium model that investigates the impact of labor pool size on worker welfare in on-demand service platforms. In terms of fair task allocation, fairness in last-mile delivery has been primarily discussed in the context of equitable service provision to customers (e.g., Chen et al. (2023)). However, fairness from the drivers' perspective, particularly concerning task allocation, is less explored. Basık et al. (2018) investigate fairness metrics in driver-order matching, while Alnaggar, Gzara, and Bookbinder (2024b) indirectly tackle fairness by developing a dynamic matching algorithm that ensures drivers earn a minimum wage and show that this approach reduces income disparities when compared to models focused solely on maximizing platform profit. Our study contributes to this body of literature by examining how to optimize fleet size decisions to enable high driver utilization targets, to address the criticism that on-demand drivers often face prolonged idle time between deliveries, which is a significant concern for drivers paid on a per-task basis as it leads to low overall earnings (Benjaafar and Hu 2020).

3. Fleet Size Planning Optimization Model

In this section, we first provide a general description of the problem and introduce the main notation. Then, we describe the proposed optimization model.

3.1. Problem Description

We consider a last-mile delivery network managed by a crowdsourced platform, where orders and drivers arrive stochastically over a planning horizon. Drivers do not pre-announce their work hours; instead, the platform controls driver availability by selecting the number of independent contractor requests to accept (i.e., pool size) at discrete periods $p \in \{1, \dots, P\}$, where P is the last period in the planning horizon. Each period p comprises $\hat{t}$ decision epochs, where $\hat{t} = \frac{\delta(p)}{\delta(t)}$ is the ratio of the duration of a period duration to that of an epoch. For example, with a 60-minute period and 5-minute epochs, each period would contain $\hat{t} = 12$ decision epochs. While periods p are used for tactical decisions, the shorter decision epochs t are used for operational decisions. For a given pool size x_p, driver arrivals follow a binomial distribution with entry probability q_p, capturing the uncertainty of whether drivers log into the app on a given day. The binomial distribution, used in previous research to model crowdsourced driver availability (e.g., Lei et al. 2020, Torres, Gendreau, and Rei 2022a), reflects the probabilistic nature of driver arrival. Thus, the platform does not know the exact number of drivers in ahead of time.

At the operational level, in addition to maximizing its matching profit, the platform aims (1) to meet as much customer demand as possible by achieving a targeted service level, and (2) to maximize the utilization of available drivers, ensuring their efficient assignment to delivery tasks. To evaluate the expected driver utilization, $L(\mathbf{x})$, and order fulfillment, $Q(\mathbf{x})$, for a given driver pool size $\mathbf{x}$, the platform solves a Markov Decision Process (MDP) that captures operational decisions considering the stochastic and dynamic arrivals of drivers and orders.

Thus, the platform's problem can be formulated as a *two-stage optimization* model. In the first stage, tactical decisions determine the fleet size x_p for each period $p \in \{1, \dots, P\}$. The second stage involves operational decisions modeled as an MDP, where drivers and orders are sequentially matched to maximize platform profit and minimize deviations from target service levels and driver utilization. In contrast to existing models in the literature, the proposed two-stage optimization model considers jointly optimizing fleet sizing and dynamic matching decisions, where the first-stage fleet size decisions affect the transition probabilities in the second-stage MDP, introducing endogenous uncertainty. Furthermore, to compute the platform's expected service level and expected driver utilization given a particular fleet size, the second-stage MDP incorporates comprehensive state variables that capture detailed driver and order attributes so as to accurately model the operational dynamics of the platform.

Table 1 Notation summary

Decision Variables		Parameters	
x_p	integer variable indicating the number of drivers the platform accepts in period $p \in [P]$	T	planning horizon, e.g., a day, where $T = P \times \hat{t}$
y_{tab}	binary variable which equals 1 if a driver with attributes a is matched with an order of attributes b at decision epoch t	p	period $p \in [1, P]$ indicates the discrete time intervals for pool size planning
		t	a decision epoch of the MDP model
		$\delta(p)$	Duration of period p in minutes
		$\delta(t)$	Duration of a decision epoch t in minutes
		$\hat{t}$	Number of decision epochs per period, where $\hat{t} = \frac{\delta(p)}{\delta(t)}$
		$L(\mathbf{x})$	Expected driver utilization given pool size vector $\mathbf{x}$
		$Q(\mathbf{x})$	Expected service level given pool size vector $\mathbf{x}$
		μ	target driver utilization level
		β	service level target
		w^s	service level target weight
		c	Per-unit cost vector of adding a driver to the fleet, where $c = (c_p)_{p \in [1,P]}$
		α	Penalty for deviations from service level or utilization targets

3.1.1. Key Modeling Assumptions Our model incorporates several assumptions to maintain computational tractability while capturing the essential dynamics of crowdsourced delivery systems. Drivers operate as independent contractors who make autonomous decisions about when to join the platform, with their arrival following a binomial distribution to reflect the inherent uncertainty in driver availability. Once in the system, drivers have preferred exit times that vary between half to one-and-a-half times the duration of a period, reflecting the flexible working patterns typical in gig work. On the demand side, order arrivals follow a Poisson process with predetermined origins, destinations, and fulfillment time windows; orders not fulfilled within these windows are considered lost. We model service times deterministically based on travel distances between driver location, pickup point, and delivery destination, a reasonable simplification for longer time scales where individual trip variations tend to average out. The platform coordinates the entire system through centralized driver-order matching decisions that aim to maximize profit while balancing service level and driver utilization objectives. This framework allows us to focus on the most critical sources of uncertainty in crowdsourced delivery: the probabilistic nature of driver participation and the stochastic patterns of customer demand.

3.2. Two-Stage Driver Fleet Size Optimization Problem

The platform aims to determine the optimal driver pool size vector for each time interval p throughout the planning horizon. Given the selected pool size, crowdsourced drivers join the platform probabilistically. The main notation used in the proposed model is defined in Table 1 and a comprehensive list of notation is available in Appendix EC.2. For brevity, we denote the set of running indices $\{1, 2, \ldots, P\}$ by $[1, P]$.

The objective of the platform is to choose the minimum pool size that minimizes deviations from the service level and driver utilization targets, modeled as follows:

$$\min_{\mathbf{x},\alpha} \ \boldsymbol{c}^\top \mathbf{x} + \alpha \tag{1a}$$

$$\text{s.t. } \alpha \geq w^s f^{serv}(\beta, Q(\mathbf{x})) + (1 - w^s) f^{util}(\mu, L(\mathbf{x})) \tag{1b}$$

$$\mathbf{x} \in \mathbb{Z}^P_{\geq 0} \tag{1c}$$

The objective function (1a) minimizes the total cost of selecting the driver fleet size vector $\mathbf{x} \in \mathbb{Z}^P_{\geq 0}$, where $\boldsymbol{c}$ represents the per-unit cost of adding a driver to the fleet. Additionally, it incorporates a penalty term, α, to account for deviations from service level or utilization targets. This penalty is computed in constraint (1b) as the weighted sum of penalty functions $f^{util}(\cdot), f^{serv}(\cdot)$ associated with the deviation from driver utilization and service level targets, respectively, where $0 \leq w^s \leq 1$ is the weight of the service level objective. Since α is minimized in the objective, constraint (1b) will be binding at optimality. The service level, $f^{serv}(\beta, Q(\mathbf{x}))$, and driver utilization, $f^{util}(\mu, L(\mathbf{x}))$, penalty functions are defined as follows

$$f^{serv}(\beta, Q(\mathbf{x})) = \begin{cases} \zeta^{serv}(\beta - Q(\mathbf{x})), & \text{if } Q(\mathbf{x}) < \beta \\ 0, & \text{otherwise} \end{cases}$$

$$f^{util}(\mu, L(\mathbf{x})) = \begin{cases} \zeta^{util}(\mu - L(\mathbf{x})), & \text{if } L(\mathbf{x}) < \mu \\ 0, & \text{otherwise} \end{cases}$$

We later vary the functional forms of $f^{serv}(\cdot), f^{util}(\cdot)$ in the computational experiments to assess their impact on the performance of the model.

In model (1), the calculation of $L(\mathbf{x})$ and service level $Q(\mathbf{x})$ for a given $\mathbf{x}$ requires solving a large-scale MDP problem that captures the platform's operational dynamics and decisions throughout the planning horizon, which we define in Section 3.3. Thus, model (1) has two stages representing decision problems faced by the platform at different time scales. The first stage pertains to tactical decisions, namely fleet size planning, whereas the second stage includes operational decisions, i.e., driver-order matching. Model (1) shares some similarities with the study of Ulmer and Savelsbergh (2020), which investigates scheduling contracted drivers in a hybrid crowdsourced delivery system

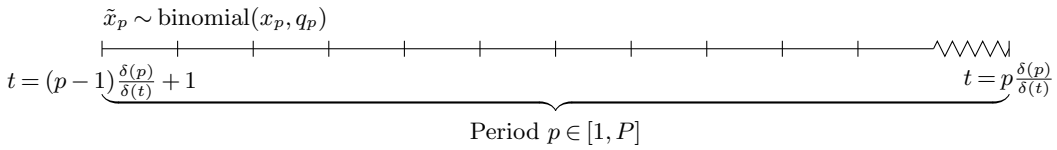

Figure 1 **Illustration of the relationship between a first-stage period $p \in P$ used in fleet size planning, decision epochs t in the second stage and the uncertain arrival of drivers. The figure shows period $p = 1$ containing decision epochs $t = 0$ to $t = \hat{t}$. Each subsequent period follows the same pattern.**

that combines both contracted and crowd drivers to maintain a minimum service level. While their study also employs a two-stage optimization model, it is cast using a simulation-optimization framework rather than an MDP in the second stage. Furthermore, we consider platforms that rely fully on crowdsourced drivers and consider not only the platform service level but also the utilization of drivers.

3.3. Second-Stage Markov Decision Process (MDP) Model

Given fleet size vector $\mathbf{x}$, at the operational level, both orders and drivers arrive randomly to the crowdsourced delivery platform within a given planning horizon T (e.g., a day). The arrival of orders follows a Poisson process with parameter λ per decision epoch t, such that the probability of k orders arriving in epoch t is $P(N_t = k) = \frac{\lambda^k e^{-\lambda}}{k!}$, where N_t is the random number of arrivals in epoch t. A decision epoch represents discrete time points within the planning horizon at which operational decisions are made. We assume that the duration of an epoch is shorter than the duration of a period for fleet size selection, i.e., $\delta(t) < \delta(p)$, where $\delta(\cdot)$ computes the duration in minutes. Figure 1 illustrates the relationship between a first-stage period p and a second-stage decision epoch t, with each period p containing $\hat{t}$ epochs, where $\hat{t} := \frac{\delta(p)}{\delta t}$. For instance, if $\delta(p)$ and δt are 60 and 5 minutes, respectively, the decision epoch corresponding to period $p = 2$ will range from $t = 13$ to $t = 24$.

When an order is announced, the platform receives information on its origin, destination, and fulfillment time window. For a given period p, driver arrival follows a binomial distribution $\tilde{x}_p \sim \text{binomial}(x_p, q_p)$, where x_p is determined in the first-stage of (1) and q_p is the probability of arrival to the platform, i.e., the probability that any given driver from the driver pool will log in or be available to accept matches during period p. Thus, the self-scheduling feature of drivers is captured by modeling driver arrivals as random binomial trials.

At the start of each decision epoch t, the platform assigns orders to drivers in a manner that maximizes its profit and service level, while considering how matching decisions impact drivers utilization. The outcomes of these assignments, along with the uncertainties in driver arrivals/departures and the arrival of new orders, transition the system to the state of the subsequent decision epoch. The components of the MDP are formally defined as follows.

3.3.1. State Space

The state variable S_t at decision epoch t is the minimal amount of information necessary and sufficient to make a decision. In the proposed MDP, the state variable comprises three main components: *driver information*, *order information*, and the *platform's performance metrics*.

Driver information. This includes the set of drivers in the system and their attributes, such as their availability for matching, current location and utilization up to the current time. At decision epoch t, this information is stored in a unique attribute vector a for each driver in the system, defined as:

$$a = (m_a, o_a, h_a, l_a, t_a^s, t_a^m, t_a^e)$$

where m_a is a binary indicator that equals 1 if the driver is available for matching, and 0 otherwise. o_a represents the driver's current location, and h_a indicates the number of epochs the driver has spent on-task up to time t, or up to the delivery of the current order if the driver is en-route. l_a denotes the fraction of time the driver has been utilized up to time t. The variables t_a^s, t_a^m, and t_a^e respectively represent the time the driver entered the platform, the time the driver becomes available after completing a delivery request, and the planned exit time of the driver.

We denote the set of drivers and their attributes at time t as $\mathcal{A}_t$, which is composed of $\mathcal{A}_t^{\text{avail}}$ for *available* drivers (i.e., those with $m_a = 1$), and $\mathcal{A}_t^{\text{route}}$ *en-route* drivers (i.e., those with $m_a = 0$). The vector of available drivers is denoted as $R_t = (R_{ta})_{a \in \mathcal{A}_t}$ where R_{ta} is a binary indicator that equals 1 if driver a is available at time t.

Order information. This describes the set of active orders, including their origin-destination (o-d) locations and time windows. Each order request is represented by a unique attribute vector b, defined as:

$$b = (o_b, d_b, [t_b^{\min}, t_b^{\max}])$$

where o_b and d_b denote the origin and destination of the order, respectively, and $[t_b^{\min}, t_b^{\max}]$ represents the time window for order fulfillment, i.e., the earliest and latest allowable delivery times.

We let $\mathcal{B}_t$ denote the set of all order attributes b at time t. To consider the possibility of no assignment, we define the expanded set $\mathcal{B}_t^+ := \mathcal{B}_t \cup \{0\}$, where the dummy order 0 represents no assignment. The vector of orders at time t is denoted by $D_t = (D_{tb})_{b \in \mathcal{B}_t}$, where D_{tb} indicates the number of orders with attribute b at time t.

Platform's performance metrics. In addition to the current drivers and orders, the platform keeps track of past demand fulfillment and driver utilization from previous decision epochs. The number of matched orders and the total number of orders up to, but not including epoch t are denoted by B_t^{matched} and B_t^{total}, respectively. Similarly, the platform tracks the utilization of all drivers who were previously available but have since exited the system. We denote the average utilization of these drivers as A_t^{util} and their total number as A_t^{total}. In the first epoch, the number of

matched orders and the utilization of drivers in the system are both set to zero. These performance metrics are compactly stored in vector $K_t = (B_t^{\text{total}}, B_t^{\text{matched}}, A_t^{\text{util}}, A_t^{\text{total}})$

Consequently, the state of the system at epoch t is expressed as

$$S_t = (R_t, D_t, K_t).$$

3.3.2. Action Space

The action space defines a time-feasible match between the set of drivers and the set of orders. At time t, the binary assignment of a driver with attribute a to an order with attribute b is denoted by y_{tab}. Similarly, y_{ta0} represents not assigning driver a to any order. At decision epoch t, the time required for a driver a to fulfill an order b is denoted by ι_{ab}. This time is converted into decision epochs using the following equation:

$$\iota_{ab} = \left\lceil \frac{\eta\big(||o_a - o_b|| + ||o_b - d_b||\big)}{\delta(t)} \right\rceil \tag{2}$$

where $||\cdot||$ is a norm that computes the distance between two nodes, η is a scalar converting the distance to a time estimate, and $\delta(t)$ is the length of a decision epoch. The set of time-feasible assignments, $\mathcal{D}_{ab}$, is defined as:

$$\mathcal{D}_{ab} = \{(a,b) \in \mathcal{A}_t^{\text{avail}} \times \mathcal{B}_t^{+} | t + \iota_{ab} \leq t_b^{\max}\}.$$

Thus, at decision epoch t, the decision vector $\mathbf{y}$ belongs to feasible set $\mathcal{Y}$ defined as

$$\mathcal{Y}_t = \left\{ \mathbf{y}_t \in \{0,1\}^{|\mathcal{A}_t^{\text{avail}}| \times |\mathcal{B}_t^{+}|} : \begin{array}{ll} \displaystyle\sum_{b:(a,b)\in\mathcal{D}_{ab}} y_{tab} = 1, & \forall a \in \mathcal{A}_t^{\text{avail}} \\ \displaystyle\sum_{a \in \mathcal{A}_t^{\text{avail}}} y_{tab} \leq D_{tb}, & \forall b \in \mathcal{B}_t \end{array} \right\}$$

The feasible set ensures that each available driver is assigned exactly once, which includes the null assignment. Furthermore, the total number of drivers assigned to orders of type b cannot exceed the number of orders of that type.

3.3.3. Transition Function and Post-decision State

The post-decision state $S_t^y = (R_t^y, D_t^y, K_t)$ captures the change in driver and order attributes immediately after a decision $\mathbf{y}_t$ is made. Given decision $\mathbf{y}_t$, a driver with attribute a transitions to attribute a', which is captured by the function $a^M(a, \mathbf{y}_t)$. To keep track of driver attribute transitions, we define indicator variable $\mathbb{1}(a^M(a,\mathbf{y}_t) = a')$, which equals 1 if a driver's attribute changes to a', and 0 otherwise. The post-decision resource state $R_t^y = (R_{ta}^y)_{a \in \mathcal{A}_t}$ is updated as:

$$R_{ta'}^y = \mathbb{1}(a^M(a,\mathbf{y}_t) = a') y_{tab}, \quad \forall a \in \mathcal{A}_t^{\text{avail}} \tag{3}$$

When a driver transitions from attribute a to a' as a result of a match, $l_{a'}$ is updated as the total utilization time after fulfilling order b, divided by the total active time in the system, i.e., $l_{a'} = \frac{h_a + \iota_{ab}}{t + \iota_{ab} - t_a^s}$. Similarly, if a driver is unmatched, $l_{a'}$ is updated as: $l_{a'} = \frac{h_a}{t+1-t_a^s}$.

For the post-decision order vector, we first define the travel time between the origin and destination of a demand as ι_b, discretized into decision epochs, as: $\iota_b = \left\lceil \frac{\eta(||o_b - d_b||)}{\delta(t)} \right\rceil$. Any order with an expired time window is considered lost and is not moved forward to the next epoch. The set of lost orders is denoted as $\overline{\mathcal{B}}_t := \{\mathcal{B}_t : t + 1 + \iota_b > t_b^{max}\}$. For the remaining orders, order information is updated as follows:

$$D_{tb}^y = D_{tb} - \sum_{a \in \mathcal{A}_t} y_{tab}, \qquad \forall b \in \mathcal{B}_t \setminus \overline{\mathcal{B}}_t \tag{4}$$

The number of matched orders is updated as

$$B_t^{\text{matched},y} = B_t^{\text{matched}} + \sum_{b \in \mathcal{B}_t} \sum_{a \in \mathcal{A}_t^{\text{avail}}} y_{tab}, \tag{5}$$

reflecting the addition of newly fulfilled orders to the cumulative total match. The average utilization of drivers is updated upon drivers' exit, and thus is not updated in the post-decision-state, i.e., $A_t^{\text{util},y} = A_t^{\text{util}}$. Similarly, the total number of drivers and the total number of orders remains unchanged in the post-decision state.

3.3.4. Stochastic Information and Pre-decision State

Let W_t denote the random information that is continuously arriving between epochs $t-1$ and t. $W_t = (W_{ta}, W_{tb})_{a \in \mathcal{A}_t, b \in \mathcal{B}_t}$, where W_{ta} denotes the arrival/departure of a driver with unique attribute identifier a during the interval between $t-1$ and t. Specifically, $W_{ta} = 1$ denotes the arrival of a driver, $W_{ta} = -1$ denotes the departure of a driver, and $W_{ta} = 0$ indicates no change. Similarly, W_{tb} is the number of new orders with attribute b that arrive during this time interval.

In period p, the number of drivers $\tilde{x}_p$ arriving is a binomial random variable with parameters x_p (fleet size decision) and q_p (probability of driver arrival). All $\tilde{x}_p$ drivers arrive in the first t' epochs of the period. Furthermore, we assume that a driver has a preferred exit at a random time t_a^e, uniformly distributed within the interval $[0.5\delta(p), 1.5\delta(p)]$, where $\delta(p)$ is the duration of a period. This implies that drivers remain for at least half of the duration of the period and overlap with a maximum of half of the duration of the subsequent period. A driver exits if $t_a^e \leq t + 1$. In the last epoch, all remaining drivers in the system exit.

Remove this subheading as per reviewer's suggestion:

Pre-decision State Transition

The arrival of random information transitions the system from post-decision state S_t^y to the next pre-decision state S_{t+1}, where $S_{t+1} = (R_{t+1}, D_{t+1}, K_{t+1})$. To capture these changes, we again use indicator variable $\mathbb{1}(a^M(a, W_t) = a')$, which equals 1 if a driver's attribute, a, changes to a', and 0 otherwise. Similar to the post-decision state transition function, $a^M(a, W_t) = a'$ determines the change in driver attribute due to W_t. The pre-decision resource vector, $R_{t+1} = (R_{(t+1)a})_{a \in \mathcal{A}_{t+1}}$, is computed as:

$$R_{(t+1)a'} = \mathbb{1}(a^M(a, W_t) = a') R_{ta}^x + W_{(t+1)a'}, \quad \forall a \in \mathcal{A}_{t+1} \tag{6}$$

The pre-decision demand vector, $D_{t+1} = (D_{(t+1)b})_{b \in \mathcal{B}_{t+1}}$, is the sum of carried forward orders D_{tb}^y and newly arrived orders $W_{(t+1)b}$, and is calculated as:

$$D_{(t+1)b} = D_{tb}^y + W_{(t+1)b}, \quad \forall b \in \mathcal{B}_{t+1} \tag{7}$$

The number of matched orders does not change from that in the pre-decision state ($B_{t+1}^{\text{matched}} = B_t^{\text{matched},y}$), while the total number of orders is updated by adding the new orders, i.e.,

$$B_{t+1}^{\text{total}} = B_t^{\text{total},y} + \sum_{b \in \mathcal{B}_t} W_{(t+1)b}. \tag{8}$$

Finally, the number of drivers that exit and their expected utilization of drivers are respectively updated as

$$A_{t+1}^{\text{total}} = A_t^{\text{total},y} + \sum_{a \in \mathcal{A}_t | W_{(t+1)a} < 0} W_{(t+1)a}, \tag{9}$$

and

$$A_{t+1}^{\text{util}} = \frac{(A_t^{\text{util},y} \times A_t^{\text{total},y}) \; + \sum_{a \in \mathcal{A}_t | W_{(t+1)a} < 0} l_a}{A_{t+1}^{\text{total}}} \tag{10}$$

and thus performance metric vector is updated to $K_{t+1} = (B_{t+1}^{\text{total}}, B_{t+1}^{\text{matched}}, A_{t+1}^{\text{util}}, A_{t+1}^{\text{total}})$. The system evolves as follows: $(S_0, \mathbf{y}_0, S_0^y, W_1, S_1, \mathbf{y}_1, S_1^y, W_2, S_2, \ldots, S_t, \mathbf{y}_t, S_t^y, \ldots, S_T)$.

3.3.5. MDP Contribution Function

The contribution function of the Markov Decision Process (MDP) model is given by:

$$J_t(S_t) = \max_{\mathbf{y}_t \in \mathcal{Y}_t} \left(C_t(S_t, \mathbf{y}_t) + \gamma \sum_{S_{t+1}} \text{prob}(S_{t+1} \mid S_t, \mathbf{y}_t, \mathbf{x}) J_{t+1}(S_{t+1}) \right) \tag{11}$$

where $C_t(S_t, \mathbf{y}_t) \coloneqq \sum_{b \in \mathcal{B}_t, a \in \mathcal{A}_t} (r(b) - \bar{c}(a, b))\, y_{tab}$ represents the immediate reward of matching decision $\mathbf{y}_t$ in state S_t, with $r(b)$ being the revenue from delivering order b and $\bar{c}(a, b)$ being the cost

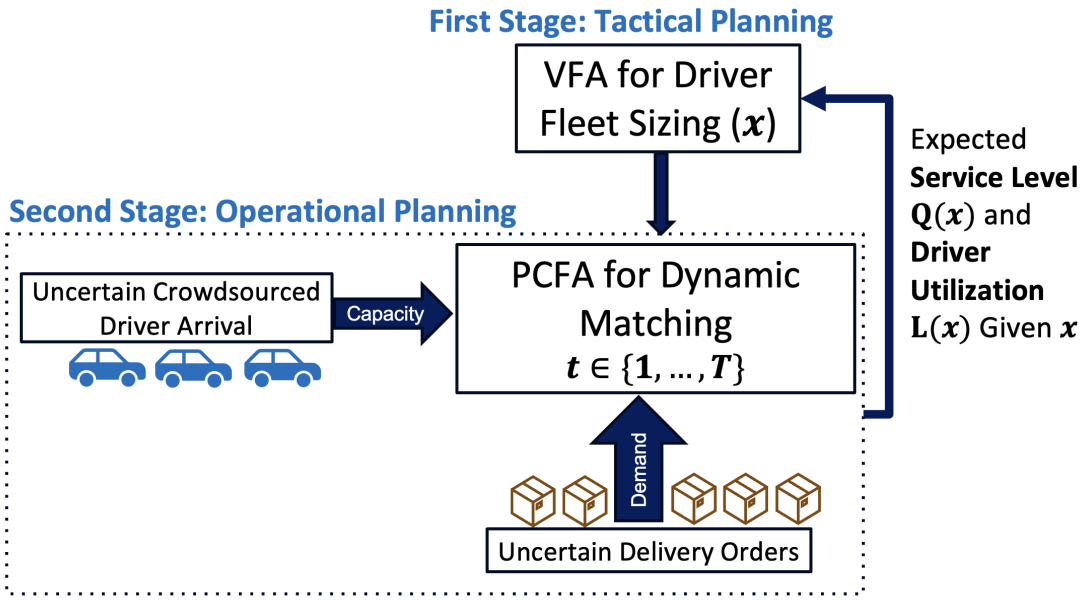

Figure 2 **Illustration of the VFA algorithm**

of matching driver a with order b. $\text{prob}(S_{t+1} \mid S_t, \mathbf{y}_t, \mathbf{x})$ denotes the probability of transitioning to state S_{t+1}, given the current state S_t, decision vector $\mathbf{y}_t$ and fleet size vector $\mathbf{x}$.

Equation (11) represents the Bellman recursion function, which determines the optimal decision $\mathbf{y}_t$ in state S_t. This function balances the immediate myopic profit with the discounted expectation of future profit, where γ is the discount factor $(0 < \gamma < 1)$. Additionally, we have the boundary condition $C_T(S_T) = -w^s f^{\text{serv}}(\beta, Q(\mathbf{x})) - (1 - w^s) f^{\text{util}}(\mu, L(\mathbf{x}))$, for all $S_T \in \mathcal{S}$, which computes the penalty incurred when the expected service level and/or driver utilization fall below the target. In the final decision epoch, the expected utilization of drivers and the expected platform service level over the full planning horizon are computed as

$$L(\mathbf{x}) = A_T^{util} \quad \text{and} \quad Q(\mathbf{x}) = \frac{B_T^{\text{matched}}}{B_T^{\text{total}}} \tag{12}$$

respectively, and the penalty functions $f^{serv}(\cdot), f^{util}(\cdot)$ are as defined in Section 3.2.

The proposed MDP model builds on Alnaggar, Gzara, and Bookbinder (2024b), with the following notable differences. While their approach employs an MDP for dynamic matching and compensation, our model utilizes an MDP as the second stage of an optimization problem, where first-stage decisions influence the MDP transition probabilities. Moreover, we augment the definition of the MDP state space, transition and reward functions to keep track of the platform's performance metrics.

4. Value Function Approximation Solution Method

Fleet size problem (1) is a two-stage optimization problem, where for each decision vector $\mathbf{x}$, one needs to solve the large-scale MDP described in Section 3.3 to compute the expected driver

utilization and service level. For realistic-sized problems that accurately capture the platform's dynamics, MDP problem (11) suffers from the curse of dimensionality in the state, action and outcome spaces, due to the large number of drivers and orders, their detailed attribute spaces and the large set of feasible decisions. Moreover, the underlying MDP is a function of the selected fleet size $\mathbf{x}$, since the fleet size choice affects the realization of random driver arrival. Thus, solving (1) exactly becomes computationally intractable for realistic problem instances.

In this section, we develop a value function approximation algorithm that iteratively searches the solution space of driver fleet sizes. The algorithm utilizes reinforcement learning, specifically Boltzmann exploration, to select a driver fleet size. For a given pool size $\mathbf{x}$, an approximation of the MDP (11) is solved using a rolling-horizon Parametric Cost Function Approximation (PCFA) to obtain estimates of $Q(\mathbf{x})$ and $L(\mathbf{x})$. The value function is iteratively updated guiding the selection of fleet sizes in subsequent iterations. This process continues until a stopping condition is reached, such as a predefined maximum number of iterations or no change in the best pool size for several consecutive iterations. An overview of the algorithm is depicted in Figure 2.

4.1. Value Function Approximation Algorithm

The solution space of model (1) is very large. Specifically, enumerating all possible pool sizes requires evaluating $(\overline{x})^P$ possible solutions, where $\overline{x}$ is an upper bound on the maximum pool size for each period. Since fleet size problem (1) is a multi-period problem, similar to Ulmer and Savelsbergh (2020), we leverage Value Function Approximation (VFA) - an approximate dynamic programming method for solving sequential decision problems (Powell 2022) - to efficiently navigate the solution space. VFA estimates the expected reward or value of being in a given state for each period. In a given time period p, the state $\overline{S}_p$ is the selected pool sizes of previous periods, i.e., $\overline{S}_p \coloneqq (x_{p'})_{p' \in [1,p-1]}$. The decision given state $\overline{S}_p$ determines the optimal pool size for all future periods. Thus, the value function is expressed as:

$$V(\overline{S}_p) \coloneqq \min_{[x_p, x_P] \in \mathbb{Z}_{\geq 0}^{P-p+1}} \sum_{p' \in [p,P]} c_{p'} x_{p'} + w^s f^{serv}(\beta, Q(x_p, \ldots, x_P)) + (1 - w^s) f^{util}(\mu, L(x_p, \ldots, x_P)) \tag{13}$$

The goal is to find the best fleet size by iteratively improving a value function that evaluates the quality of a potential fleet size. Given the large number of possible states $\overline{S}_p$, we employ separation as a dimensionality reduction technique to better approximate the value of the state space. This approach is utilized in Ulmer and Savelsbergh (2020) and is detailed in Powell (2011). Instead of storing all previous decisions, we approximate the state variable as $V(p, x_p)$, thus previous decisions are not included explicitly in the state space, but are accounted for implicitly in the approximation (Powell 2011). In the proposed VFA algorithm, we use Boltzmann exploration to balance exploring

the solution space and exploiting known value function estimates to select the next pool size during the search process.

The VFA procedure is outlined in Algorithm 1. Initially, the value function $V(p, x_p)\ \forall p \in [1, P], \forall x_p \in X$ is set to zero, where X is the solution space that can represent upper and lower bounds on the driver pool size of a given period obtained from domain knowledge. The best solution $\mathbf{x}^*$ is initialized to a randomized pool size and its value v^* is initialized to infinity (Line 2). For each iteration i of the algorithm, Boltzmann exploration is employed to generate a new solution $\mathbf{x}^{(i)}$ (Lines 4-8), where the current value function $V(\cdot)$, the solution space X, and the iteration count i are used to probabilistically select solutions that balance exploration and exploitation. We discuss the Boltzmann exploration step in detail in Section 4.2.

Once a new solution $\mathbf{x}^{(i)}$ is generated, its objective value $v^{(i)}$ is evaluated using Monte Carlo-based parametric cost function approximation (Line 9) for estimating the expected utilization and service level in the underlying MDP. This involves generating multiple sample scenarios, and solving a series of matching problems in a rolling horizon fashion to estimate the expected service level $Q(\mathbf{x}^{(i)})$ and driver utilization $L(\mathbf{x}^{(i)})$, reflecting the quality of the solution $\mathbf{x}^{(i)}$ at iteration i. The objective value $v^{(i)}$ is computed as:

$$v^{(i)} = \sum_{p \in [1,P]} c_p \mathbf{x}_p^{(i)} + w^s f^{serv}(\beta, Q(x_p^{(i)})) + (1 - w^s) f^{util}(\mu, L(x_p^{(i)})) \tag{14}$$

If the evaluated value $v^{(i)}$ is better than the current best value v^*, the best value v^* is updated to $v^{(i)}$, and the best solution $\mathbf{x}^*$ is updated to $\mathbf{x}^{(i)}$ (Lines 10-13). Subsequently, the new value function is computed as follows:

$$v^{(i)}(p, x_p) = \sum_{p' \in [p,P]} c_{p'} x_{p'} + (i+1) \left(w^s f^{serv}(\beta, Q(x_{p'}^{(i)})) + (1 - w^s) f^{util}(\mu, L(x_{p'}^{(i)})) \right) \tag{15}$$

where the second term is a penalty function that increases linearly in the iteration count i to discourage deviating from the service level and utilization targets as the iteration count increases and encourage exploration of the solution space at earlier iterations. Given the new value function estimate, $V(p, \mathbf{x}^{(i)})$ is updated in Line 14 as a weighted average of the value function of previous iteration and the new estimate as follows:

$$V(p, x_p) = (1 - \rho) V^{\text{previous}}(p, x_p) + \rho v^{(i)}(p, x_p) \tag{16}$$

where ρ is a step size defined as $\rho = \frac{1}{\sqrt{N(x_p)}}$, where $N(x_p)$ is the number of times the solution x_p has been encountered in previous iterations. The algorithm returns the best fleet size $\mathbf{x}^*$ and its associated value v^*. Next, we will describe the details of the Boltzmann exploration and the cost function approximation algorithms.

Algorithm 1 Value Function Approximation (VFA) Algorithm

1: **Inputs**: Maximum iteration count I, learning rate ρ, Number of periods P
2: Initialize $V^{(0)}, v^* \leftarrow \infty$, and the initial random best solution $\mathbf{x}^*$
3: **for** $i = [1, I]$ **do**
4: **if** $i = 1$ **then**
5: Set current fleet size $\mathbf{x}^{(i)} \leftarrow \mathbf{x}^*$ // First Solution
6: **else**
7: $\mathbf{x}^{(i)} \leftarrow$ Boltzmann Algorithm 2 // Generate New Pool Size
8: **end if**
9: $v^{(i)} \leftarrow$ Cost Function Approximation Algorithm 3 Given $\mathbf{x}^{(i)}$ // Evaluate Pool Size
10: **if** $v^{(i)} < v^*$ **then**
11: $v^* \leftarrow v^{(i)}$ // Update of Best Found Value
12: $\mathbf{x}^* \leftarrow \mathbf{x}^{(i)}$ // Update Best Pool Size
13: **end if**
14: Update value function $V(p, \mathbf{x}_i)$ using Equation (16)
15: **end for**
16: **return** $\mathbf{x}^*, v^*$

4.2. Boltzmann Exploration

At each iteration of Algorithm 1, Boltzmann exploration, a reinforcement learning technique, is employed (Line 7) to navigate the large decision space and select a candidate fleet size x_p, $p \in [1, P]$. Boltzmann exploration aims to balance the exploration-exploitation trade-off when navigating a solution space by assigning a probability to each potential solution, which is calculated using an exponential function of the state's estimated value. Specifically, the probability $\text{Prob}(x_p)$ of selecting fleet size x_p in period p is given by:

$$\text{Prob}(x_p) = \frac{e^{-V(p,x_p)/\tau}}{\sum_{x'_p \in X} e^{-V(p,x'_p)/\tau}} \tag{17}$$

where $V(p, x_p)$ is the estimated value (cost) of solution x_p and τ is a temperature parameter that controls exploration. A higher τ yields a more uniform probability distribution over the actions, thereby favoring exploration. Conversely, a lower τ results in a more peaked probability distribution, promoting the exploitation of the best-known solutions. The steps of the Boltzmann exploration are detailed in Algorithm 2. We define d as the value function spread of all possible x_p solutions in period p, adjusted by the tolerance parameter ϵ to prevent division by zero (Line 4). The temperature τ is computed in Line 5. Initially, τ is set to a high value to encourage the exploration of the decision space due to the lower confidence in the value function estimates, as the algorithm has not

yet extensively explored the solution space. As the number of iterations increases, the algorithm gradually shifts towards exploitation, making higher-valued actions more likely to be chosen while ensuring that lower-valued actions still maintain a non-zero probability of selection. This contrasts with fixed probability methods such as the epsilon-greedy approach. Under certain conditions, the probabilistic approach of Boltzmann exploration can lead to more efficient exploration and potentially better convergence properties (Sutton and Barto 2018).

For each candidate solution $x_p \in X$, where $X \coloneqq [0, \overline{x}]$, let $u(k)$ represents the weight of the k-th potential solution x_p, $k \in [0, \overline{x}]$, calculated as $e^{-V(p,x_p)/\tau}$ (Line 9). This weight reflects the probability of selecting each solution based on its estimated value, with lower-cost $V(p, x_p)$ yielding higher weights. The sum of these weights computed in Line 12, which we denote as ω, normalizes the probability distribution, ensuring that the total probability across all potential solutions sums to 1. A uniform random variable $\hat{u}$ is sampled from the range $[0, \omega]$ (Line 13) to introduce stochasticity in the selection process. The cumulative weight sum, u^{sum}, is then incrementally accumulated by iterating through the weights $u(k)$ until u^{sum} exceeds u (Lines 14-19). This procedure probabilistically determines the selected solution x_p^* for period p (Line 20), balancing exploration and exploitation as guided by the temperature parameter τ.

4.3. Parametric Cost Function Approximation (PCFA) for Solving MDP (11)

For each candidate solution $\mathbf{x}$ in the value function approximation algorithm, we need to solve MDP (11) to obtain the expected driver utilization $L(\mathbf{x})$ and service level $Q(\mathbf{x})$. However, for realistically sized problems, solving MDP (11) becomes computationally intractable due to the curse of dimensionality in the state, action, and outcome spaces. This complexity arises from the detailed attributes associated with both drivers and orders, which are required to accurately track the status of drivers throughout the planning horizon and compute their utilization. Furthermore, the MDP must be solved at each iteration of the value function approximation algorithm. Therefore, it is imperative to develop an efficient approximation method to solve the MDP.

To address these challenges, we propose a Monte-Carlo-based cost function approximation method that solves a series of matching problems in a rolling horizon manner over a set of discrete scenarios Ξ. This contrasts with the myopic heuristic used in Ulmer and Savelsbergh (2020) to solve the platform's operational problem. In our proposed PCFA algorithm, each scenario $\xi \in \Xi$ represents a sample path of the random information in the system. Specifically, a sample path denotes a realization of the binomially distributed driver arrival $\tilde{x}_p$, the corresponding random arrival (t_a^s) and preferred departure (t_e^s) epochs of those drivers, as well as the realization of demand arrivals for epoch $t \in [1, T]$. For each scenario ξ, at decision epoch t an optimization model selects the best matching decisions given driver and order information. Based on the chosen matching decisions $\mathbf{y}$

Algorithm 2 Boltzmann Exploration

Input: Current value function $V(\cdot)$, solution space $X \coloneqq [0, \overline{x}]$, iteration i, constants ϵ, ω
$\mathbf{x}^* \leftarrow \mathbf{0}_P$ // Initialize solution
for $p \in [1, P]$ **do**
  $d \leftarrow \max_{x_p \in X_p} V(x_p) - \min_{x_p \in X_p} V(x_p) + \epsilon$ // Value spread
  $\tau \leftarrow \frac{10 \times d}{i}$ // Temperature parameter
  $u \leftarrow \mathbf{0}_{|X|}$ // Initialize weights
  $k \leftarrow 0$ // Initialize solution index
  for $x_p \in X$ **do**
    $u(k) \leftarrow e^{-V(p, x_p)/\tau}$ // Calculate weight
    $k \leftarrow k + 1$
  end for
  $\omega \leftarrow \sum_{k \in X} u(k)$ // Sum of weights
  $\hat{u} \sim \mathcal{U}(0, \omega)$ // Sample random variable
  $u^{\text{sum}} \leftarrow 0$ // Temporary weight sum
  $k \leftarrow 0$ // Initialize solution index
  while $u^{\text{sum}} \leq \hat{u}$ **do**
    $u^{\text{sum}} \leftarrow u^{\text{sum}} + u(k)$
    $k \leftarrow k + 1$
  end while
  $x_p^* \leftarrow X(k)$ // Return element k in the solution set X
end for
return $\mathbf{x}^* = (x_p^*)_{p \in [1,P]}$

and the arrival of new information between epochs t and $t+1$, the status of the drivers and order information is updated using Equations (3)-(7). The model is then resolved for epoch $t+1$ and the process continues until the last decision epoch T.

The matching problem is formulated as follows

$$\max_{\mathbf{y}} \sum_{a,b \in \mathcal{D}_{ab}} \pi_{ab} y_{tab} \tag{18a}$$

$$\sum_{b:(a,b) \in \mathcal{D}_{ab}} y_{tab} = 1, \qquad \forall a \in \mathcal{A}_t^{\text{avail}} \tag{18b}$$

$$\sum_{a \in \mathcal{A}_t} y_{tab} \leq D_{tb}, \qquad \forall b \in \mathcal{B}_t \tag{18c}$$

$$y_{tab} \in \{0, 1\}, \qquad \forall (a, b) \in \mathcal{D}_{ab} \tag{18d}$$

where π_{ab} denotes the profit of matching a driver with attributes a to an order with attributes b. In the simple myopic case, $\pi_{ab} \coloneqq r(b) - \overline{c}(a,b)$ represents the revenue from fulfilling order b minus the cost of matching driver a with order b. However, the myopic policy does not account for the impact of matching decisions on driver utilization l_a or the proximity of orders to their delivery deadlines $t_b^{\max}$. To address this, we modify the objective function using parametric cost function approximation, which allows us to incorporate these factors so as to partially capture the effect of decisions made at time t on the future (Powell 2019).

The proposed parametric cost function approximation incorporates non-negative penalty (utility) terms to prioritize matching drivers who fall short of the target utilization ($h_a < \mu$) and orders nearing their delivery deadlines ($t_b^{\max}$). This proactively ensures that the driver utilization remains close to the target and orders that are about to expire are prioritized for matching to ensure high service level. The profit of matching driver a with order b under the parametric cost function approximation is given by:

$$\pi_{ab} = r(b) - \overline{c}(a,b) + (1-w^s)g^{util}(l_a) + w^s g^{serv}(t, t_b^{\max}) \tag{19}$$

Here, the revenue is defined as $r(b) \coloneqq \theta_b(w_b)$, where w_b is the distance between the origin and destination of an order with attributes b, and $\theta_b(w_b)$ is the revenue as a function of distance. The cost is given by $\overline{c}(a,b) \coloneqq \overline{\theta}_{ab} w_{ab}$, where w_{ab} is the travel distance of fulfilling delivery b by driver a, and $\overline{\theta}_{ab}$ is the cost per unit distance. Finally, $g^{util}(l_a)$ and $g^{serv}(t, t_b^{\max})$ are non-negative penalty functions; the former measures deviations from the target utilization, based on a driver's current utilization l_a, while the latter prioritizes the fulfillment of order b as it approaches its delivery deadline $t_b^{\max}$. We propose the following piecewise-linear penalty functions:

$$g^{util}(l_a) = \begin{cases} \zeta(\mu - l_a) & \text{if } l_a < \mu \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

$$g^{serv}(t, t_b^{\max}) = \zeta \max\left[1 - \frac{t_b^{\max} - t}{t_b^{\min} - t_b^{\max}},\ 0\right] \tag{21}$$

where non-negative scalar ζ controls the magnitude of the penalty. The proposed penalty functions are motivated by their simple and intuitive interpretation. Specifically, when a driver's utilization falls below the target μ, the penalty's magnitude is a linear function of the difference between the actual utilization and the target. Similarly, for each order b, the penalty increases linearly as the delivery time window shrinks. In Equation (19), the revenue $r(b)$ and cost $\overline{c}(a,b)$ are on a similar scale by definition. Similarly, $g^{util}(l_a)$ and $g^{serv}(t, t_b^{\max})$ are scaled comparably since they range between $[0, \zeta]$. The value of ζ must be selected to align the penalty terms with the profit terms. When $\zeta \gg \max(r(b) - \overline{c}(a,b))$, the penalty term dominates the objective if driver utilization or

demand fulfillment fall below targets. In Appendix EC.3.2, we introduce alternative quadratic and exponential penalty functions to evaluate the impact of different penalty functional forms on the algorithm's performance.

The steps of the cost function approximation method are outlined in Algorithm 3. We repeat the rolling horizon procedure over $|\Xi|$ sample paths (Line 2) since relying on a single sample path can result in high sensitivity to the specific demand and driver realization, leading to results that may not be representative of the expected platform performance. For each sample path, the rolling horizon procedure (Lines 3 to 6) solves a sequence of matching problems (18), one for each decision epoch, updating the state of the system between consecutive epochs. Then the service level and driver utilization at that sample path are computed using (12) and are denoted by Q_ξ and L_ξ, respectively (Line 7). Finally the mean service level and driver utilization are computed as the average of the Ξ sample paths (Line 9) and $v^{(i)}$ is calculated by Equation (14) (Line 10) and returned to the VFA algorithm.

Algorithm 3 Monte-Carlo-based Cost Function Approximation for Estimating Service Level and Driver Utilization

1: **Input**: Current solution $\mathbf{x}^{(i)}$, number of decision epochs T, sample scenarios Ξ
2: **for** $\xi \in \Xi$ **do**
3: **for** $t \in [1, T]$ **do**
4: Solve optimization model (18)
5: Update the state variable: $(S_t^y|S_t, y_t)$, $(S_{t+1}|S_t^y, W_{t+1}(\xi))$ using Equations (3)-(10)
6: **end for**
7: **Compute** Q_ξ, L_ξ using Equations (12)
8: **end for**
9: **Compute** $Q(\mathbf{x}) = \frac{\sum_{\xi\in\Xi} Q_\xi}{|\Xi|}, L(\mathbf{x}) = \frac{\sum_{\xi\in\Xi} L_\xi}{|\Xi|}$
10: **Return** $v^{(i)}$ computed by Equation (14)

5. Computational Experiments

We test the proposed VFA algorithm on both synthetic datasets and real-world instances from the Chicago ridehailing dataset. Section 5.1 outlines the experimental setup and instance generation process. Section 5.2 presents the performance results of the VFA algorithm, analyzing key metrics for both the platform and drivers under varying priority weights assigned to service level and utilization targets. Sensitivity analysis is conducted to evaluate the robustness of the results across several critical parameters, including driver arrival probabilities, the length of scheduling periods in the first stage, and the penalty functional forms in both the value function approximation and

cost function approximation. Section 5.3 presents the application of the algorithm on larger-scale real-world instances. Finally, Section 5.4 highlights the main insights from the experiments and the practical implications for crowdsourced delivery platforms.

5.1. Experimental Setup and Performance Benchmarks

In both the synthetic and real-world instance, we use a planning horizon of 16 hours, which corresponds to a platform operating from 8:00 am to 12:00 am. The length of scheduling periods p for fleet size selection is set to 1 hour, resulting in 16 periods per day. We conduct sensitivity analysis on the duration of p to understand its impact on the results. At the operational level, the length of a decision epoch in the second-stage MDP problem is set to 5 minutes, allowing for frequent state updates and matching of incoming orders. This results in $T = 192$ decision epochs per day.

To account for spatial and temporal demand uncertainty, we set a target minimum service level of $\beta = 0.95$, acknowledging that some demand may be unmet due to these uncertainties. The delivery time window for orders is set at 90 minutes, to reflect the short time frame of crowdsourced delivery operations. Orders that are not fulfilled within their delivery deadlines are considered lost or fulfilled by third-party providers. The target driver utilization is set to $\mu = 0.8$, to allow for some idle repositioning that accounts for spatial imbalances over the service network. In the main experiments, driver arrival probability in the binomial distribution is set to $q_p = 0.7,\ \forall p \in [P]$; which we later vary to understand its impact on performance. The unit cost c_p of fleet size selection is set to one, reflecting the objective of finding the minimal fleet size that meets performance targets.

Service Network and Demand Generation

In the synthetic instances, demand follows a Poisson distribution with an arrival rate of $\lambda = 10$ per decision epoch, where order origins and destinations are uniformly distributed across the service region. The service region is a 10×10 grid, with each unit of distance representing 3 km. The grid contains 400 discrete locations, each is the center of 0.5×0.5 square. Drivers arrive at random locations within the grid network, each location being equally likely.

For the real-world dataset, we utilize the Chicago ride-hailing trip dataset (Chicago Data Protal 2024) similar to (Alnaggar, Gzara, and Bookbinder 2024b). This dataset encompasses trip data from 2018 to 2022. Given the large volume of trips in a single day, we focus specifically on those with an origin or destination in the downtown area (community areas 8, 32, and 33). The data is originally reported in 15-minute intervals; however, to generate demand information for each decision epoch, we randomly divide the trips into three 5-minute intervals.

For both real-world and synthetic instances, the revenue of an order b consists of two components, reflecting the pricing model of meal delivery services like UberEats (Alnaggar, Gzara, and Bookbinder 2024b): a distance-based charge and a fixed delivery charge. The distance-based charge

is a step function that equals 4, 6, and 7 for trips less than 2 km, between 2 and 5 km, and more than 5 km, respectively. The fixed charge is 30% of the meal cost, which is uniformly distributed between \$3 and \$12. The cost $\bar{c}(a,b)$ represents the driver's pay, calculated as \$1.1185 per kilometer traveled, \$0.20 per minute of travel time, plus a fixed charge of \$2.50 for pickup and delivery.

Algorithmic Parameter Values

In the Monte-Carlo-based cost function approximation algorithm, we set the number of sample paths to 10, i.e., $|\Xi| = 10$ and conduct sensitivity analysis on varying the number of sample paths to understand its impact on the results. We set the lower and upper limits of the pool size of each period in Boltzmann exploration to 1 and 250 drivers, respectively. The lower limit ensures that we can compute an estimated utilization rate at each period, while the upper limit is a sufficient large number based on preliminary testing that indicated the pool size per period does not exceed that value. The number of iterations in the value function approximation algorithm is set to 1000. The penalty constants for deviating from the target service level and target driver utilization are set to 250, i.e., $\zeta^{serv} = \zeta^{util} = 250$.

Performance Benchmarks

To assess the quality of the solutions obtained by the VFA algorithm, we compare its performance against the following benchmark policies:

a. **Deterministic driver behavior fleet size planning policy**. This benchmark assumes perfect driver compliance, disregarding any uncertainty in driver availability. The platform selects fleet sizes based on the assumption that all scheduled drivers will be available in their designated periods during the planning horizon.

b. **Constant fleet size planning policy**. Under this policy, the platform selects a single fleet size for the entire planning period, without the flexibility to adjust fleet sizes dynamically over time. This benchmark serves to evaluate the benefits of temporal fleet size adjustments.

c. **Fleet size decisions as a function of expected demand**. Under this policy, fleet size decisions are selected as a simple function of demand arrival λ. Specifically, for $p \in P$, we set $x_p = \frac{\frac{\delta(p)}{\delta(t)}\lambda}{\rho}$. The numerator converts the order arrival rate to an arrival rate per period, while the denominator, ρ, is the estimated number of orders that a driver can complete during a period p.

d. **Second-stage greedy matching policy based on travel time.** Under this policy, we replace Line 4 of Algorithm 3 with a greedy matching heuristic, which matches drivers to orders based on ascending order of travel time.

e. **Second-stage myopic policy**. Under this policy, we replace Line 4 of Algorithm 3 with solving a myopic matching problem. This matching problem is a variant of (18), where the

profit of matching a driver with an order is expressed as $\pi_{ab} = r(b) - \overline{c}(a,b)$, i.e., there is no penalty associated with low driver utilization or service level.

Furthermore, we test several variants of the proposed VFA algorithm to verify its effectiveness and robustness. These variants include:

a. **Impact of penalty/utility function selection:** We examine how different choices for the driver utilization and service level penalty functions ($f^{util}(\cdot)$ and $f^{serv}(\cdot)$) and utility functions in the PCFA ($g^{util}(\cdot)$ and $g^{serv}(\cdot)$) influence the solution quality and platform performance.

b. **Impact of the number of sample paths in the Monte-Carlo-based PCFA:** We analyze how varying the number of scenarios in the Monte Carlo parametric cost function approximation affects the accuracy and convergence of the VFA solutions.

5.2. Results

In this section, we evaluate the performance of the proposed VFA algorithm for fleet size selection under varying priority weights assigned to service level and driver utilization targets. Due to the computational demand of the real-world instances, we focus here on the synthetic instances, then present results for the real-world instances in the subsequent section. We provide a detailed analysis of the quality of driver-order matches by examining metrics such as driver utilization distribution, empty distance traveled by drivers, and daily platform profit. Additionally, we perform a sensitivity analysis on key parameters to assess their impact on the best fleet size.

5.2.1. Overall Performance Metrics

We evaluate VFA Algorithm 1 using six different values for the service level weights, specifically $w^s \in \{0, 0.2, 0.5, 0.6, 0.8, 1\}$, to obtain the best driver pool size $\mathbf{x}^*$. Recall that the weight of the driver utilization target is defined as $1 - w^s$. Thus, when $w^s = 0.5$, the platform assigns equal importance to both performance metrics. Conversely, a weight of $w^s = 1$ indicates that the platform only considers the service level target and neglects driver utilization. For each specified weight w^s, once the optimal pool size $\mathbf{x}^*$ is determined, we evaluate its operational performance using Algorithm 3 on 100 test sample paths that are different from those used in training, with each sample representing a different operational day. Table 2 presents the best aggregate pool size (i.e., $\sum_{p\in[P]} x_p^*$) corresponding to each weight and provides a comprehensive analysis of various performance metrics, including demand fulfillment, driver utilization, the proportion of drivers achieving the utilization target, driver empty distance, platform daily profit, driver idle time, and total time spent by drivers in the system. For all performance metrics, we report the mean value and the standard deviation (SD) across the 100 test instances.

Table 2: Performance Metrics

Metric	$w^s=0$	$w^s=0.2$	$w^s=0.5$	$w^s=0.6$	$w^s=0.8$	$w^s=1$
Aggregate Fleet Size	88	343	376	378	381	1078
Demand Fulfilled (%)						
Mean	29	94	97	98	98	97
SD	2	2	2	1	2	1
Driver Utilization (%)						
Mean	100	97	93	93	93	37
SD	0	7	10	9	10	31
Drivers Meeting Utilization Target (%)						
Mean	91	88	83	83	83	13
SD	29	28	27	27	27	4
Driver Empty Distance (km)						
Mean	1.74	2.28	2.52	2.53	2.56	1.93
SD	1.16	1.76	1.96	1.99	2.04	2.26
Platform Daily Profit						
Mean ($\times 10^3$)	6.03	14.1	14.1	14.1	14.0	14.6
SD ($\times 10^3$)	0.36	0.32	0.34	0.36	0.37	0.32
Driver Idle Time (min)						
Mean	0.00	2.25	4.51	4.48	4.48	40.4
SD	0.00	4.93	6.85	6.60	6.75	22.4
Driver Time in System (min)						
Mean	64.8	65.7	66.0	65.9	65.8	65.0
SD	17.9	17.8	17.9	17.9	17.9	18.7

Observe from Table 2 that when the platform prioritizes service level exclusively ($w^s=1$), the resulting total fleet size becomes substantially large (1078 drivers), while the average driver utilization rate is considerably low (37%). Although this scenario yields the highest platform profit (approximately \$14,600 daily), the mean driver idle time is also very high around 40 minutes, suggesting under-utilization of the majority of the driver pool.

Conversely, when w^s is reduced to 0.8, placing a 20% weight on driver utilization, the driver pool size significantly decreases by 64.66% from 1078 (at $w^s=1$) to 381. This adjustment results in a marked improvement in average driver utilization, rising from 37% to 93%. Interestingly, this substantial enhancement in driver utilization leads to only a marginal reduction in platform profit, which decreases by 4.11%. Additionally, the average driver idle time drastically drops to 4.48 minutes compared to 40.4 minutes, indicating more efficient use of the driver pool.

Further reductions in the service level weight to 0.6, 0.5, and 0.2 show less pronounced changes in fleet size, platform profit, and utilization, with the most notable change occurring at $w^s=0.2$. At this point, the driver pool size decreases to 343, causing a slight decline in mean demand fulfillment

from 97% to 94% and a marginal improvement in driver utilization from 93% to 97%. An interesting observation is that when $0 < w^s < 1$, the average empty distance traveled by drivers is higher than when $w^s = 1$, despite the lower utilization in the latter scenario. This may be attributed to the excess availability of drivers, which allows the platform to match orders with drivers located closer to pickup points.

In the extreme case where driver utilization is the sole priority ($w^s = 0$), the fleet size further declines to 88 drivers, accompanied by a significant drop in mean demand fulfillment (29%) and platform profit ($6,030). Overall, these results suggest that incorporating driver utilization into fleet size decisions can lead to a more manageable driver pool with only a slight reduction in platform profit while significantly enhancing driver utilization and minimizing idle time between order assignments.

5.2.2. Analysis of the Matching Outcome on the Platform and Drivers

To evaluate the quality of matches for the best fleet sizes under varying priority weights, we analyze detailed metrics for both drivers and the platform. Figure 3 shows the distribution of driver utilization across 100 test instances, discretized into five utilization brackets ranging from 0 to 100%. Observe that, when the service level target is the sole objective ($w^s = 1$), approximately 35% of drivers have a utilization rate between 0 and 20%, representing the largest proportion of drivers in any single bracket. Conversely, only about 15% of drivers are frequently matched, achieving a utilization rate between 80% and 100%.

On the other hand, when both service level and utilization targets are considered, i.e., $0 < w^s < 1$, at least 83% of drivers achieve high utilization, and those who do not meet the target fall into the next lower bracket, with utilization rates between 60% and 80%. It is important to note here that the proposed model 1 focuses on the average utilization across the entire driver pool rather than individual driver utilization, which explains why some drivers may not meet the utilization target. In the extreme scenario where driver utilization is the sole priority ($w^s = 0$), all drivers reach a utilization rate between 80% and 100%. This is because the platform maintains a smaller pool of drivers who are highly utilized, though this comes at the cost of lower order fulfillment.

Figure 4 illustrates the distribution of empty travel distances for drivers to order pickup locations after being matched, across different priority weights. Interestingly, the shortest average distance occurs when $w^s = 1$, which is due to the high number of drivers in the system leading to more efficient matches and fewer empty miles. Conversely, when the driver utilization target is considered, although driver utilization significantly increases, the average distance to pickup also rises by up to 31%, from 1.93 km to 2.56 km, as there are fewer drivers available. When $w^s = 0$, the average pickup distance decreases again because many orders remain unfulfilled, allowing the platform to prioritize closer, more efficient matches to maximize driver utilization.

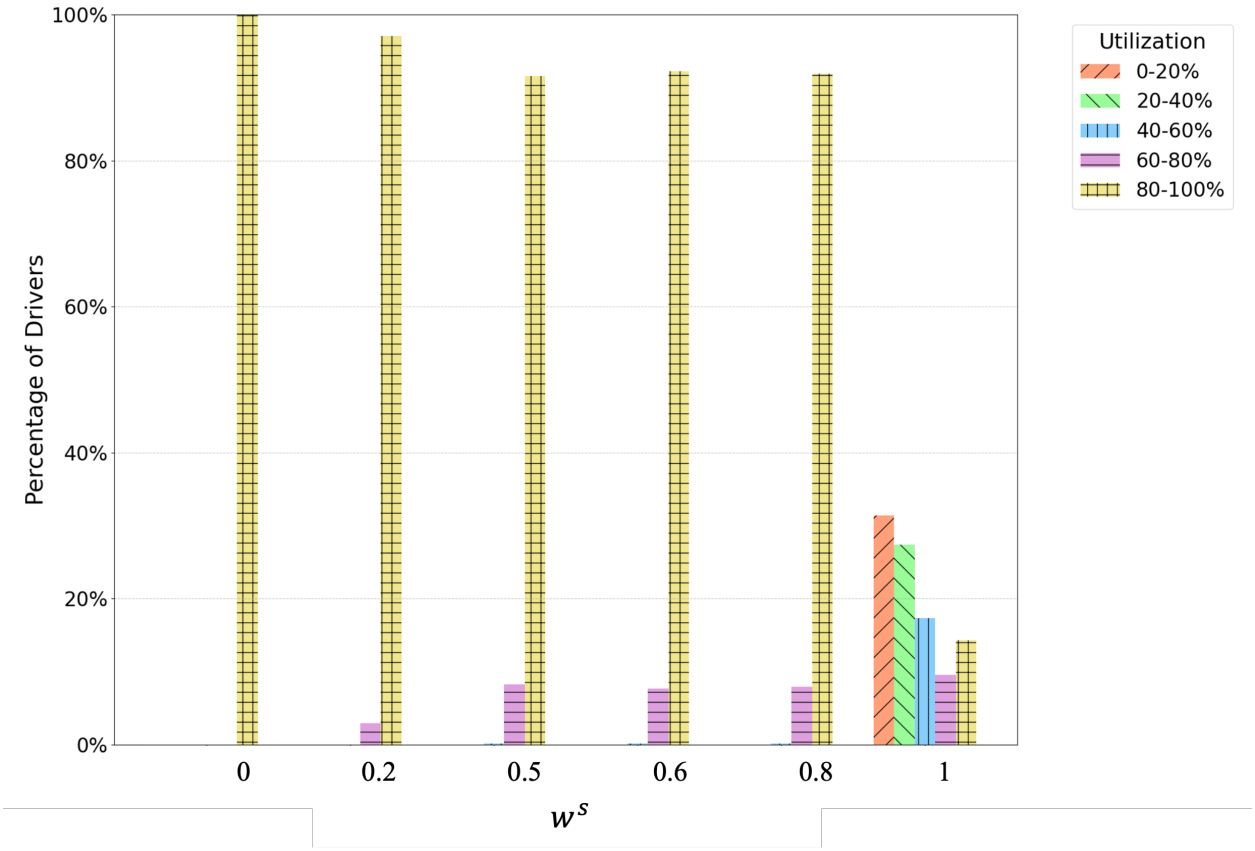

Figure 3 **Driver Utilization Distribution**

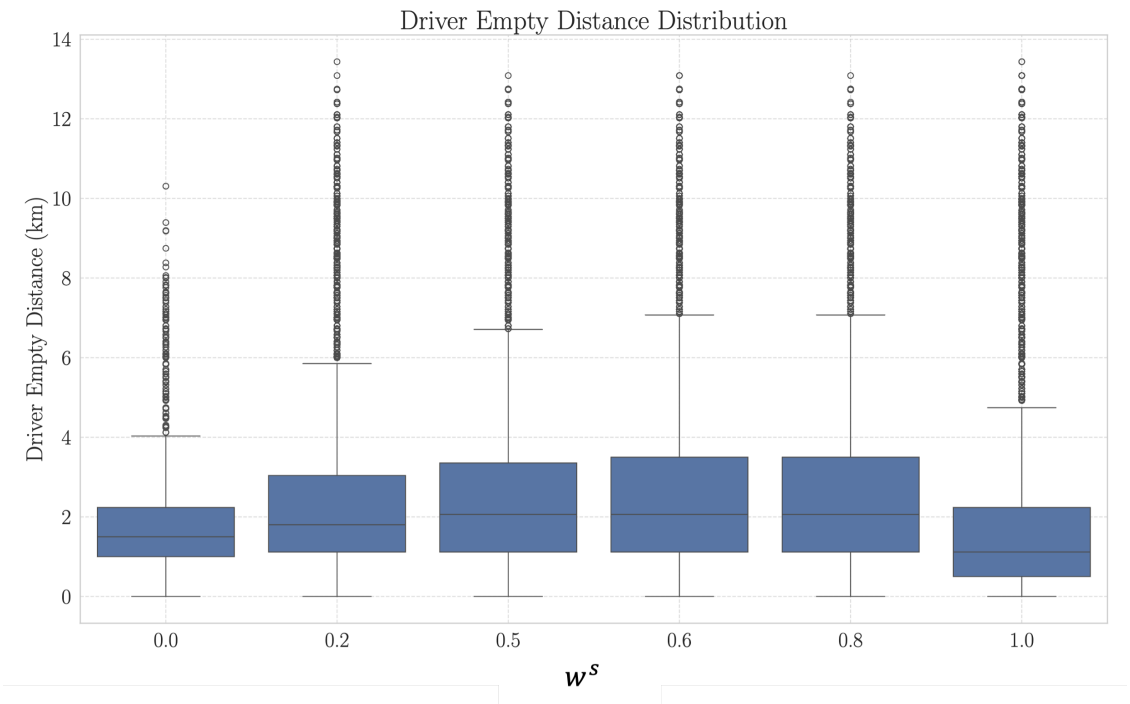

Figure 4 **Distribution of driver empty travel distance for different weight parameters**

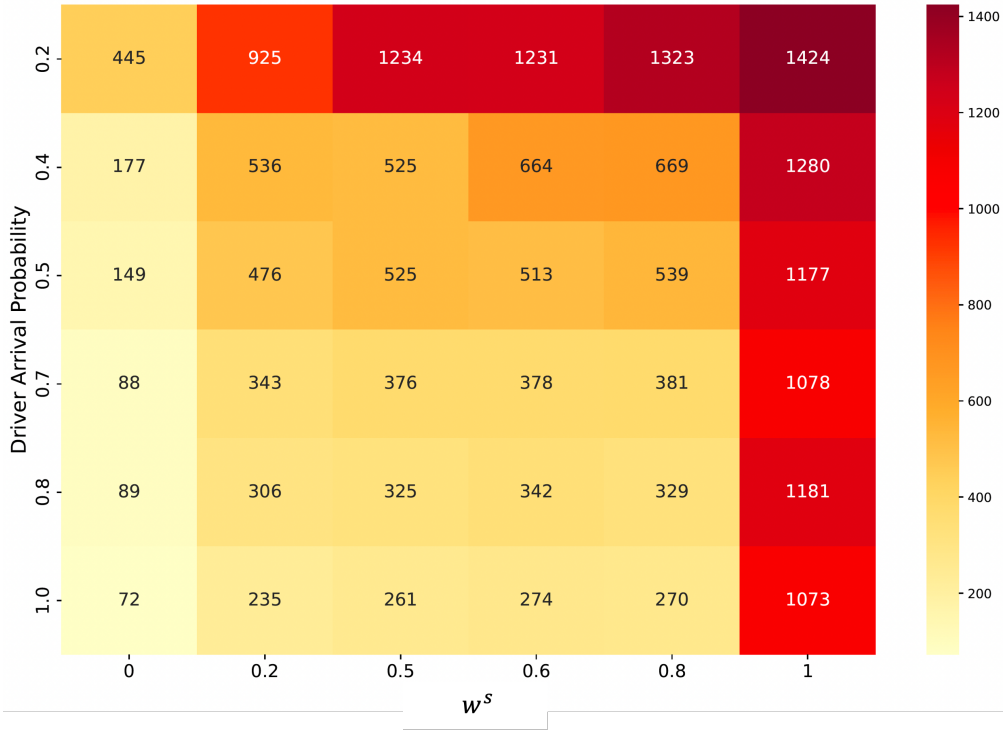

Figure 5 **Heatmap Visualization of the Aggregate Fleet Size as a Function of Driver Arrival Probability and Service Level Weight**

Table 3 presents the platform's daily profit across 100 test instances for different priority weights. When both driver utilization and demand fulfillment are given non-zero weights, i.e., $0 < w^s < 1$, the platform's profit decreases by 4.1% relative to the baseline of $w^s = 1$, and the variance of the profit rises by up to 31%, increasing from 1.05×10^5 to 1.38×10^5. In contrast, in the scenario where $w^s = 0$, the platform's average profit significantly drops to \$6,030 from \$14,600 since the fleet size selection focuses solely on optimizing driver utilization, neglecting demand fulfillment.

Table 3 **Platform profit statistics for different weight parameters.**

Weight	Mean ($\times 10^4$)	Variance ($\times 10^5$)	Min ($\times 10^3$)	Max ($\times 10^4$)
0.0	0.603	1.27	0.491	0.685
0.2	1.41	1.02	1.33	1.48
0.5	1.41	1.16	1.33	1.49
0.6	1.41	1.30	1.31	1.49
0.8	1.40	1.38	1.31	1.49
1.0	1.46	1.05	1.36	1.54

;

5.2.3. The Impact of Driver Arrival Probability on the Best Pool Size

In this section, we aim to understand the impact of driver arrival probability on the fleet size selected by Algorithm 1. Recall that we assume that for a given fleet size x_p in period p, the number of drivers that arrive to the platform is a binomially distributed random variable with success

probability q_p. The binomial distribution assumption implies that each driver's joining behavior is an independent Bernoulli trial, thus the binomial random variable is the sum of x_p independent Bernoulli trials each representing the individual driver decision to join the platform on a given day. In the baseline experimental setup discussed in Section 5.1, we set $q_p = 0.7,\ \forall p \in [P]$. In this section, we vary q_p between 0.2 and 1 to understand its impact on the best pool size for varying priority weights. When the entry probability $q_p = 1$, driver arrival is deterministic and mimics a situation where the platform has direct control over driver scheduling or where crowdsourced drivers are perfectly compliant with the scheduling of the platform.

Observe, from Figure 5, that when $0 < w^s < 1$, indicating that both service level and driver utilization target objectives are considered, the aggregate fleet size decreases monotonically as the driver driver arrival probability increases for a fixed priority weight w^s. This pattern is expected; as the platform reduces uncertainty associated with variable driver arrival, fewer drivers are required to meet the desired service level and utilization targets. Interestingly, when q_p is 0.2, the reduction in the aggregate pool size as w^s decreases from 1 to 0.8 is only 7%, which is substantially smaller than the 65% reduction observed when $q_p = 0.7$ for the same weight change. Conversely, when $q_p = 1$, a significant 75% reduction in pool size is achievable as the weight shifts from $w^s = 1$ to 0.8. This demonstrates that as driver arrival behavior becomes more predictable, the platform can achieve its service level and utilization targets with a smaller driver pool. However, when drivers rarely log into the app to provide their services —reflected in a low arrival probability— a larger fleet size is necessary. This underscores the importance of having compliant drivers with limited uncertainty in their arrival behavior to achieve reliable service and high driver utilization.

In Appendix EC.3.1, we evaluate the model's performance under entry probability misspecification, where the actual probability of drivers entering the system deviates from the platform's expectations. This includes the special case of deterministic driver behavior, where the platform assumes perfect adherence to planned work periods. Our results indicate that platform performance is sensitive to variations in driver entry probability: when more drivers enter than anticipated, the platform struggles to maintain high driver utilization; conversely, when fewer drivers enter, the platform's service level (percentage of demand fulfilled) declines. This highlights the critical role of driver adherence to suggested work periods in balancing service level and driver utilization.

5.2.4. The Impact of Varying the Duration of the Scheduling Period

To assess how the length of scheduling periods for fleet size selection affects the proposed solution method and the platform's ability to meet its service level and driver utilization targets, we experimented with varying the number of periods per planning day from 4 to 64. These variations represent period lengths ranging from 4 hours (4 periods per day) to 15 minutes (64 periods per

day). The results of these experiments over 100 test instances are summarized in Table 4. Observe that as the scheduling periods become more granular, both the mean driver utilization and demand fulfillment improve for a given weight of service level, w^s. For instance, when $w^s = 0.5$, increasing the number of periods from 4 to 64 enhances demand fulfillment from 93% to 98% and boosts mean driver utilization from 84% to 95%. Additionally, for a fixed number of periods, we consistently observe that assigning a weight to the driver utilization objective (e.g., reducing w^s from 1 to 0.8) significantly improves expected utilization, while causing only a slight reduction in platform profit and demand fulfillment.

Table 4: Impact of Varying the Number of Scheduling Periods

Metric	Weight (w^s)	4 Periods	8 Periods	16 Periods	32 Periods	64 Periods
Mean demand fulfilled (%)	0	13	24	29	47	85
	0.2	92	92	94	95	96
	0.5	93	97	97	98	98
	0.6	94	96	98	98	99
	0.8	95	97	98	98	99
	1.0	97	100	97	100	100
Mean driver utilization	0	100	100	100	100	98
	0.2	87	95	97	97	96
	0.5	84	90	93	95	95
	0.6	81	89	93	94	94
	0.8	79	91	93	94	92
	1	43	46	37	30	21
Mean fraction drivers meeting utilization target	0	91	91	91	91	87
	0.2	72	88	88	87	83
	0.5	63	79	83	84	81
	0.6	53	75	83	81	79
	0.8	45	82	83	82	75
	1	7	11	13	14	12
Mean driver empty distance (km)	0	1.69	1.77	1.74	1.82	1.91
	0.2	2.18	2.27	2.28	2.20	2.16
	0.5	2.33	2.50	2.52	2.49	2.52
	0.6	2.37	2.44	2.53	2.49	2.65
	0.8	2.45	2.58	2.56	2.75	2.67
	1	2.00	2.18	1.93	1.84	1.61
Mean platform profit ($\$ \times 10^3$)	0	2.86	5.10	6.03	9.14	13.56
	0.2	14.04	13.94	14.13	14.29	14.42
	0.5	13.87	14.02	14.05	14.13	14.11
	0.6	13.92	14.04	14.08	14.16	13.99
	0.8	13.96	13.96	14.03	13.83	14.00
	1	14.60	14.59	14.64	14.99	15.28

5.2.5. The Value of Temporal Fleet Size Adjustments

In this section, we evaluate the benefit of dynamically adjusting fleet sizes over time, comparing this approach to a constant fleet size policy. The constant pool size policy is solved using a modified version of Algorithm 1, where the Boltzmann exploration step is adjusted so that x_p remains constant across all periods $p \in [P]$. To assess the performance of the constant pool size policy against our proposed dynamic policy, which allows for period-specific adjustments in fleet size,

we report the objective value of the best-found solution based on Equation (14), along with the average service level and driver utilization over the test instances.

The results are summarized in Table 5, where the *Objective Value* column reports the evaluation of Equation (14) for the best pool size with the expected service level $Q(\mathbf{x}^*)$ and utilization $L(\mathbf{x}^*)$ computed over the 100 test instances. Observe that, when both driver utilization and service level objectives are considered ($0 < w^s < 1$), the constant pool size policy only achieves both targets when $w^s = 0.8$, attaining a service level of 97.6% and an average driver utilization of 88.8%. However, this comes with an 8.7% increase in the aggregate fleet size compared to the proposed variable fleet size policy. For other priority weights, the algorithm falls short of the 95% target service level, achieving service levels of 88.3% and 92.8% when w^s is set to 0.2 and 0.4, respectively. Intuitively, the service network may become imbalanced at certain decision epochs, depending on the origin-destination characteristics of orders, resulting in drivers being located far from pickup locations. By allowing fleet sizes to be flexibly adjusted, we can mitigate spatial imbalances (for example, by increasing fleet size during specific periods) without substantially increasing the overall fleet size. In contrast, maintaining a constant fleet size forces a trade-off: improvements in driver utilization come at the expense of lower demand fulfillment, or vice versa, since the option to adjust only at specific periods to accommodate spatial imbalance is no longer available.

Table 5: Comparison of Constant and Variable Fleet Sizes

w^s	*Fleet Size Policy*	*Objective Value*	*Mean Demand Fulfilled (%)*	*Mean Driver Utilization (%)*	*Aggregate Fleet Size*
0	Constant	176.0	54.8	100.0	176
	Variable	88.0	29	100	88
0.2	Constant	359.3	88.3	99.0	304
	Variable	357.0	94	97	343
0.5	Constant	405.6	92.8	96.7	336
	Variable	381.1	97	94	376
0.6	Constant	415.8	96	93	368
	Variable	382.2	98	93	378
0.8	Constant	433.6	97.6	88.8	400
	Variable	387.3	97	94	381
1	Constant	2560.0	100.0	19.1	2560
	Variable	1159.9	97	37	1078

We further analyze the fleet size decisions obtained by the simple policy, where fleet size decisions are a function of demand arrival ($x_p = \frac{12\lambda}{\rho}$), as detailed in Benchmark (c) in Section 5.1. The results are report in Table 6. The value of ρ was selected based 100 samples of driver locations and order origin-destination locations. Observe that the obtained fleet size of $x_p = 17$ is inadequate to cover

demand and achieve the desired service level. In this case, the fleet size is not sufficiently large to balance demand fulfillment and driver utilization, resulting in a very high driver utilization of about 99.8%, while demand fulfillment of around 80%. One can vary the values of ρ to determine a pool size that achieves better overall performance. However, doing so will approach the results obtained for the constant pool size policy, discussed in Table 5, which is demonstrated to underperform the dynamically adjusted policy.

5.2.6. Additional Computational Results

We discuss additional computational results in Appendix EC.3, where we discuss the impact of driver arrival probability mis-specification (EC.3.1), the impact of the choice of penalty function (EC.3.2), the computational performance of the VFA algorithm (EC.3.3), comparison of second-stage policies (EC.3.4), convergence analysis of the VFA algorithm (EC.3.5), and applying hierarchical optimization to the edge cases, when $w^s = 0$ or $w^s = 1$ (EC.3.6).

Table 6: Performance Metrics for Simple Fleet Sizing Policy

Metric (Mean Value)	$w^s = 0$	$w^s = 0.2$	$w^s = 0.5$	$w^s = 0.6$	$w^s = 0.8$	$w^s = 1$
Aggregate Fleet Size	272	272	272	272	272	272
Demand Fulfilled (%)	82.7	80.0	79.7	79.3	79.7	79.8
Driver Utilization (%)	99.8	99.8	99.9	99.8	99.8	99.8
Drivers Meeting Utilization Target (%)	99.9	99.9	100	99.9	99.9	99.9
Driver Empty Distance (km)	1.9	2.1	2.2	2.2	2.2	2.2
Platform Profit ($\times 10^3$)	13.5	13.0	12.9	12.9	12.9	12.9
Driver Idle Time (min)	0.15	0.10	0.09	0.1	0.1	0.1
Driver Time in System (min)	60.1	60.1	59.9	59.8	59.8	59.9

5.3. Application to Real-World Problem Instances

In this section, to further evaluate the performance of the proposed VFA, we test it on a real-world Chicago ridehailing dataset, as described in Section 5.1, using trip information from the Chicago area. The objectives of this analysis are twofold. First, we aim to examine a more complex setting, with higher variability in demand throughout the dataset, rather than assuming a fixed arrival rate. Second, the instances in these experiments are larger, enabling us to assess the effectiveness of the proposed algorithms at a greater scale. We summarize the main performance metrics in Table 7. The results exhibit a trend similar to the synthetic instances regarding the tradeoff between service level (demand fulfillment) and driver utilization. As shown in Table 7, when the platform prioritizes service level exclusively ($w^s = 1$), driver utilization becomes highly variable, with some drivers

experiencing low utilization rates, and the mean idle time of drivers accounting for approximately 33% of the time they spend in the system. Conversely, when the platform allocates some weight to meeting driver utilization targets, $0 < w^s < 1$, the fleet size decreases by 59% ($w^s = 0.2$) to 39% ($w^s = 0.5$), leading to a significant improvement in driver utilization. Specifically, the expected driver utilization increases from 66% to 99% and 97% for $w^s = 0.2$ and $w^s = 0.5$, respectively.

A key distinction in the real-world results, however, is that fulfilling some orders is not always profitable. Unlike in the synthetic instances, where higher demand fulfillment generally leads to increased platform profit, certain orders in the real-world case study may incur costs that outweigh the revenue generated. As a result, the platform achieves its highest profit at a lower level of demand fulfillment, specifically when $w^s = 0.2$, with 81% demand fulfillment and a profit of 12.82 $\times 10^3$. This underscores the fact that maximizing service level does not necessarily lead to maximum profitability, as profitability depends on the cost structure of fulfilling orders.

With respect to driver utilization, Figure 6 illustrates the distribution of utilization across 100 test instances based on real-world data, categorized into five utilization brackets. As with the synthetic instances, when the platform prioritizes service level exclusively ($w^s = 1$), driver utilization shows significant variability. Approximately 20% of drivers fall into the lowest utilization bracket (0%-20%), around 55% are in the highest bracket (80%-100%), with the remaining drivers distributed across the intermediate ranges (20%-80%). Conversely, when driver utilization is considered in the platform's objective, regardless of the weight assigned, more than 95% of drivers achieve the target utilization.

Table 7: Performance Metrics of Real-World Instances

Metric	$w^s = 0$	$w^s = 0.2$	$w^s = 0.5$	$w^s = 0.6$	$w^s = 0.8$	$w^s = 1$
Aggregate Fleet Size	234	474	703	698	685	1158
Demand Fulfilled (%)						
Mean	46	81	99	98	97	96
SD	2	1	1	1	1	1
Driver Utilization (%)						
Mean	95	99	97	96	96	66
SD	8	4	6	7	7	39
Drivers Meeting Utilization Target (%)						
Mean	97	100	98	97	96	55
SD	2	1	2	2	3	2
Driver Empty Distance (km)						
Mean	2.53	3.45	5.05	4.90	4.89	4.87
SD	4.03	5.01	6.84	6.59	6.67	6.89

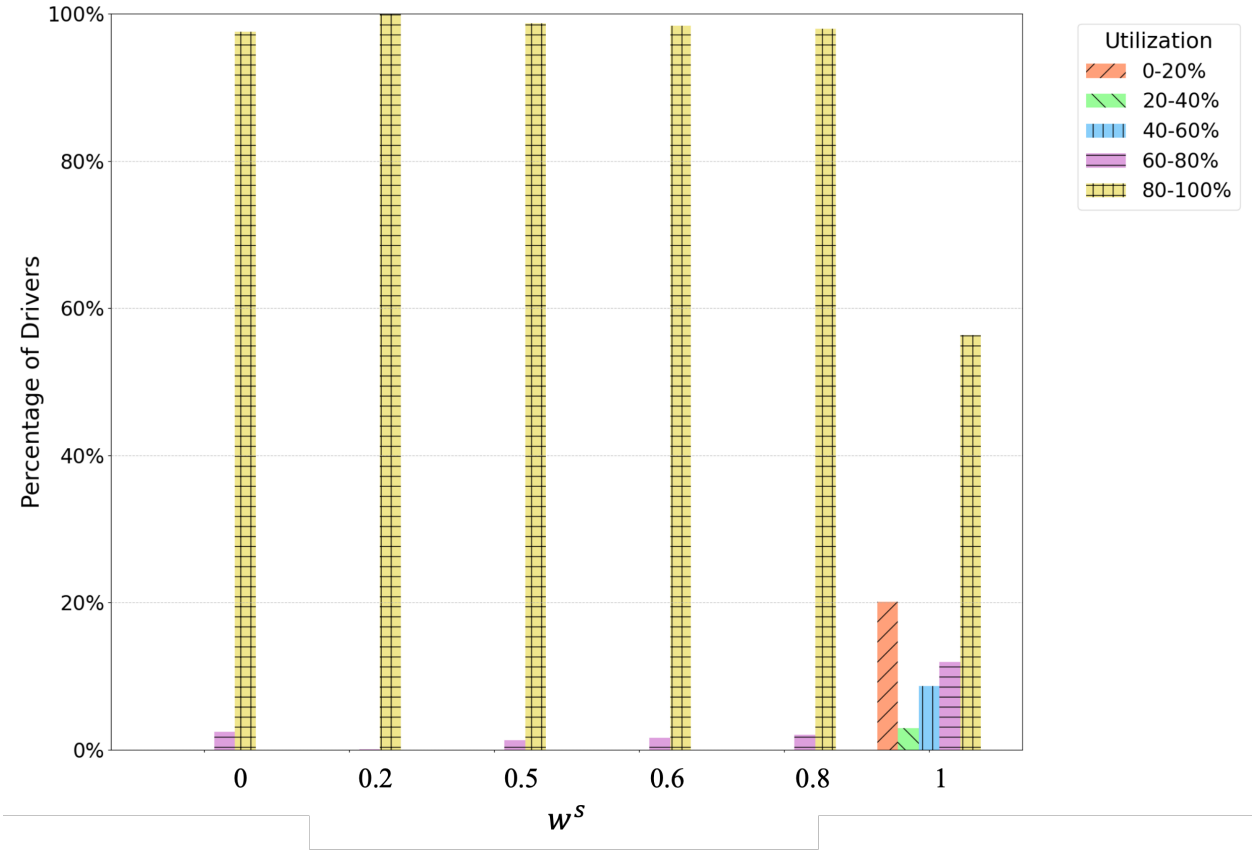

Figure 6 **Distribution of Driver Utilization: Real-World Instances**

Table 7: Performance Metrics (continued)

Metric	$w^s=0$	$w^s=0.2$	$w^s=0.5$	$w^s=0.6$	$w^s=0.8$	$w^s=1$
Platform Profit ($\times 10^3$)						
Mean	10.32	12.82	10.12	10.44	10.54	10.33
SD	0.41	0.36	0.42	0.37	0.43	0.39
Driver Idle Time (min)						
Mean	3.22	0.74	1.92	2.31	2.53	19.94
SD	5.13	2.45	4.14	4.55	4.96	23.96
Driver Time in System (min)						
Mean	60.08	60.05	59.91	60.10	59.85	60.00
SD	17.23	17.35	17.33	17.39	17.33	17.30

5.4. Key Takeaways and Managerial Insights

The computational experiments highlight several practical insights:

1. Incorporating expected driver utilization when determining fleet size can achieve or even exceed the platform's service level target with a significantly smaller fleet of highly-utilized

drivers. This may result in only a marginal reduction in platform profit, as demonstrated in Tables 2, 4, and EC.4, depending on the profitability of the orders being fulfilled.

2. Increased driver adherence to suggested work periods leads to smaller fleet sizes and enhances both driver utilization and overall platform performance, including service levels and daily profit margins. This is evident from the analysis of driver entry probabilities on platform performance (Figure 5) and the impact of misspecified probabilities (Appendix EC.3.1.
3. Utilizing a temporal approach to fleet size planning is more effective for sustaining reliable platform service while maintaining high driver utilization. This can be seen in the results with the analysis on the duration of the scheduling period (Section 5.2.4) as well as comparison to the constant pool size benchmark (Table 5).
4. Maintaining a constant fleet size makes it more challenging to achieve both service level and driver utilization targets. With a fixed fleet size, the platform risks either having an aggregate pool that is too small, leading to unreliable service, or too large, resulting in underutilized drivers. Allowing for flexible adjustments in fleet size over time enables more effective demand fulfillment while minimizing idle drivers. This can be observed in Table 5.
5. Reducing fleet size can increase the distance drivers must travel to reach order pickup locations. While a smaller fleet effectively balances demand fulfillment and high driver utilization, it can lead to longer travel distances due to fewer idle drivers being available in the system. This is illustrated by the increased mean driver empty distance when the service level priority weight is between $0 < w^s < 1$ in Table 2.

6. Concluding Remarks

In this paper, we studied the fleet size planning problem faced by crowdsourced delivery platforms in determining the number of crowdsourced drivers to have during different periods of an operating day to balance the trade-off between achieving high platform service level and maintaining adequate driver utilization. This is motivated by recent regulatory measures targeting gig-economy transportation platforms to improve working conditions for drivers, who often face extended idle periods, potentially earning less than the minimum wage. We develop a two-stage optimization model to determine the optimal fleet sizes considering uncertain driver and order arrival. In the first-stage, representing tactical decisions, the platform determines the number of crowdsourced drivers needed at different periods of the day. In the second stage, the operational problem faced by the platform is modeled by a Markov decision process, where uncertain driver arrival in the MDP is affected by fleet size decisions in the first stage, introducing decision-dependent uncertainty. The platform aims to select the fleet sizes that minimize first-stage fleet size costs and the penalties for deviations from predefined service level and driver utilization targets, as evaluated via the second-stage MDP.

To efficiently solve the proposed model, we utilized a value function approximation solution method that iteratively solves the first and second stage problems separately. At each iteration, a first-stage fleet size solution is determined using Boltzmann exploration, then for a fixed fleet size, the second-stage MDP is approximated using a Monte-Carlo-based cost function approximation method solved in a rolling horizon manner to estimate the platform's expected service level and driver utilization.

We conducted an extensive computational study to assess the performance of the proposed VFA algorithm and to explore how the best fleet sizes vary with different model parameters such as service level priority weights, driver arrival probabilities, the durations of the first-stage periods. The results demonstrate the effectiveness of the VFA algorithm, verify its convergence and its ability to select fleet sizes the meet both performance objectives in expectation. Additionally, the experiments highlights several key insights: platforms can achieve sufficiently high driver utilization without compromising service level; more predictable driver behavior benefits both drivers and the platform, improving profitability and utilization; and temporal fleet size planning is more effective in meeting performance targets by adapting fleet sizes to demand fluctuations.

This research gives rise to several new research directions related to both methodology and applications. First, the proposed mathematical model considered in this paper is a relatively understudied class of problems, namely two-stage optimization problems where the second stage is an MDP with transition probabilities dependent on first-stage decisions. This is a very interesting class of problems that potentially has many applications beyond crowdsourced delivery and transportation. Thus, a natural area of future work is investigating exact and good-quality approximate solution methods for this class of problems. Second, in this work we limited the second-stage decisions to driver-order matching due to the short time windows. An interesting direction for extending this work is to consider order bundling or routing for settings where orders have longer delivery deadlines. Finally, future research could consider spatial fleet management to optimize the geographical distribution of drivers, incorporate order acceptance and rejection probabilities, and examine competition between platforms sharing a common workforce, among other factors.

Acknowledgments

This research is partially funded by a discovery grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) awarded to the first author (grant number RGPIN-2024-04881). We gratefully acknowledge their support.

References

Alnaggar A, Gzara F, Bookbinder J, 2024a *Heatmap design for probabilistic driver repositioning in crowdsourced delivery. Transportation Science* .

Alnaggar A, Gzara F, Bookbinder JH, 2021 *Crowdsourced delivery: A review of platforms and academic literature. Omega* 98:102139.

Alnaggar A, Gzara F, Bookbinder JH, 2024b *Compensation guarantees in crowdsourced delivery: Impact on platform and driver welfare. Omega* 122:102965.

Archetti C, Savelsbergh M, Speranza MG, 2016 *The vehicle routing problem with occasional drivers. European Journal of Operational Research* 254(2):472–480.

Arslan AM, Agatz N, Kroon L, Zuidwijk R, 2019 *Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. Transportation Science* 53(1):222–235.

Auad R, Erera A, Savelsbergh M, 2023 *Courier satisfaction in rapid delivery systems using dynamic operating regions. Omega* 121:102917.

Ausseil R, Pazour JA, Ulmer MW, 2022 *Supplier menus for dynamic matching in peer-to-peer transportation platforms. Transportation Science* 56(5):1304–1326.

Ausseil R, Ulmer MW, Pazour JA, 2024 *Online acceptance probability approximation in peer-to-peer transportation. Omega* 123:102993.

Barbosa M, Pedroso JP, Viana A, 2023 *A data-driven compensation scheme for last-mile delivery with crowdsourcing. Computers & Operations Research* 150:106059.

Basciftci B, Ahmed S, Shen S, 2021 *Distributionally robust facility location problem under decision-dependent stochastic demand. European Journal of Operational Research* 292(2):548–561.

Basık F, Gedik B, Ferhatosmanoğlu H, Wu KL, 2018 *Fair task allocation in crowdsourced delivery. IEEE Transactions on Services Computing* 14(4):1040–1053.

Behrendt A, Savelsbergh M, Wang H, 2023 *A prescriptive machine learning method for courier scheduling on crowdsourced delivery platforms. Transportation Science* 57(4):889–907.

Behrendt A, Savelsbergh M, Wang H, 2024 *Task assignment, pricing, and capacity planning for a hybrid fleet of centralized and decentralized couriers. Transportation Research Part C: Emerging Technologies* 160:104533.

Benjaafar S, Ding JY, Kong G, Taylor T, 2022 *Labor welfare in on-demand service platforms. Manufacturing & Service Operations Management* 24(1):110–124.

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.

Boysen N, Emde S, Schwerdfeger S, 2022 *Crowdshipping by employees of distribution centers: Optimization approaches for matching supply and demand. European Journal of Operational Research* 296(2):539–556.

Casaletto L, Nakhavoly M, 2024 *Uber hit with cap as New York City takes lead in crackdown.* `https://toronto.citynews.ca/2024/06/21/toronto-uber-lyfts-city-streets-cap/`, Accessed: 2024-08-01.

Cerulli M, Archetti C, Fernández E, Ljubić I, 2024 *A bilevel approach for compensation and routing decisions in last-mile delivery. Transportation Science* 58(5):1076–1100.

Chen X, Wang T, Thomas BW, Ulmer MW, 2023 *Same-day delivery with fairness. European Journal of Operational Research* 308(3):738–751.

Chicago Data Protal, 2024 *Transportation network providers - trips.* `https://data.cityofchicago.org/Transportation/Transportation-Network-Providers-Trips/m6dm-c72p`, Accessed: 2024-06-20.

Cleophas C, Cottrill C, Ehmke JF, Tierney K, 2019 *Collaborative urban transportation: Recent advances in theory and practice. European Journal of Operational Research* 273(3):801–816.

Dahle L, Andersson H, Christiansen M, Speranza MG, 2019 *The pickup and delivery problem with time windows and occasional drivers. Computers & Operations Research* 109:122–133.

Dai H, Liu P, 2020 *Workforce planning for o2o delivery systems with crowdsourced drivers. Annals of Operations Research* 291:219–245.

Dayarian I, Savelsbergh M, 2020 *Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. Production and Operations Management* 29(9):2153–2174.

Diao M, Kong H, Zhao J, 2021 *Impacts of transportation network companies on urban mobility. Nature Sustainability* 4(6):494–500.

Fatehi S, Wagner MR, 2022 *Crowdsourcing last-mile deliveries. Manufacturing & Service Operations Management* 24(2):791–809.

Fitzsimmons EG, 2018 *Uber hit with cap as New York City takes lead in crackdown.* `https://www.nytimes.com/2018/08/08/nyregion/uber-vote-city-council-cap.html`, Accessed: 2024-08-01.

Goyal A, Zhang Y, Benjaafar S, 2023 *Crowdsourcing last-mile delivery with hybrid fleets under uncertainties of demand and driver supply: Optimizing profitability and service level. Available at SSRN 4322670* .

Gray J, 2022 *Ontario to legislate $15 minimum wage for gig workers.* `https://centreforfuturework.ca/2022/02/28/dont-be-fooled-by-ontarios-minimum-wage-for-gig-workers/`, Accessed: 2022-04-01.

Horner H, Pazour J, Mitchell JE, 2021 *Optimizing driver menus under stochastic selection behavior for ridesharing and crowdsourced delivery. Transportation Research Part E: Logistics and Transportation Review* 153:102419.

Horner H, Pazour J, Mitchell JE, 2024 *Increasing driver flexibility through personalized menus and incentives in ridesharing and crowdsourced delivery platforms. Naval Research Logistics* 71(1):3–26.

Kaspi M, Raviv T, Ulmer MW, 2022 *Directions for future research on urban mobility and city logistics. Networks* 79(3):253–263.

Le TV, Ukkusuri SV, Xue J, Van Woensel T, 2021 *Designing pricing and compensation schemes by integrating matching and routing models for crowd-shipping systems. Transportation Research Part E: Logistics and Transportation Review* 149:102209.

Lei YM, Jasin S, Wang J, Deng H, Putrevu J, 2020 *Dynamic workforce acquisition for crowdsourced last-mile delivery platforms.* Available at SSRN: `https://ssrn.com/abstract=3532844`, Accessed: 2023-04-01.

Luy J, Hiermann G, Schiffer M, 2024 *Strategic workforce planning in crowdsourced delivery with hybrid driver fleets. Production and Operations Management* 33(11):2177–2200.

Mancini S, Gansterer M, 2022 *Bundle generation for last-mile delivery with occasional drivers. Omega* 108:102582.

Mousavi K, Bodur M, Cevik M, Roorda MJ, 2024 *Approximate dynamic programming for pickup and delivery problem with crowd-shipping. Transportation Research Part B: Methodological* 187:103027.

Mousavi K, Bodur M, Roorda MJ, 2022 *Stochastic last-mile delivery with crowd-shipping and mobile depots. Transportation Science* 56(3):612–630.

Nohadani O, Sharma K, 2018 *Optimization under decision-dependent uncertainty. SIAM Journal on Optimization* 28(2):1773–1795.

Parrott JA, Reich M, Rochford J, Yang X, 2019 *The New York City app-based driver pay standard: Revised estimates for the new pay requirement. New York: Center for New York City Affairs* .

Powell WB, 2011 *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 842 (John Wiley & Sons).

Powell WB, 2019 *A unified framework for stochastic optimization. European Journal of Operational Research* 275(3):795–821.

Powell WB, 2022 *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*, volume 22 (Wiley).

Pugliese LDP, Ferone D, Macrina G, Festa P, Guerriero F, 2023 *The crowd-shipping with penalty cost function and uncertain travel times. Omega* 115:102776.

Qi W, Li L, Liu S, Shen ZJM, 2018 *Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. Manufacturing & Service Operations Management* 20(4):737–751.

Rothenberg E, 2024 *Minnesota lawmakers strike minimum pay deal for uber, lyft drivers.* `https://www.businessinsider.com/uber-lyft-doordash-drivers-earn-below-minimum-wage-tips-study-2024-5`, Accessed: 2024-08-15.

Savelsbergh MW, Ulmer MW, 2022 *Challenges and opportunities in crowdsourced delivery planning and operations. 4OR* 20(1):1–21.

Silva M, Pedroso JP, Viana A, 2023 *Stochastic crowd shipping last-mile delivery with correlated marginals and probabilistic constraints. European Journal of Operational Research* 307(1):249–265.

Simoni MD, Winkenbach M, 2023 *Crowdsourced on-demand food delivery: An order batching and assignment algorithm. Transportation Research Part C: Emerging Technologies* 149:104055.

Stanford J, 2022 *Uber drivers earn less than $8 per hour, says new report from gig worker advocacy group. The Globe and Mail*, `https://www.theglobeandmail.com/business/article-uber-drivers-pay-toronto/`, Accessed: 2024-08-01.

Su E, Qin H, Li J, Pan K, 2023 *An exact algorithm for the pickup and delivery problem with crowdsourced bids and transshipment. Transportation Research Part B: Methodological* 177:102831.

Sutton RS, Barto AG, 2018 *Reinforcement learning: An introduction* (MIT press).

Torres F, Gendreau M, Rei W, 2022a *Crowdshipping: An open vrp variant with stochastic destinations. Transportation Research Part C: Emerging Technologies* 140:103677.

Torres F, Gendreau M, Rei W, 2022b *Vehicle routing with stochastic supply of crowd vehicles and time windows. Transportation Science* 56(3):631–653.

Ulmer MW, Savelsbergh M, 2020 *Workforce scheduling in the era of crowdsourced delivery. Transportation Science* 54(4):1113–1133.

Wang L, Xu M, Qin H, 2023 *Joint optimization of parcel allocation and crowd routing for crowdsourced last-mile delivery. Transportation Research Part B: Methodological* 171:111–135.

Yildiz B, Savelsbergh M, 2019 *Service and capacity planning in crowd-sourced delivery. Transportation Research Part C: Emerging Technologies* 100:177–199.

E-Companion for Fleet Size Planning in Crowdsourced Delivery

EC.1. Alternative Penalty Functions

EC.1.1. Value Function Approximation

For the linear penalty, the value function is computed as follows:

$$v^{(i)}(p, x_p) = \sum_{p' \in [p+1,P]} c_{p'} x_{p'} + (i+1)\left(w^s f^{serv}(\beta, Q(x_{i,p})) + (1-w^s) f^{util}(\mu, L(x_{i,p}))\right) \tag{EC.1}$$

The objective value $v^{(i)}$ is computed as:

$$v^{(i)} = \sum_{p \in [1,P]} \mathbf{c}^\top \mathbf{x}^{(i)} + w^s f^{serv}(\beta, Q(\mathbf{x}^{(i)})) + (1-w^s) f^{util}(\mu, L(\mathbf{x}^{(i)})) \tag{EC.2}$$

For the linear penalty case, the service level and driver utilization penalty functions are defined as follows:

$$f^{util}(\mu, L(\mathbf{x})) = \begin{cases} \zeta^{util}(\mu - L(\mathbf{x})) & \text{if } L(\mathbf{x}) < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.3}$$

$$f^{serv}(\beta, Q(\mathbf{x})) = \begin{cases} \zeta^{serv}(\beta - Q(\mathbf{x})) & \text{if } Q(\mathbf{x}) < \beta \\ 0 & \text{otherwise} \end{cases} \tag{EC.4}$$

For the quadratic penalty case, the service level and driver utilization penalty functions are defined as follows:

$$f^{util}(\mu, L(\mathbf{x})) = \begin{cases} \zeta^{util}(\mu - L(\mathbf{x}))^2 & \text{if } L(\mathbf{x}) < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.5}$$

$$f^{serv}(\beta, Q(\mathbf{x})) = \begin{cases} \zeta^{serv}(\beta - Q(\mathbf{x}))^2 & \text{if } Q(\mathbf{x}) < \beta \\ 0 & \text{otherwise} \end{cases} \tag{EC.6}$$

For the exponential penalty case, the service level and driver utilization penalty functions are defined as follows:

$$f^{util}(\mu, L(\mathbf{x})) = \begin{cases} \zeta^{util}(e^{\kappa(\mu - L(\mathbf{x}))} - 1) & \text{if } L(\mathbf{x}) < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.7}$$

$$f^{serv}(\beta, Q(\mathbf{x})) = \begin{cases} \zeta^{serv}(e^{\kappa(\beta - Q(\mathbf{x}))} - 1) & \text{if } Q(\mathbf{x}) < \beta \\ 0 & \text{otherwise} \end{cases} \tag{EC.8}$$

Here, κ is a parameter controlling the steepness of the exponential function, which we set to 2. ζ^{serv} and ζ^{util} were both set to 250 throughout the experiments.

EC.1.2. Parametric Cost Function Approximation Penalty Functions

The profit of matching driver a with order b under the parametric cost function approximation is given by:

$$\pi_{ab} = r(b) - \overline{c}(a,b) + (1-w^s)g^{util}(l_a) + w^s g^{serv}(t, t_b^{\max})$$

where $g^{util}(l_a)$ is a penalty function that measures deviations from the target utilization, based on a driver's current utilization l_a, and $g^{serv}(\theta)$ is a penalty function that prioritizes the fulfillment of order b as it approaches its delivery deadline $t_b^{\max}$. $r(b)$ is the revenue from order b, $c(a,b)$ is the cost of matching driver a to order b, l_a is the current utilization of driver a, and t_b is the percentage of time window remaining for order b.

For $g^{util}(l_a)$, we have the following formulations based on the penalty type:

- Linear:

$$g^{util}(l_a) = \begin{cases} \zeta(1 - l_a/\mu) & \text{if } l_a < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.9}$$

- Exponential:

$$g^{util}(l_a) = \begin{cases} \zeta(e^{\kappa(\mu - l_a)} - 1) & \text{if } l_a < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.10}$$

- Quadratic:

$$g^{util}(l_a) = \begin{cases} \zeta(\mu - l_a)^2 & \text{if } l_a < \mu \\ 0 & \text{otherwise} \end{cases} \tag{EC.11}$$

where μ is the utilization target and κ is a parameter controlling the steepness of the exponential function.

Defining $\theta = \max\left[1 - \frac{t_b^{\max} - t}{t_b^{\min} - t_b^{\max}},\ 0\right]$ as the fraction remaining time window for order fulfilment, we have the following formulations for g^{serv}:

- Linear:

$$g^{serv}(\theta) = \begin{cases} 0.5\zeta & \text{if } \theta \leq 0.25 \\ 0 & \text{if } \theta > 0.5 \\ \zeta(0.5 - \theta) & \text{otherwise} \end{cases} \tag{EC.12}$$

- Exponential: $g^{serv}(\theta) = \zeta(e^{\kappa\theta} - 1)$
- Quadratic: $g^{serv}(\theta) = \zeta\theta^2$

EC.2. Notation Table

Table EC.1: Notation

Notation	Definition
Sets	
$\mathcal{A}_t$	Set of drivers at time t
$\mathcal{A}_t^{\text{avail}}$	Set of available drivers at time t
$\mathcal{A}_t^{\text{route}}$	Set of en-route drivers at time t
$\mathcal{B}_t$	Set of orders at time t
$\mathcal{B}_t^+$	Set of orders at time t including dummy order 0, i.e., $\mathcal{B}_t^+ \coloneqq \mathcal{B}_t \cup \{0\}$
$\mathcal{D}_{ab}$	Set of time-feasible assignments
X	Set of all possible fleet sizes in a given period, i.e., $X \coloneqq [0, \overline{x}]$
Indices	
p	Index for periods, $p \in [1, P]$
t	Index for decision epochs
a	Index for drivers
b	Index for orders with given attributes
i	Iteration count of the VFA algorithm
Decision Variables	
x_p	Number of drivers the platform accepts in period p
y_{tab}	Binary variable for matching driver a with order b at time t
Parameters	
T	Planning horizon
P	Number of periods
λ	Order arrival rate per decision epoch
q_p	Probability of driver arrival in period p
β	Service level target
μ	Target driver utilization level
w_s	Service level target weight
ι_{ab}	Time required for driver a to fulfill order b
π_{ab}	Profit of matching driver a with order b
$\delta(p)$	Duration of a period p
$\delta(t)$	Duration of a decision epoch t
η	Scalar converting distance to time estimate
γ	Discount factor in MDP
$\overline{x}$	Upper bound on the maximum pool size at a given period
ζ	Penalty constant
τ	Temperature parameter in Boltzmann exploration
ρ	Step size in value function update
$N(x_p)$	The number of times the solution x_p has been encountered in previous iterations of the VFA algorithm
I	Maximum number of iterations
State Variables	
S_t	State of the system at epoch t
R_t	Vector of available drivers at time t
D_t	Vector of orders at time t
K_t	Performance metrics vector at time t
m_a	Binary indicator equals 1 if driver a is available for matching
o_a	driver a's current location

Continued on next page

Table EC.1 – *Continued from previous page*

Notation	Definition
h_a	The number of epochs driver a has spent on-task
l_a	The fraction of time driver a has been utilized
t_a^s	The time driver a entered the system
t_m^s	The time driver a becomes available after completing a delivery
t_e^s	The planned exit time of driver a
D_{tb}	The number of orders with attribute b at time t
(o_b, d_b)	The origin-destination pair of order b
$[t_b^{\min}, t_b^{\max}]$	The earliest and latest allowable delivery times of order b
B_t^{matched}	The number of matched orders up to but not including epoch t
B_t^{matched}	The total number of received orders up to but not including epoch t
A_t^{util}	The average utilization of drivers who were previously available but exited the system before epoch t
A_t^{total}	The total number of drivers that were previously available but exited the system before epoch t
Functions	
$L(\mathbf{x})$	Expected driver utilization given pool size vector $\mathbf{x}$
$Q(\mathbf{x})$	Expected service level given pool size vector $\mathbf{x}$
$f^{\text{serv}}(\cdot)$	Service level penalty function
$f^{\text{util}}(\cdot)$	Driver utilization penalty function
$g^{\text{util}}(\cdot)$	Driver utilization utility function in PCFA
$g^{\text{serv}}(\cdot)$	Service level utility function in PCFA
$\bar{c}(a, b)$	Cost of matching driver a and order b
$r(b)$	Revenue of fulfilling order b
Other Notation	
$\mathbf{x}^{(i)}$	Fleet size vector at iteration i of the VFA algorithm
$\mathbf{x}^*$	Best fleet size vector solution
$v^{(i)}$	Objective value of fleet size vector $\mathbf{x}^{(i)}$ at iteration i
v^*	Objective value of best fleet size vector
d	Value function spread of all possible solutions in a given period
ϵ	Value spread tolerance
$u(k)$	the weight of potential solution k
ω	the sum of the weights of all potential solutions $k \in X$

EC.3. Additional Computational Results

EC.3.1. The Impact of Driver Arrival Probability Mis-specification on Platform Performance

Table EC.2 presents the key performance metrics when the platform selects a fleet size based on the assumption of deterministic driver behavior, i.e., drivers fully adhere to their suggested work periods. When the actual entry probability falls below 1, demand fulfillment decreases in proportion to the actual driver entry probability. For example, with an entry probability of 0.8, demand fulfillment drops from 98% to 86%. This decline becomes more severe as p decreases further; at $p = 0.5$, demand fulfillment drops to 56%, and platform profit decreases from 14.57×10^3 to 10.33×10^3.

Table EC.2 Performance Metrics for $w^s = 0.5$ Assuming Deterministic Driver Behavior ($p = 1$) During VFA Training

Metric	$p = 0.2$	$p = 0.4$	$p = 0.5$	$p = 0.7$	$p = 0.8$	$p = 1$
Demand Fulfilled (%)						
Mean	24	45	56	77	86	98
SD	3	3	3	3	3	1
Driver Utilization (%)						
Mean	100	100	100	100	100	94
SD	0	0	0	1	2	8
Drivers Meeting Utilization Target (%)						
Mean	91	91	91	91	91	86
SD	29	29	29	29	29	27
Driver Empty Distance (km)						
Mean	1.88	2.04	2.09	2.16	2.24	2.28
SD	1.32	1.48	1.54	1.66	1.73	1.86
Platform Profit ($\$ \times 10^3$)						
Mean	5.18	8.82	10.33	12.73	13.55	14.57
SD	0.63	0.53	0.46	0.34	0.32	0.38
Driver Idle Time (min)						
Mean	0.00	0.00	0.00	0.03	0.28	0.48
SD	0.00	0.00	0.00	0.46	1.63	2.34
Driver Time in System (min)						
Mean	65.45	65.04	65.05	65.56	65.85	65.56
SD	17.34	17.46	17.53	17.79	17.93	17.99

Table EC.3 presents the performance results for another example of probability mis-specification. In this case, the platform assumes a driver entry probability of 0.7, but the actual probability during testing ranges from 0.2 to 1. When the actual probability is higher, as in the case of $p = 1$, more drivers are available, leading to a decrease in driver utilization and an increase in idle time. Conversely, when fewer drivers arrive, such as at $p = 0.5$, the platform experiences a significant drop in demand fulfillment.

EC.3.2. The impact of the choice of objective function and penalty function on the VFA algorithm performance

In Model (1), the objective function consists of two components: a fleet size cost, $\mathbf{c}^\top \mathbf{x}$, and a total deviation penalty, α, which accounts for a weighted sum of the platform's service level and driver utilization penalties. This section explores the impact of these functional choices on the optimal fleet size and overall platform performance.

In our base model, we define $\mathbf{c}^\top \mathbf{x} = \sum_{p \in [P]} x_p$, implying that the platform seeks to determine the smallest pool size that meets the service level and utilization targets in expectation. However, in crowdsourced delivery platforms, the cost of adding drivers to the pool is typically negligible, as drivers are not employees and are compensated only on a per-task basis. Therefore, we examine the

Table EC.3 Performance Metrics under Mis-specified Entry Probability with $w^s = 0.5$

Metric	$p = 0.2$	$p = 0.4$	$p = 0.5$	$p = 0.7$	$p = 0.8$	$p = 1$
Demand Fulfilled (%)						
Mean	33	64	77	97	100	100
SD	4	4	4	2	0	0
Driver Utilization (%)						
Mean	100	100	100	94	86	67
SD	0	0	2	10	13	13
Drivers Meeting Utilization Target (%)						
Mean	91	91	91	82	63	15
SD	29	29	29	27	22	6
Driver Empty Distance (km)						
Mean	1.95	2.12	2.19	2.51	2.63	2.47
SD	1.37	1.58	1.69	1.95	2.03	1.88
Platform Profit ($\$ \times 10^3$)						
Mean	6.84	11.29	12.75	14.07	14.06	14.30
SD	0.63	0.51	0.43	0.34	0.34	0.33
Driver Idle Time (min)						
Mean	0.00	0.00	0.13	4.39	9.69	21.54
SD	0.00	0.00	1.12	6.90	9.16	10.63
Driver Time in System (min)						
Mean	64.50	65.32	65.71	65.93	65.87	65.31
SD	17.31	17.70	17.85	17.86	17.77	17.90

effect of excluding the pool size term $\mathbf{c}^\top \mathbf{x}$ from model (1) and value function estimates (14)-(15) on the optimal pool size and the quality of driver-order matches at the operational level.

Table EC.4 summarizes the results of this analysis. Compared to the scenario where we directly minimize the fleet size (Table 2), the fleet size increases by 6% (when $w^s = 0.5$) to 89.51% (when $w^s = 1$). As a consequence of this increase, the expected demand fulfillment improves by 1% to 4%, while the expected driver utilization decreases by 2% to 15%. We note however, that for all weights in the range $0 < w^s < 1$, both the service level and utilization targets are achieved in expectation. In this cases, larger pool sizes may be chosen to enhance platform profit. For instance, when $w^s = 0.2$, the platform's profit rises to 14.4×10^4, compared to 14.1×10^4 when the sum of fleet sizes over the planning periods is minimized directly.

Table EC.4: Performance Metrics with No Fleet Selection Cost

Metric	$w^s = 0$	$w^s = 0.2$	$w^s = 0.5$	$w^s = 0.6$	$w^s = 0.8$	$w^s = 1$
Aggregate Fleet Size	168	378	399	407	414	2043
Demand Fulfilled (%)						
Mean	52	98	98	99	99	100
SD	3	1	1	1	1	0
Driver Utilization (%)						
Mean	99	92	91	89	89	22
SD	5	10	11	12	11	27
Drivers Meeting Utilization Target (%)						

Table EC.4: Performance Metrics with Pool Summation Term Excluded (continued)

Metric	$w^s=0$	$w^s=0.2$	$w^s=0.5$	$w^s=0.6$	$w^s=0.8$	$w^s=1$
Mean	90	81	77	72	74	7
SD	29	26	26	24	25	2
Driver Empty Distance (km)						
Mean	1.91	2.31	2.58	2.62	2.71	1.65
SD	1.31	1.72	1.99	2.04	2.16	2.36
Platform Profit ($\$ \times 10^3$)						
Mean	9.29	14.41	14.04	14.05	13.92	15.24
SD	0.40	0.34	0.33	0.36	0.38	0.35
Driver Idle Time (min)						
Mean	0.95	5.61	6.33	7.33	7.71	49.60
SD	3.37	7.12	7.82	8.35	8.20	22.18
Driver Time in System (min)						
Mean	65.45	65.56	66.05	65.91	66.00	64.07
SD	17.76	17.66	17.81	17.92	17.97	18.21

Next, we explore the impact of different forms of the penalty functions f^{serv} and f^{util} used in model (1), as well as the utility functions g^{serv} and g^{util} employed in the parametric cost function approximations (20) and (21), on both the best pool size identified by the VFA Algorithm 1 and the platform's operational performance. In our baseline model, these penalty and utility functions were assumed to be piecewise linear. We propose alternative quadratic and exponential functions in EC.3.2 and re-solve Algorithm 1 then evaluate its performance on the same 100 test instances.

The results of this analysis, shown in Table EC.5, indicate that proposed exponential penalty functions yield the highest demand fulfillment across all priority weights in the range $0 < w^s < 1$, with improvements ranging from 2% to 6% compared to linear penalties. However, this comes at the cost of a reduction in driver utilization, which decreases by 3% to 30% relative to the linear case. Observe that with exponential functions, the driver utilization target is not achieved for any weights between $0 < w^s < 1$. In contrast, quadratic penalties offer a more balanced performance: the service level target is unmet only when $w^s = 0.2$, and driver utilization falls short when $w^s = 0.8$. Overall, piecewise linear penalty functions exhibit the most balanced outcomes in terms of meeting or exceeding service level and driver utilization targets, in expectation, for various priority weights.

EC.3.3. Computational performance

In this section, we discuss the computational performance of the proposed VFA algorithm, as summarized in Table EC.6. Columns 2 to 5 report, respectively: (i) the average computational time required to solve an instance of Model (18) for a given scenario and decision epoch; (ii) the average computational time of Algorithm 3; (iii) the average time for the Boltzmann Exploration step (Algorithm 2); and (iv) the average iteration time for one iteration of Algorithm 1. All values are reported in seconds.

Table EC.5 Comparative Analysis of Penalty Functions across Different Metrics

Metric	Penalty Function	Weight (w^s)					
		0	**0.2**	**0.5**	**0.6**	**0.8**	**1**
Aggregate Fleet Size	Linear	88	343	376	378	381	1078
	Quadratic	134	276	399	415	453	1239
	Exponential	106	512	522	538	529	1340
Mean Demand Fulfillment (%)	Linear	29	94	97	98	98	97
	Quadratic	42	80	95	98	94	97
	Exponential	34	100	100	100	100	100
Mean Driver Utilization (%)	Linear	100	97	93	93	93	37
	Quadratic	100	97	86	84	74	30
	Exponential	100	67	70	68	71	31
Mean Empty Travel Distance (km)	Linear	1.74	2.28	2.52	2.53	2.56	1.93
	Quadratic	1.80	1.98	2.34	2.36	2.40	1.51
	Exponential	1.76	2.14	2.55	2.56	2.71	1.71
Mean Driver Idle Time (min)	Linear	0.00	2.25	4.51	4.48	4.48	40.42
	Quadratic	0.09	1.74	9.35	10.36	16.86	44.66
	Exponential	0.00	21.16	19.58	20.96	19.17	44.56
Mean Platform Profit ($\$ \times 10^3$)	Linear	6.03	14.13	14.05	14.08	14.03	14.64
	Quadratic	8.24	12.99	14.10	14.32	13.79	15.09
	Exponential	7.01	14.74	14.19	14.16	13.98	15.18

Observe that at a given iteration of the VFA algorithm, the majority of the computational time is spent in the PCFA step (the evaluation of a given $\mathbf{x}^i$ through Algorithm 3). Although a single run of Model (18) requires approximately 0.02 seconds, Algorithm 3 solves $T \times |\Xi|$ instances (a total of 1920 runs) in one call. As a result, the average duration of each call ranges from 36 to 75 seconds.

Table EC.6 Computational Time Analysis for Different Weights

Weight (w^s)	Avg. Optimization Time per Decision Epoch (s)	Avg. PCFA Time per Iteration (s)	Avg. Boltzmann Exploration Time per Iteration (s)	Avg. Iteration Time (s)
0.0	0.0247	47.3462	0.0035	47.3656
0.2	0.0219	42.0920	0.0035	42.1159
0.5	0.0195	37.3462	0.0032	37.3675
0.6	0.0188	36.1210	0.0031	36.1720
0.8	0.0228	43.7217	0.0033	43.7494
1.0	0.0392	75.3364	0.0037	75.3361

EC.3.4. The Impact of the Second-Stage Policy on the Quality of Fleet Size Decisions

In this section, we assess the sensitivity of Algorithm 1 to the choice of second-stage solution method. Specifically, we replace the CFA algorithm in Line 9 of Algorithm 1 with a greedy matching policy as discussed in Benchmark (d) in Section 5.1. The results of the greedy matching policy are reported in Table EC.7. Observe from the results that the platform is able to reach its service level and utilization targets when this heuristic method is used to solve the operational problem. However, the resulting platform profit deteriorates. For instance, when $w^s = 0.5$, the platform daily

profit reduces to 13.59×10^3 when the greedy matching policy is used, whereas the platform daily profit is 14.1×10^3 (as reported in Table 2). This demonstrates that the type of policy in the second stage affects the profitability of the matches.

We further evaluated the performance of using the myopic policy to solve the second stage, as discussed in Benchmark (e) in Section 5.1. We found that the resulting fleet sizes and performance metrics are highly comparable, and thus these results have been omitted to avoid redundancy. However, the primary difference lies in the convergence behavior of the VFA algorithm; the myopic policy exhibits greater variability between iterations compared to the PCFA policy, as shown in Figure EC.1. Additional analysis on the convergence of the VFA is provided in Appendix EC.3.5.

Table EC.7: Performance Metrics for Greedy Matching Policy

Metric	$w^s=0$	$w^s=0.2$	$w^s=0.5$	$w^s=0.6$	$w^s=0.8$	$w^s=1$
Aggregate Fleet Size	104	325	366	376	392	1035
Demand Fulfilled (%)						
Mean	37.7	98.6	99.7	99.6	99.8	99.6
SD	02.9	0.9	0.3	0.5	0.3	0.5
Driver Utilization (%)						
Mean	100	95.9	87.8	90.9	88.8	42.4
SD	0	7.8	13.1	11.4	13.9	32.8
Drivers Meeting Utilization Target (%)						
Mean	100	96.0	77.9	86.7	80.1	21.0
SD	0	19.7	41.5	33.9	39.9	40.8
Driver Empty Distance (km)						
Mean	2.0	2.8	3.2	3.1	3.1	3.5
SD	1.2	1.9	2.1	2.0	2.0	2.2
Platform Profit ($\times10^3$)						
Mean	6.17	13.86	13.59	13.66	13.56	13.06
SD	0.45	0.43	0.41	0.45	0.50	0.35
Driver Idle Time (min)						
Mean	0	2.467	7.281	5.448	6.659	34.521
SD	0	4.67	7.87	6.79	8.35	21.70
Driver Time in System (min)						
Mean	59.9	59.9	60.2	60.0	59.9	59.9
SD	17.4	17.3	17.3	17.3	17.3	17.3

EC.3.5. Convergence analysis

In this section, we discuss the convergence of the service level and the driver utilization metrics over the iterations of the VFA algorithm, as w^s and the number of sample paths $|\Xi|$ vary.

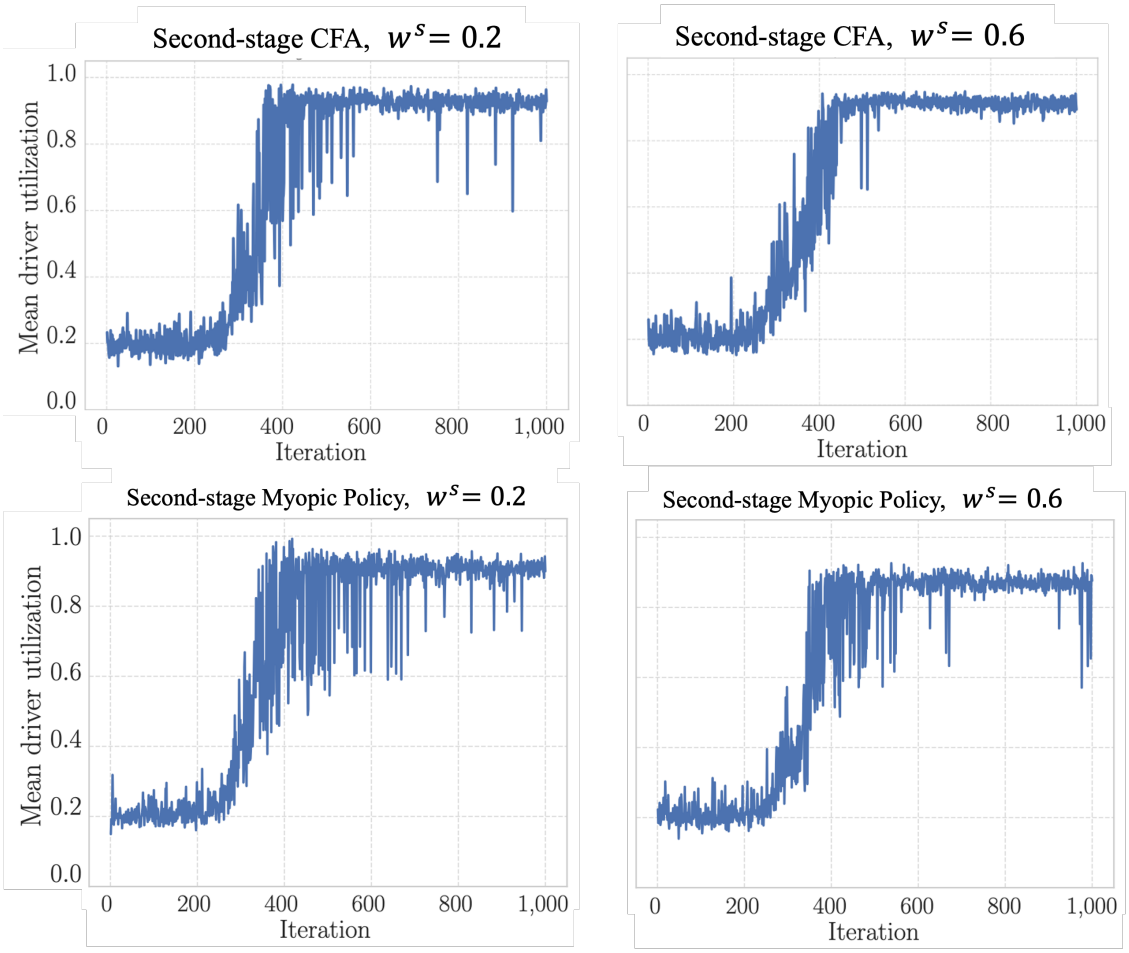

Figure EC.1 Comparison of the convergence of driver utilization when using CFA vs. a myopic policy

EC.3.5.1. Service level convergence

Figure EC.2 illustrates the convergence of the expected service level across different values of w^s. Observe that the algorithm converges rapidly for all values of w^s, consistently surpassing the service level target. The exception is when $w^s = 0$, where the algorithm converges more slowly, stabilizing at approximately 50% demand fulfillment, but with high variability.

To investigate the algorithm's sensitivity to the number of sample paths used in the cost function approximation, we conducted experiments with varying numbers of sample paths. Figures EC.3 and EC.4 depict the convergence of expected service level for 1 and 25 sample paths, respectively. As shown in Figure EC.6, a reduced number of sample paths leads to greater variability in some weights. However, when increasing from the base case of 10 sample paths (Figure EC.2) to 25 paths (Figure EC.4), the impact on demand fulfillment variability is minimal, highlighting the diminishing returns of adding more sample paths.

EC.3.5.2. Driver utilization convergence

Figure EC.5 shows the convergence of driver utilization across different values of w^s over the

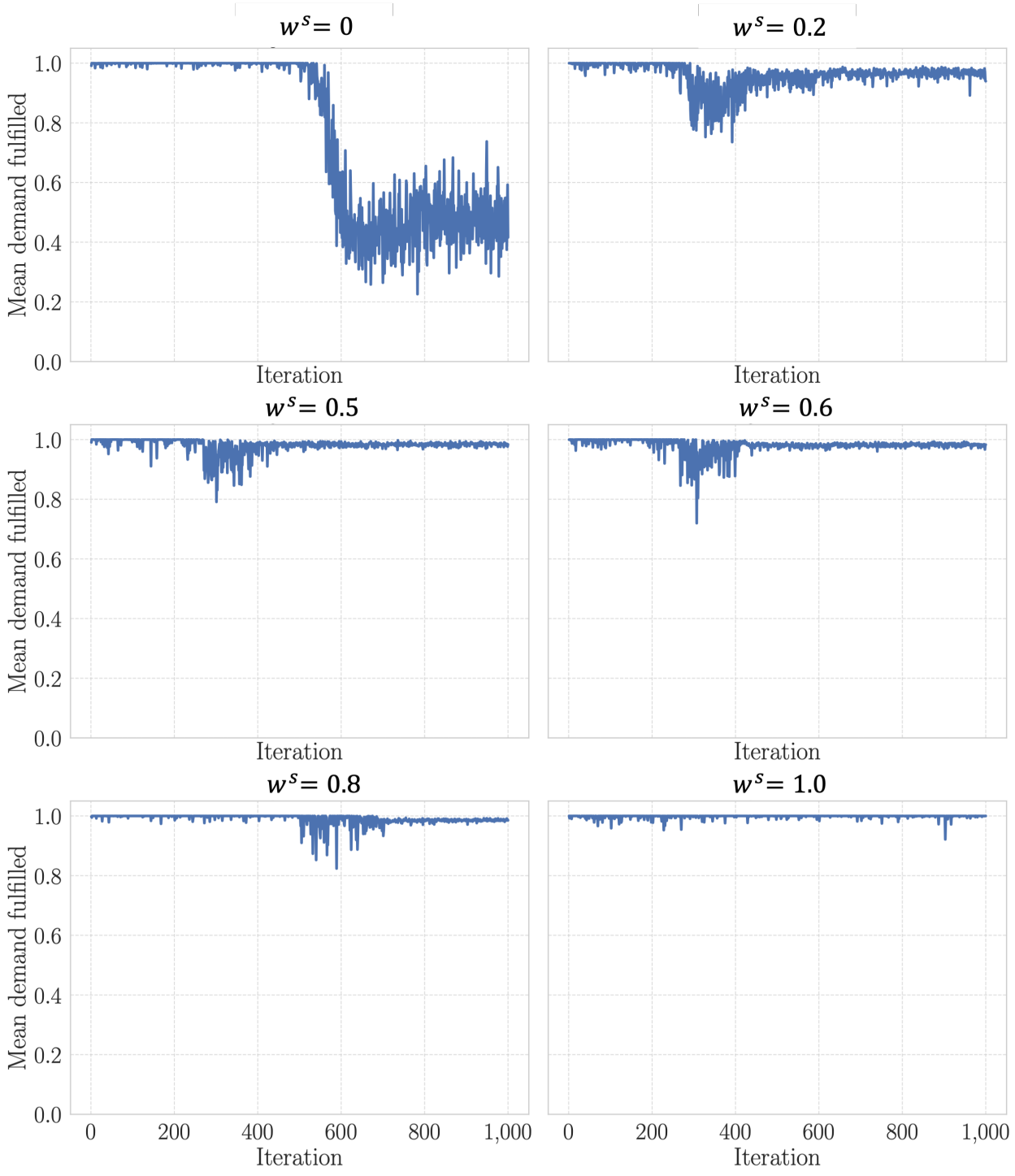

Figure EC.2 Convergence of expected service level for different weight parameter values

iterations of the VFA algorithm. Driver utilization exhibits a slower convergence compared to service level, with a sharp increase in utilization occurring around the 400th iteration. In the earlier iterations, the utilization hovers around 20%, suggesting that the algorithm initially selects a large

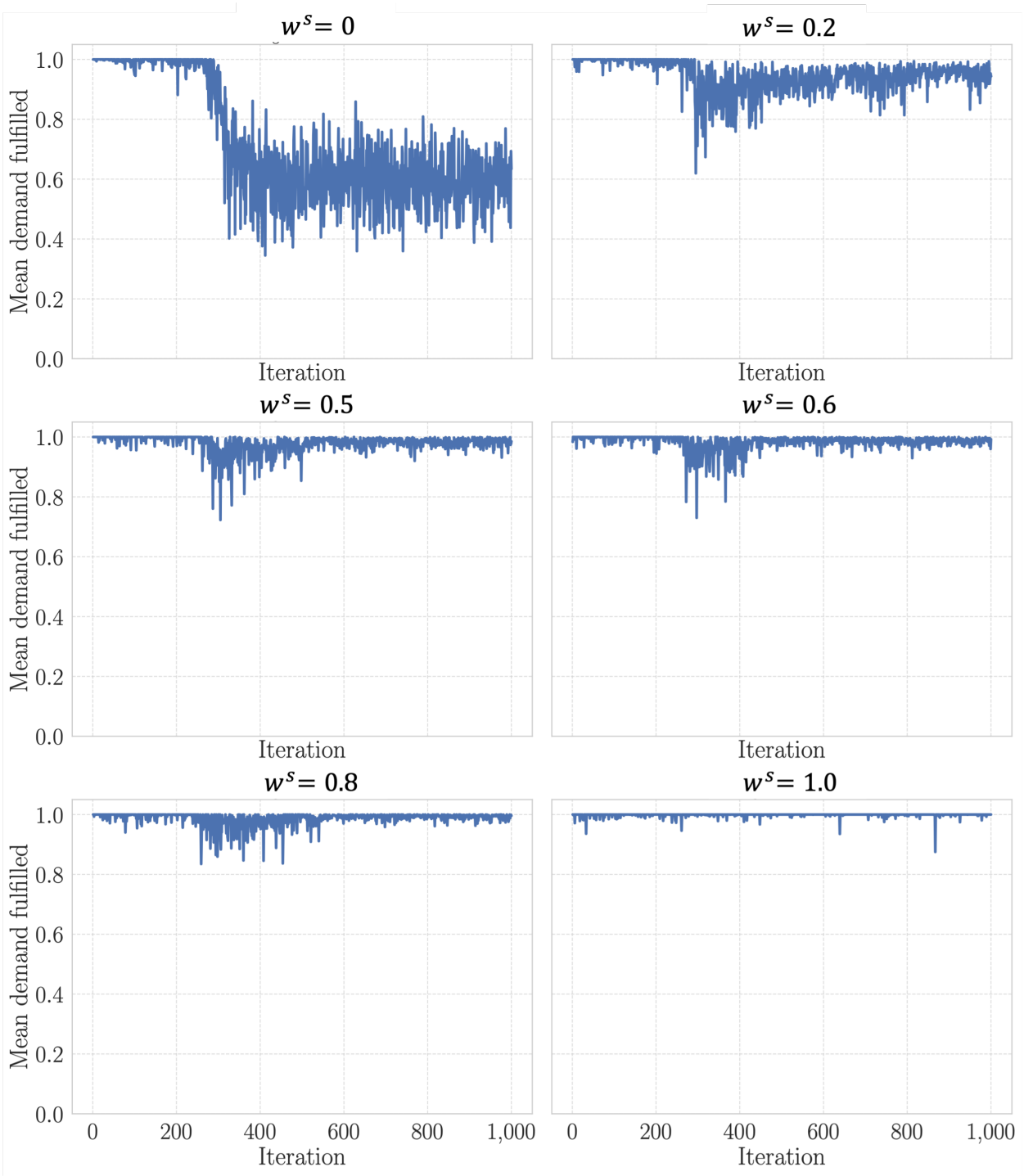

Figure EC.3 Impact of sample paths on expected service level (1 sample path)

pool size to prioritize the service level. As iterations progress, the algorithm adjusts to account for driver utilization, eventually achieving convergence.

Similar to the behavior observed in service level convergence, the variance in driver utilization is significant when using a single sample path, as illustrated in Figure EC.6. This variability

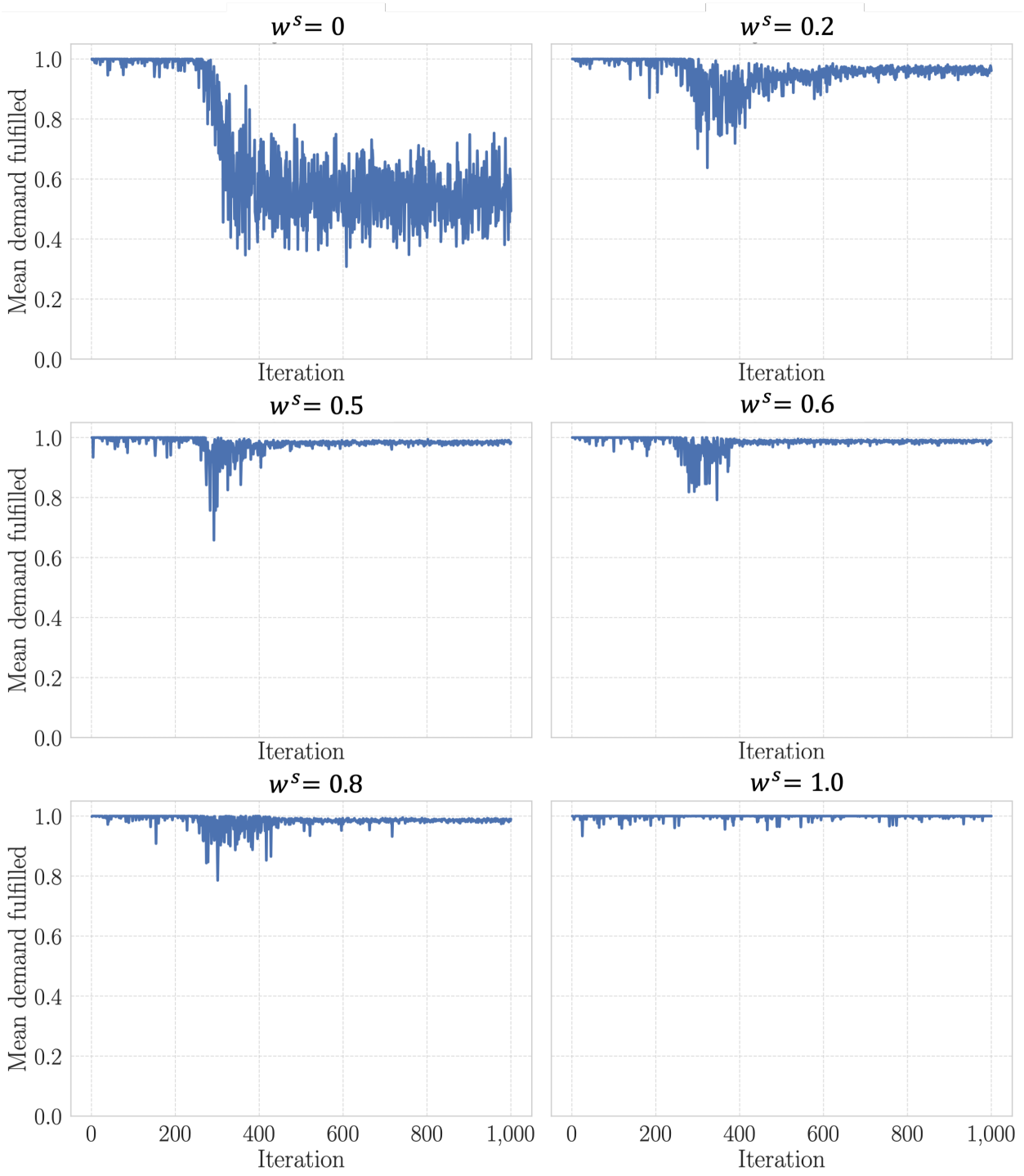

Figure EC.4 Impact of sample paths on expected service level (25 sample paths)

decreases substantially when the number of sample paths is increased to 10, as seen in Figure EC.5. However, further increasing the number of sample paths to 25 (Figure EC.7) leads to only marginal improvements in reducing variability compared to the baseline of 10 sample paths, underscoring the limited benefits of additional sample paths beyond a certain point.

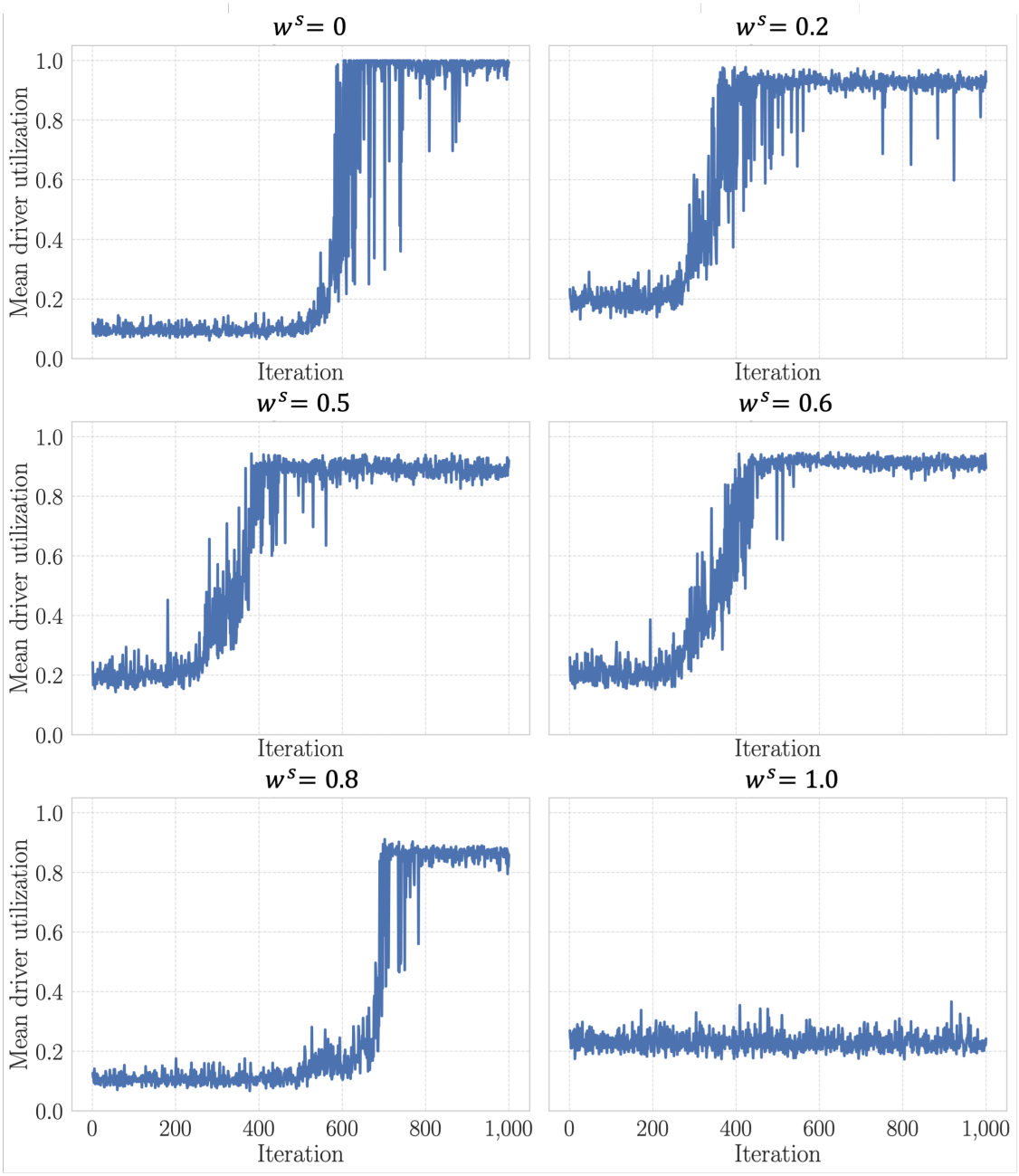

Figure EC.5 Convergence of expected driver utilization

EC.3.6. Performance of the VFA algorithm with hierarchical optimization applied to edge cases

In this section, we examine the effect of applying hierarchical optimization to the edge cases where $w^s = 0$ or $w^s = 1$, such that only one objective is considered. Specifically, in this modification, Line 10 of Algorithm 1 is modified such that if the objective value at iteration iteration i, $v^{(i)}$, satisfies $v^{(i)} \leq v^*$ (i.e., we now consider the equality condition), then we check if the alternative objective

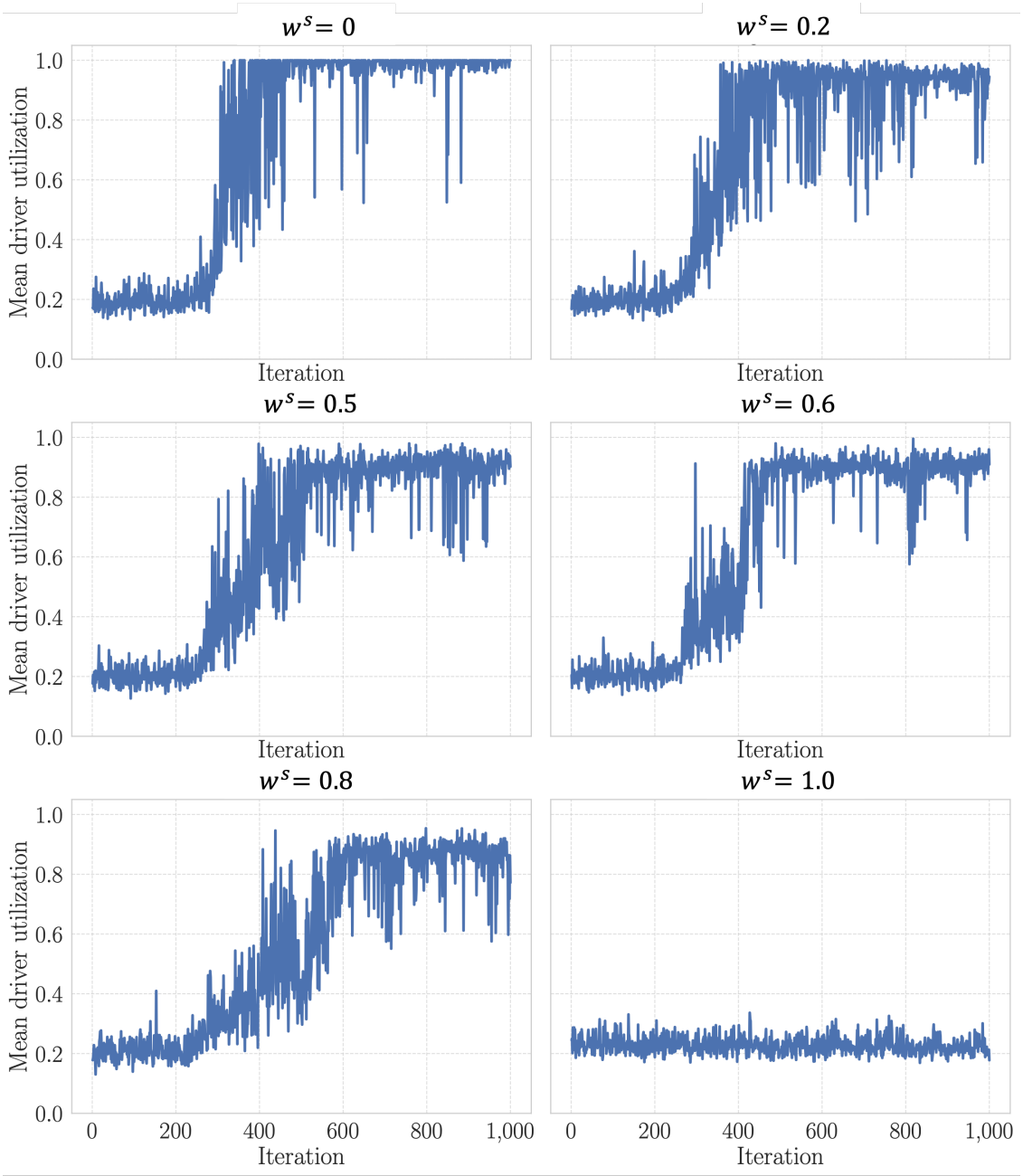

Figure EC.6 Impact of sample paths on expected driver utilization (1 sample paths)

yields a better value. For instance, when $w^s = 0$, we check if

$$Q(\mathbf{x}^{(i)}) > Q(\mathbf{x}^*).$$

If this condition is met, we update the best known solution by setting v^* and $\mathbf{x}^*$ to the values obtained in iteration i.

The results are shown in Table EC.8. Observe that the results are very close to those without hierarchical optimization, as reported in Table 2. This is because in evaluating the objective value

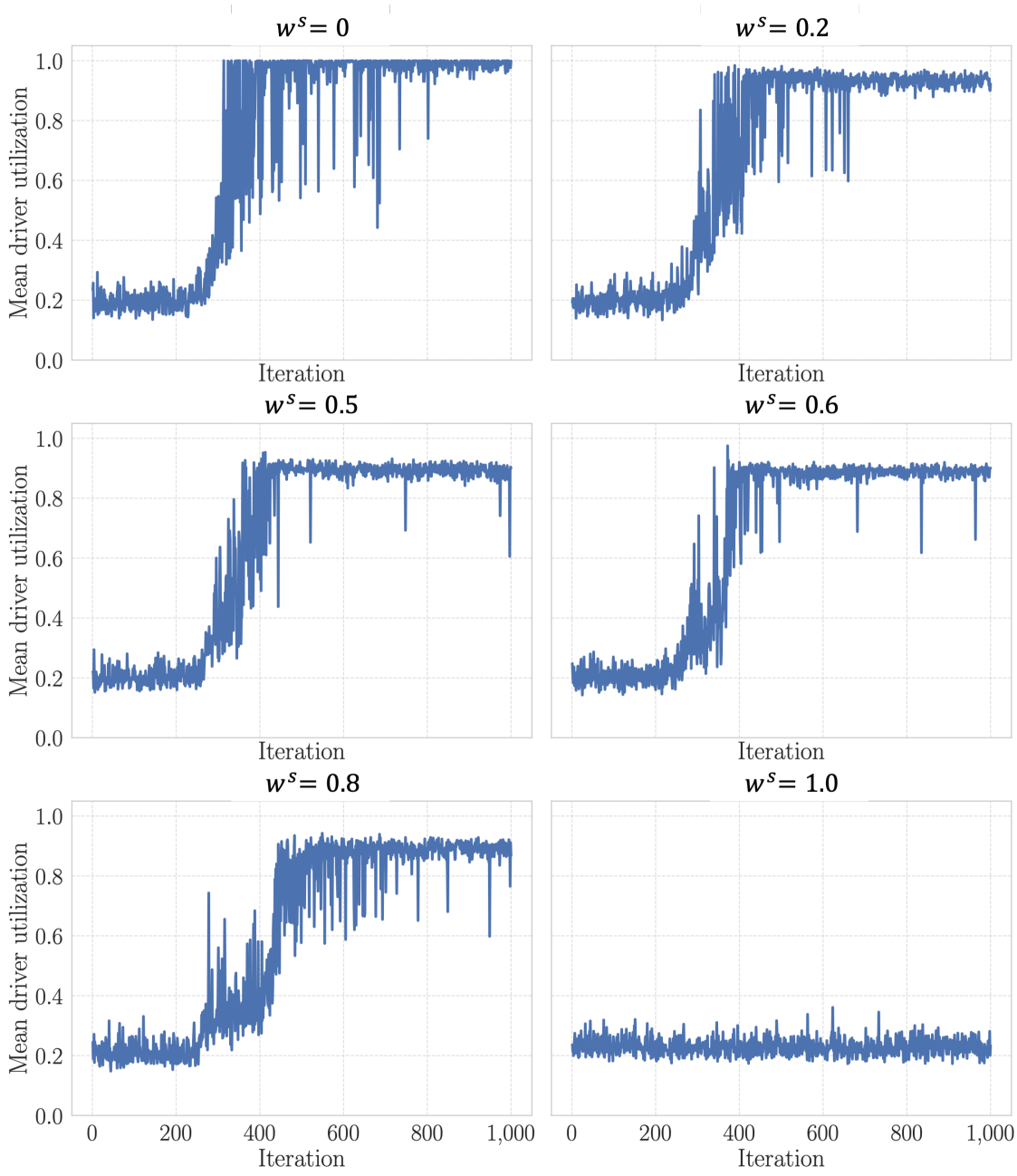

Figure EC.7 Impact of sample paths on expected driver utilization (25 sample paths)

of a solution $\mathbf{x}^{(i)}$ in Algorithm 3—as computed by Equation (14)—the penalty term is applied for each period p. Thus, to avoid penalties in every period, only a very limited set of fleet size decisions can improve the non-prioritized objective without degrading the other components of Equation (14).

Table EC.8: Performance Metrics: Hierarchical Optimization for Edge Cases

Metric	$w^s = 0$	$w^s = 1$
Aggregate Fleet Size	89	1200
Demand Fulfilled (%)		
Mean	28	100
SD	2	0
Driver Utilization (%)		
Mean	100	36
SD	0	30
Drivers Meeting Utilization Target (%)		
Mean	91	11
SD	29	4
Driver Empty Distance (km)		
Mean	1.89	2.09
SD	1.25	2.37
Platform Profit ($\$ \times 10^3$)		
Mean	5.88	14.72
SD	0.37	0.35
Driver Idle Time (min)		
Mean	0.00	38.50
SD	0.20	21.08
Driver Time in System (min)		
Mean	59.77	59.81
SD	17.23	17.34