UNIVERSITA' DELLA CALABRIA

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in

Matematica e Informatica

Con il contributo di

POR Calabria FESR-FSE 2014/2020

CICLO

XXXV

Crowdshipping in Dynamic Pickup-and-Delivery Problems

Settore Scientifico Disciplinare MAT/09

Coordinatore: Ch.mo Prof. Giorgio Terracina

Firma _____ Terracina Giorgio

Supervisore: Ch.mo Prof. Demetrio Laganà

Firma_____ Demetrio Laganà 07.11.2023 18:02:19 GMT+00:00

Supervisore: Ch.ma Prof.ssa Francesca Vocaturo

Firma_____ Francesca Vocaturo 07.11.2023 18:09:41 GMT+00:00

Dottoranda: Dott.ssa Sara Stoia

Firma SARA STOIA 07.11.2023 19:29:24 UTC

“La borsa di dottorato è stata cofinanziata con risorse del Programma Operativo Regionale Calabria FSE/FESR 2014 – 2020 (CCI 2014IT16M2OP006)”

Contents

Introduction

In recent years, several significant changes in distribution logistics have been observed driven mainly by technological innovations, growing environmental awareness and global challenges such as the COVID-19 pandemic. The rapid technological development that has taken place in recent years has inevitably had a strong impact in distribution logistics as well as in other sectors. Driven by the development of e-commerce allowing consumers to buy practically anything, any time of the day, and anywhere in the world, the volume of packages delivered have been growing steadily, reaching staggering numbers (50 million packages shipped in one day in the United States alone as reported in an article written by SAP company). The Covid-19 pandemic certainly provided a boost to purchases on online platforms, in which due to restrictions, businesses were closed and user mobility restricted. While today the pandemic may seem to be just a bad memory, consumer habits have remained changed from the past. The number of online sales even though they seem to have stabilized, have higher values than in the pre-Covid era.

In the past, brick-and-mortar retail maintained an advantage of immediacy and ability to physically inspect products that online purchasing could not provide. However, online shoppers now have come to expect fast and inexpensive delivery paired with a convenient return policy in the case a product is not what they expect. The time factor has become one of the determining elements also in the choice of the supplier by the customer, in a highly competitive world. A survey conducted in 2020 by BusineesWire (2021) showed

that 65% of consumers are even willing to pay a little more to receive their products sooner. As a direct consequence of the increasing customer demands, companies that manage last mile logistics - the transportation of merchandise from its last storage location to the final destination - must manage the delivery capacity at their disposal in the best possible way. Technological development is also a valuable tool in last mile delivery, as it is commonplace to have access to real-time data which can be leveraged to improve decisions.

Alongside the management of increasing volume of packages, another important issue that has been widely discussed in recent years is the concept of sustainability, both from an environmental, social and economic point of view. A direct consequence of the increase in the number of packages to be transported is the increase in CO_2 emissions which represents a threat to the environment and a danger to the health of citizens. In large cities, some areas are off-limits to traditional vehicles in an effort to curb emissions complicating deliveries relying on conventional means of transport. This has led to an emphasis on electric micro-mobility in the form of e-bikes, cargo bikes, scooters, ground drones etc. These vehicles are certainly useful especially if they also allow the use of renewable energy sources to be recharged, but at the moment they suffer from some limitations related to the duration of the batteries, the capacity they can carry, the infrastructure of the charging network. To solve these limitations, investments in both infrastructure and research and development of ever more performing means are needed.

Another strategy used in last mile delivery uses the principles from the sharing economy. In a sharing economy, it is no longer essential to own all resources on needs as long as one has convenient access to them through a resource-sharing arrangement. The sharing economy promotes the efficient use of existing resources, thereby reducing resource redundancy and reusing existing goods and services, ultimately contributing to greater environmental sustainability. This model has gained importance in recent years as a further

consequence of the development of technology. The exchange of information regarding the resources to be shared takes place through the use of digital platforms and apps that connect providers (those sharing their goods or services) with consumers (those looking to use them). These platforms provide an efficient mechanism for searching, booking, and paying for shared services.

The earliest forms of sharing economy were car-sharing, bike-sharing, and home-sharing. In the world of logistics, the sharing economy has emerged as crowdshipping, also known as crowdsourced delivery. In this paradigm, individuals often referred to as "crowdshippers" use their own vehicles and time to deliver packages or goods to others. It's a decentralized and community-driven approach to shipping and logistics. Unlike traditional courier or delivery services, crowdshipping relies on a network of non-employees who are willing to transport goods. This approach can potentially provide more flexible and localized delivery options.

By reducing the number of dedicated delivery vehicles needed, crowdsourced delivery can reduce the number of delivery vehicles on the road. That is, instead of having dedicated vehicles to make all the deliveries, vehicles ostensibly already on the road are used. This certainly has an impact in terms of environmental sustainability, but it is also a strategy to reduce the costs of delivery companies as each dedicated vehicle removed from its fleet represents a reduction in operating costs. Instead, a delivery company can provide piece-rate pay for crowdshippers. However, the delivery company must be still be able to meet customer demand and ensure high levels of customer service, requiring the efficient management of a sufficiently large crowdshipper work force.

While crowdshipping has many beneficial aspects, there are critical issues that limit its applicability as the only delivery strategy of companies. The first critical issue is related to reliability. Having to manage drivers that are independent and have no contract can be risky, as the capacity needed to meet demand in a specific time period may not be guaranteed. It is difficult to

predict in advance the behavior of crowdshippers and this choice can be less consistent compared to traditional services, particularly in areas with fewer crowdshippers. Another critical issue when referring to crowdsourced deliveries is related to regulatory and insurance issues since crowdshippers often operate as independent contractors. It is important to ensure regulatory compliance and adequate insurance coverage.

The elements listed above reflect the rapid and dynamic evolution of distribution logistics, which has become increasingly technology-driven, sustainable, and customer-centric. Businesses that embrace these innovations are better positioned to remain competitive in an ever-evolving market. The evolving world of last mile logistics has spawned several versions of combinatorial optimization problems. For example, regarding the concept of sustainable logistics, the literature offers several contributions in the area of the green vehicle routing problems (GVRPs), to which several variants belong, depending on the objective being pursued and the elements of the logistics network being considered. For more comprehensive knowledge, we refer to the study of Asghari and Mirzapour Al-e-hashem (2021).

There is an evergrowing body of literature on vehicle routing problems that consider crowdshipping in some manner. Alnaggar et al. (2021b) contains a classification of different forms of real-world online platforms that exploit crowdsourced resources to make deliveries, and then an analysis of the scientific literature to assess the realism of the assumptions and applicability to real-world applications. Alnaggar et al. (2021b) concludes that the mathematical models are often too limited with respect to reality.

The purpose of this thesis is to study a particular class of vehicle routing problem identified as the pickup-and-delivery problem (PDP) in which each customer is characterized by a request pickup location and a delivery location in a context of same-day delivery. Furthermore, we consider the customer requests to arrive stochastically throughout a day and that decisions are made

dynamically in response to the newly arriving requests. We study the stochastic and dynamic variant of the PDP because it more closely aligns with our motivating real-world problem setting than the deterministic and static version of the PDP. For the collaborative local delivery platform we consider, customer demands arriving throughout the day and are not known in advance. Furthermore, for platforms utilizing crowdsourcing, the availability of these crowdshippers as well as their originating location and/or destination may not be available a priori.

The main contribution of this thesis is to consider the uncertainty both in the customer demand as well as in the crowdshipping delivery capacity. Furthermore, we also consider time-dependency of travel times so that the amount of time it takes a delivery vehicle to travel between two locations depends on the time of day as well as the geographical location of the origin and destination of the route segment. To the best of our knowledge, there is no other work in the literature that addresses a dynamic PDP with all these characteristics. To solve this challenging problem, we propose algorithmic techniques based on the approximate dynamic programming paradigm (Powell et al. 2012), (Bertsekas 2005).

This thesis is organized as follows. Chapter 2 contains the work titled *Dynamic Pickup-and-Delivery for Collaborative Platforms with Time-Dependent Travel and Crowdshipping*. In this chapter, we model the dynamic PDP as a Markov decision process and present a solution approach using destroy-and-repair principles from heuristic search incorporating a cost function approximation. We benchmark this proposed approach with a myopic solution method based on adaptive large neighborhood search.

Chapter 3 titled *Demand and Capacity Management in a Stochastic Dynamic Pickup and Delivery Problem with Crowdsourced resources* contains a short paper that represents an extension of the work presented in Chapter 1. In this paper the idea is to generalize the approach proposed in Chapter 1

considering different forms of crowdshippers such as in-store shoppers or gig-workers. Chapter 4, titled *Analysis of Crowdshipping Alternatives in Stochastic Dynamic Pickup-and-Delivery* is a work in progress whose goal is to evaluate the economic impact of having a crowdsourced fleet of vehicles, as opposed to considering only a dedicated fleet of vehicles, and to figure out how to size the capacity of both according to the different level of demand they are intended to serve. We conclude the work by summarizing what has been presented and making explicit future research directions in the Conclusions chapter.

Chapter 1

Dynamic Pickup-and-Delivery for Collaborative Platforms with Time-Dependent Travel and Crowdshipping

Joint work with: ***Demetrio Laganà*** **and** ***Jeffrey Ohlmann***
Submitted on an international journal and currently under review process

Abstract: We study a pickup-and-delivery problem that arises when customers randomly submit requests over the course of a day from a choice of vendors on a collaborative e-commerce portal. Based on the attributes of a customer request, a dispatcher dynamically schedules the delivery service on either a dedicated vehicle or a crowdshipper, both of whom experience time-dependent travel times. While dedicated vehicles are available throughout the day, the availability of crowdshippers is unknown a priori and they appear randomly for only portions of the day. With an objective of minimizing the sum

of routing costs, piece-rate crowdshipper payments, and lateness charges, we model the uncertainty in request arrivals and crowdshipper appearances as a Markov decision process. To determine an action at each decision epoch, we employ a heuristic that partially destroys the existing routes and repairs them guided by a parameterized cost function approximation that accounts for the remaining temporal capacity of delivery vehicles. Through a set of computational experiments, we demonstrate the improvement of our approach over a myopic approach in several key performance metrics. In addition, we conduct computational experiments that demonstrate the impact of inserting wait time in the route scheduling and the benefit of explicitly modeling time-dependent travel times. Through our computational testing, we also investigate the effect of demand management mechanisms that facilitate many-to-one request bundles or one-to-many request bundles on reducing the cost to service requests.

1.1 Introduction

For decades, the advance of information and communications technology has facilitated a trending increase in e-commerce. In the early years of e-commerce, brick-and-mortar businesses still possessed the advantage of immediacy as delivery from a website was expensive, slow, and/or inconvenient. The expanding reach of companies such as Amazon.com has allowed it to provide a market place with a wide variety of goods and the ability to deliver many of them within an hour or two in urban areas and within a day or two practically anywhere else.

In this study, we examine the strategy of local brick-and-mortar businesses competing with a large e-commerce provider such as Amazon.com through the establishment of an e-platform that aggregates these local businesses' product offerings and provides delivery service. A primary motivation for local businesses to join a collaborative delivery platform is to compete with the dis-

tribution power of companies like Amazon.com (Mattioli 2021). Pairing an e-shopping platform with a high level of customer service (professional shopper consultation, favorable return policy, etc.) provides local brick-and-mortar companies to facilitate "shop local" initiatives and stave off a "retail apocalypse." The global COVID-19 pandemic that stretched from 2020 to 2023 has altered consumer behavior (perhaps permanently) and local brick-and-mortar stores may need to serve as both a showroom (for in-person shopping) and a warehouse, with employees doubling as salespeople and as order-pickers.

We model the delivery service associated with an e-shopping platform of a consortium of local businesses as a stochastic dynamic pickup-and-delivery problem. In this setting, customers at different locations place requests with various businesses thorough a single e-platform. Each request consists of a pickup location (the business fulfilling the request), a delivery location (the customer placing the request), a ready time specifying when the request will be available for pick-up, and a soft deadline outlining the customer's timeliness expectations. As requests arrive randomly throughout the day, the dispatcher must decide how to serve these requests with the available delivery capacity.

We consider delivery capacity in the form of a fleet of dedicated vehicles supplemented by the presence of crowdsourced couriers which we refer to as crowdshippers. Dedicated vehicles have known availability for day-long shifts while crowdshippers' appearance times are unknown to the dispatcher a priori and their availability duration (declared at appearance to the system) is relatively short. Therefore, a key decision-making element of our problem is whether to service a request with one of the dedicated vehicles or with one of the crowdshippers that is currently available (or may become available in the future).

Further, we consider request dispatching decisions in the context of time-dependent travel for delivery vehicles. As stated by Vidal et al. (2021), "an inadequate management of time-dependent travel times ... represents the great-

est current barrier to the practical adoption of vehicle routing software." We account for traffic patterns by incorporating travel times that depend on the origin and destination as well as the time of day.

To represent the uncertainty in request arrivals and crowdshipper appearances, we model our problem setting as a Markov decision process (MDP) with the objective of minimizing the cost of fulfilling customer requests over an operating day. To identify a policy for the intractable MDP, we employ an approximate dynamic programming (ADP) technique called cost function approximation (CFA) within a destroy-and-repair heuristic. In our CFA, we modify the current epoch cost function via a parameter, tuned via offline simulation, accounting for the delivery vehicles' remaining temporal capacities. As CFA does not require online modeling of the future, it scales well in problem size and facilitates real-time decisions within less than a minute.

Our work contributes to the literature by being the first to consider a dynamic pickup-and-delivery problem with time-dependent travel times and delivery capacity consisting of dedicated vehicles and randomly-appearing crowdshippers. We provide a methodological contribution by devising destroy-and-repair heuristic leveraging a CFA to address the tradeoff between assigning a request to a dedicated vehicle versus a crowdshipper. Our computational experiments demonstrate the effectiveness of our CFA approach versus a myopic strategy. In addition, we computationally exhibit the value of explicitly accounting for time-dependent travel times. Finally, in our computational experiments we examine the impact of two separate demand management mechanisms that influence the arrival of customer requests to the e-platform.

The paper is organized as follows. In Section 1.2, we position our work with respect to the literature. Section 1.3 formally presents the mathematical modeling of the problem as a Markov decision process (MDP). We describe our solution approach utilizing a cost function approximation in Section 1.4. In Sections 1.5 and 1.6, we outline our data sets and present our solution

approach's computational performance, respectively. We conclude with a summary in Section 1.7.

1.2 Literature Review

To frame our contribution, we discuss research related to the two primary characteristics of our problem: dynamic pickup-and-delivery and crowdsourcing. While we focus on research considering a pickup-and-delivery problem with time windows with stochastic and dynamic elements (DPDPTW) for a broader overview of the literature, we refer the reader to the survey on dynamic pickup and/or delivery problems by Berbeglia et al. (2010), the characterization of dynamic vehicle routing problems by Psaraftis et al. (2016), and the survey on stochastic and/or dynamic vehicle routing problems by Ritzinger et al. (2016).

Mitrović-Minić and Laporte (2004) and Mitrović-Minić et al. (2004) consider a DPDPTW and incorporate waiting strategies to implicitly accommodate unknown future requests. Vonolfen and Affenzeller (2016) propose methodology to automatically tailor the waiting strategy based on problem characteristics. While our problem context is complicated by having a heterogeneous fleet of dedicated vehicles and crowdshippers, we also allow a vehicle to wait and our computational experiments demonstrate the value of waiting in our problem setting. For a DPDPTW with different priority classes of requests, Ghiani et al. (2022) devise a policy function approximation for which parameters specifying the fraction of the fleet reserved for each request class are learned by a neural network.

The dial-a-ride problem (DARP) refers to a special case of the pickup-and-delivery problem involving passenger transportation (often the elderly or disabled) and the related operational considerations. Xiang et al. (2008) study a dynamic DARP with time windows and a variety of stochastic events, including uncertain travel times. Leveraging probabilistic information about in-

bound requests provided by outbound requests, Schilde et al. (2011) apply several metaheuristic approaches to a dynamic DARP with time windows motivated by hospital patients. Schilde et al. (2014) extend Schilde et al. (2011) to account for time-dependent stochastic travel times. However, none of these works consider crowdshipping which is a critical element of our setting. While Souza et al. (2022) consider a dynamic DARP with a heterogeneous fleet, this fleet varies by capacity and rental cost, not by appearance time and temporal availability as crowdshippers do in our problem.

Schrotenboer et al. (2021) consider a dynamic stochastic pickup-and-delivery problem that shares the same motivation as our work – a delivery platform that consolidates requests from local stores. Schrotenboer et al. (2021) develop a parameterized CFA based on the Dantzig-Wolfe reformulation of a set packing formulation that assigns requests to vehicles. Once requests are assigned to a vehicle, they cannot be reassigned, requiring an urgency parameter in the CFA that determines whether a request is assigned at an epoch or postponed for consideration at a future epoch. In contrast, we allow reassignment of a request to a different vehicle up until the point at which a vehicle is en route to the request's pickup location, allowing flexibility to make routing adjustments in response to newly-arriving requests. Further, we supplement a fleet of dedicated vehicles with crowdshippers who randomly appear to the dispatching system while Schrotenboer et al. (2021) only consider delivery capacity provided by dedicated vehicles (bikes in their application). Third, in our problem the travel time between stores and customer locations is time-dependent and region-dependent.

Restaurant meal delivery is another application domain for dynamic pickup and delivery that bears similarity to our work. To account for the uncertain ready times at the pickup locations resulting from the stochastic nature of restaurant meal preparation, Ulmer et al. (2021) employ a CFA in which they introduce a time buffer parameter that alters the assignment of requests to

vehicles by effectively penalizing actions that result in deliveries too close to the customer deadlines. They pair this CFA with a strategy of postponing the assignment of requests to drivers. Our study has several differences relative to the work in of Ulmer et al. (2021). First, we supplement a fleet of dedicated vehicles with crowdshippers who randomly appear to the dispatching system while Ulmer et al. (2021) only consider delivery capacity provided by dedicated vehicles. Second, we seek to minimize the sum of routing cost, delivery lateness charges, and piece-rate payments to crowdshippers while Ulmer et al. (2021) minimize just the delivery lateness. Third, in our problem the travel time between stores and customer locations is time-dependent and region-dependent. While we do not consider uncertain ready times for requests, we do note that if there are time-dependent patterns in the ready times of requests (as may be the case with food preparation), these can be incorporated into the time-dependent travel times. Steever et al. (2019) also consider a food delivery problem in which the defining characteristic is that customers may order from more than one restaurant. In our computational experiments, we also consider a set of instances in which there also exists this type of many-pickups-to-one-delivery type of requests.

The same-day delivery problem (SDDP) refers to a variant of the stochastic dynamic pickup-and-delivery problem in which all orders are picked up from a single origin (distribution center). Of particular relevance is the work of Ulmer and Thomas (2018) that considers a SDDP with a heterogeneous fleet for which they determine request assignment (truck or drone) with a policy function approximation (PFA) based on geographical proximity to the distribution center. However, this PFA is not directly applicable to our heterogeneous fleet of dedicated vehicles and crowdshippers that differ with respect to temporal capacity and must serve requests with varying pickup locations. Chen et al. (2022) also consider a SDDP with a heterogeneous fleet of trucks and drones and apply deep reinforcement learning to incorporate more detailed state in-

formation in the request assignment policy. While deep RL is a promising methodology, Chen et al. (2022) consider fleet sizes smaller than in our setting and their results suggest that the benefit of their approach declines as fleet size increases. Dayarian and Savelsbergh (2020) consider a dynamic SDDP in which in-store customers complement a company's fleet to serve online orders; they demonstrate the benefit of incorporating probabilistic information about future online orders and in-store crowdshippers versus a myopic approach. Our modeling framework for crowdshippers is flexible enough to accommodate the in-store customer as a special case. Silva et al. (2023) also consider a SDDP with crowdshippers integrate a value-based function approximation using a neural network with an optimization recourse model to reduce the action space for a scenario, where a scenario consists of a set of delivery requests and whether or not a matching in-store customer exists.

The uncertain appearance time of crowdshippers that provide delivery capacity is a key feature of our dynamic pickup and delivery model. For an overview of operations research literature on crowdsourcing, we refer to surveys by Savelsbergh and Ulmer (2022) and Alnaggar et al. (2021a). We discuss treatments of crowdsourcing delivery in dynamic models.

Arslan et al. (2019) consider a DPDPTW relying on ad hoc drivers, who dynamically announce the origin and destination of planned journeys from which they will detour as specified amount, as well as a dedicated fleet as a backup option. Arslan et al. (2019) formulate the assignment of requests to ad hoc drivers and dedicated vehicles as matching problem that they reoptimize upon the arrival of new information. An exact solution of this matching problem is facilitated by an appearance lead time for the ad hoc driver, narrow time windows of availability (ranging from 10 minutes to 30 minutes), and limits on the number of stops an ad hoc driver will make. While our problem setting is similar, we assume no appearance lead time for crowdshippers, wide availability time windows of one to four hours, and no limits on the number of stops.

For a dynamic delivery problem in which orders originate from a single depot, Ulmer and Savelsbergh (2020) focus on the scheduling of company vehicles in the presence of unscheduled crowdshippers with uncertain appearance times and availability durations. Using a continuous approximation and a value function approximation, Ulmer and Savelsbergh (2020) produce a set of shifts for scheduled drivers that is expected to achieve a target service level for delivery. Behrendt et al. (2023) also focus on the scheduling of couriers while accounting for ad hoc crowdshippers and present a prescriptive machine learning method identify a set of shifts for scheduled couriers that minimizes total courier payments and penalty costs for expired requests.

While it is not a pickup-and-delivery problem, there are some dynamic implementations of the vehicle routing problem with occasional drivers related our work. Archetti et al. (2021) consider a vehicle routing problem in which a regular fleet of vehicles is complemented by a known set of occasional drivers with specified time windows of availability are deployed to serve both static requests known a priori and online requests. Archetti et al. (2021) handle the dynamic requests with an insertion approach and periodic re-optimization via variable neighborhood search. Mancini and Gansterer (2022) consider an extension of the vehicle routing problem with occasional drivers (VRP-OD) in which occasional drivers are assigned customer bundles through a bidding system. Our work differs because we assume that both customer requests and crowdshippers appear dynamically, while Archetti et al. (2021) assumes occasional drivers are known a priori to serve dynamically-arriving requests and Mancini and Gansterer (2022) consider dynamically-appearing crowdshippers to serve customer demand that is known a priori.

1.3 Problem Description and Model Formulation

In this section, we formally describe and formulate the stochastic dynamic pickup-and-delivery problem for our setting. Table 1.1 provides an overview of the notation we employ in our description.

We assume that an e-platform receives the customer requests randomly throughout the day. A customer request r corresponds to a tuple of information: the pickup location (o_r), the delivery location (d_r), the time which the request will be ready (e_r), and a soft deadline representing the latest time it should delivered (l_r). If the vehicle handling customer request r arrives at location o_r before time e_r, the vehicle must wait until e_r before executing the request pick up. If the vehicle handling customer request r delivers the request at location d_r after l_r, it incurs a lateness charge.

To provide delivery service, we assume there is a fleet of dedicated vehicles, $\mathcal{V} = \{1, \ldots, V\}$, and a set of crowdshippers, $\mathcal{G} = \{1, \ldots, G\}$. Each dedicated vehicle $v \in \mathcal{V}$ is available over a known shift starting and ending at the depot, $n_v^a = n_v^b = 0$ for $v \in \mathcal{V}$. For example, a dedicated vehicle v available for the entire day would have a shift from time $a_v = 0$ to time $b_v = \hat{T} > T$, where $\hat{T}$ is sufficiently large enough to ensure delivery of all requests. Each crowdshipper $g \in \mathcal{G}$ is available to service requests during a time window $[a_g, b_g]$ that is unknown to the dispatcher a priori. Upon appearing to the platform at time a_g and declaring their location n_g^a, a crowdshipper g communicates their availability to service requests subject to requirement that they can arrive at location n_g^b by time b_g. The crowdshipper's destination requirement prevents the dispatcher from assigning a request that would result in a crowdshipper making a delivery in a geographically distant location near the end of its declared time window.

The objective is to minimize the cost of fulfilling customer requests over an operating day. To measure the cost of providing service, we minimize the sum of three components: (1) the total travel cost of the dedicated vehicles and

Table 1.1: Overview of notation.

$\mathcal{T}$	time horizon of request arrivals, $t \in \mathcal{T} = [0, 1, \ldots, T]$
$\mathcal{R}_k^p$	set of in-process requests at epoch k
$\mathcal{R}_k^o$	set of outstanding requests at epoch k
$\mathcal{R}_k$	set of active requests at epoch k, $\mathcal{R}_k = \mathcal{R}_k^p \cup \mathcal{R}_k^o$
$\mathcal{U}_k$	set of newly-arriving requests to the system at epoch k
o_r	pickup location of request r
d_r	delivery location of request r
e_r	ready time of request r
l_r	deadline of request r
$\mathcal{V}_k$	set of dedicated vehicles available at epoch k
$\mathcal{G}_k$	set of crowdshippers available at epoch k
$\mathcal{M}_k$	entire pool of vehicles available at epoch k, $\mathcal{M}_k = \mathcal{V}_k \cup \mathcal{G}_k$
n_m^a	starting location of vehicle m
n_m^b	ending location of vehicle m
a_m	appearance time of vehicle m at location n_m^a
b_m	time by which vehicle m must arrive at location n_m^b
θ_m	routing plan of vehicle m
f_m	arrival time of vehicle m at the first location in θ_m
w_m	departure time of vehicle m from the first location in θ_m
s_k	pre-decision state
s_k^x	post-decision state
$\mathcal{X}(s_k)$	set of possible actions in s_k, $x \in \mathcal{X}(s_k)$
$c(s_k, x)$	cost incurred by action x when system in state s_k at epoch k
ρ	per-delivery fee (\$)
μ_1	travel time multiplier (\$ per minute)
μ_2	lateness multiplier (\$ per minute)
ω_k	set of new customer requests arriving at epoch k
γ_k	set of new crowdshippers appearing at epoch k
λ	cost multiplier for a vehicle's remaining delivery time
η	wait time multiplier for a vehicle's remaining delivery time
$\tau(i, j, t)$	travel time between locations i and j when departing at time t

crowdshippers (as function of travel time), (2) the total lateness charge (as a function of the amount of time customer requests are delivered after their soft deadlines), and (3) the total per-delivery fees paid to crowdshippers. Note that we assume a fixed per-delivery fee of ρ for all requests served by a crowdshipper and include the travel cost of the crowdshippers in the objective function to represent the variable fees paid by the e-platform to the crowdshippers to account for differences in the travel demands of requests due to their relative pickup and delivery locations.

To represent the stochastic nature of the arrival of customer requests and the appearance of crowdshippers, we model the problem as a Markov decision process (MDP) over finite, discrete-time horizon. In the remainder of this section, we describe the components of the MDP: the states, the actions and deterministic transition to the post-decision state, and the exogenous information and stochastic transition to the pre-decision state.

1.3.1 States

At a decision epoch k, the pre-decision state s_k contains all the relevant and available information to make dispatching decisions. Specifically, a pre-decision state s_k includes the current time of day (t_k), the status of each active request, the status of each active vehicle, and the set of newly-arriving requests ($\mathcal{U}_k$). We refer to a customer request for which service (pickup and delivery) has not been complete as an *active* request and let $\mathcal{R}_k$ be the set of active requests at epoch k. Each active customer request r corresponds to a tuple of information, (o_r, d_r, e_r, l_r), corresponding to geographical (o_r, d_r) and temporal (e_r, l_r) information. We partition the set of active requests into two subsets, $\mathcal{R}_k^p$ and $\mathcal{R}_k^o$. We refer to each request $r \in \mathcal{R}_k^p$ as *in-process* because a vehicle has already picked up request r or is currently en route to o_r to pick it up. We refer to each request $r \in \mathcal{R}_k^o$ as *outstanding* because a vehicle has not begun its service of request r. So, $\mathcal{R}_k = \mathcal{R}_k^p \cup \mathcal{R}_k^o$ and $\mathcal{R}_k^p \cap \mathcal{R}_k^o = \emptyset$. We differentiate between $\mathcal{R}_k^p$

and $\mathcal{R}_k^o$ because an active request $r \in \mathcal{R}_k^o$ can possibly be reassigned to another vehicle at a future epoch while an active request $r \in \mathcal{R}_k^p$ cannot.

We represent the entire pool of vehicles available at epoch k as $\mathcal{M}_k = \mathcal{V}_k \cup \mathcal{G}_k = \{m \in \mathcal{V} \cup \mathcal{G} : t_k \in [a_m, b_m]\}$. We represent the attributes of vehicle $m \in \mathcal{M}_k$ by the tuple $(\theta_m, b_m, f_m, w_m)$, where $\theta_m = \langle \theta_m(1), \theta_m(2), \ldots \rangle$ is a sequence of locations composing vehicle m's planned route, b_m is the end time of vehicle m's availability, f_m is the arrival time of vehicle m at the first location in θ_m, and w_m is the planned departure time from the first location in θ_m. If $f_m > t_k$, vehicle m is en route to location $\theta_m(1)$. If $f_m = t_k$, then vehicle m has just arrived at location $\theta_m(1)$ at epoch k. If $f_m < t_k$, then vehicle m is idle at location $\theta_m(1)$ with a planned departure time of w_m.

We formally describe a pre-decision state as $s_k = (t_k, \mathcal{R}_k^p, \mathcal{R}_k^o, \mathcal{U}_k, \mathcal{M}_k)$. Algorithm 1 outlines the dynamics of the Markov decision process. Lines 1–23 describe the deterministic transition from state s_k to post-decision state s_k^x as the result of an action x identified by the CFA-based destroy-and-repair heuristic (DRACE). Lines 24–31 describe the stochastic transition from a post-decision state s_k^x to a pre-decision state s_{k+1} as a result of the arrival of random information, W_{k+1}, consisting of arrivals of customer requests and appearances of crowdshippers. We explain these two transitions in the following two sections.

1.3.2 Actions and Deterministic Transition to Post-Decision State

At a decision epoch k, the dispatcher observes the pre-decision state s_k and selects an action $x \in \mathcal{X}(s_k)$, where $\mathcal{X}(s_k)$ is the set of possible actions corresponding to the pre-decision state s_k. An action $x \in \mathcal{X}(s_k)$ consists of two components: (i) it updates the routing plan θ_m for each vehicle $m \in M_k$ through the assignment and routing of each newly-arriving request $r \in \mathcal{U}_k$ and the possible reassignment and resequencing of outstanding requests $r \in \mathcal{R}_k^o$, and (ii) it specifies the departure time of each vehicle currently idle at a location. For

an action to be feasible at epoch k, each vehicle m must be able to complete its planned route before the end of its availability, b_m, and all requests must be assigned to a vehicle. The presence of one or more dedicated vehicles available until all requests are serviced ensures feasibility.

The execution of an action x in a pre-decision state s_k incurs a cost $c(s_k, x)$ and triggers a deterministic transition to a post-decision state s_k^x. Lines 1–23 of Algorithm 1 break down the execution of action x according to its two components: Line 1 addresses the assignment and sequencing of requests and lines 3–23 determine the temporal scheduling of the next segment of each vehicle's route. Specifically, Line 1 applies the CFA-based DRACE heuristic given state s_k to identify an action x that updates the routing plans for each vehicle through the assignment and routing of outstanding requests. In Section 1.4.1, we describe DRACE in detail and formally outline it in Algorithm 2.

After the execution of `DRACE` in Line 1, each request is assigned to a vehicle and each vehicle $m \in \mathcal{M}_k^x$ has an updated routing plan θ_m^x that maintains vehicle m's current location (or destination if still en route), i.e., $\theta_m^x(1) = \theta_m(1)$. Given the updated routing plans, lines 3–23 then determine when each vehicle m that is currently idle at $\theta_m^x(1)$ will depart for $\theta_m^x(2)$. While updating the timing of the next segment of the route execution, we also calculate the cost $c(s_k, x)$ of the action x applied in state s_k. Line 2 initializes the $c(s_k, x)$ to zero.

Line 4 identifies the three cases which a vehicle requires updating of its temporal scheduling: (i) if vehicle m has just arrived at $\theta_m(1)$ at epoch k ($f_m = t_k^x$), (ii) if vehicle m is idle at $\theta_m(1)$ (having previously arrived, $f_m < t_k^x$) and action x changed its next location ($\theta_m^x(2) \neq \theta_m(2)$), or (iii) if vehicle m is idle at $\theta_m(1)$ ($f_m < t_k^x$) and the current time is its planned departure time ($w_m = t_k^x$).

If the arrival time of a vehicle m at its next destination is beyond the ready time of the request corresponding to the next destination, Lines 5–15 execute the immediate departure of vehicle m from $\theta_m^x(1)$ and perform the necessary

updates. We note this implies that a vehicle will always depart immediately for its next destination if it is a delivery location of a request. When vehicle m departs its current location $\theta_m^x(1)$, Lines 6 and 7 set the arrival time and planned departure time for the next destination. Line 8 adds the travel cost to the next destination to $c(s_k, x)$.

Lines 9–15 execute request status updating and cost accounting depending on the next destination's type. If the next destination is a pickup location, then Line 10 removes the corresponding request from the set of outstanding requests and Line 11 adds it to the set of in-process requests. If the next destination is a pickup location, lines 12 and lines 13 add the per-delivery crowdshipper fee ρ to $c(s_k, x)$ if the servicing vehicle is a crowdshipper. If the next destination is a delivery location, then lines 14 and 15 adds any late delivery charge (by multiplying μ_2 by the number of minutes beyond the delivery deadline) to $c(s_k, x)$.

If the location from which a vehicle is immediately departing is a delivery location, then lines 16 and 17 remove this request from the set of in-process requests to represent its service completion. Then Line 18 updates $\theta_m(x)$ by removing $\theta_m(1)$, thus making the route one stop shorter.

If the arrival time of a vehicle m at its next destination is not beyond the ready time of the request corresponding to the next destination, vehicle does not depart immediately and lines 19 and 20 set the planned departure time. In this case, the vehicle will plan to wait at its current location at least the amount of time to make its planned arrival time at its next destination coincide with that location's ready time. This minimum amount of wait time reflects the operational restrictions of request pickup. However, we also consider the possibility of additional *strategic* wait time. By having a vehicle wait at a location beyond the time dictated by the request ready time, it may be possible to acquire additional information in the form of new request arrivals and new crowdshipper appearances. This new information may prompt a change to

the routing plan that would not have been possible if the vehicle had already committed to the next pickup by departing. We must balance the benefit from the information observed by a vehicle waiting beyond the operational minimum with the opportunity cost of consuming a vehicle's temporal delivery capacity being idle when it alternatively could be progressing through its route and therefore possibly serving future requests in a more timely manner. To determine the planned departure time, Line 20 calculates the planned wait time as the larger of: (1) a percentage (given by η) of the vehicle's remaining time availability, and (2) the operational wait time to arrive at the next destination at its ready time. By calibrating the value of η in offline simulation experiments, we consider the balance between too much and too little wait time.

Finally, Line 22 shows that the arrival time at and planned departure time from $\theta_m^x(1)$ is left unchanged if vehicle m is still en route ($f_m > t_k^x$) or if vehicle m is idle ($f_m < t_k^x$) and its next destination wasn't changed or its planned departure time hasn't been reached ($w_m > t_k^x$).

1.3.3 Stochastic Transition to Pre-Decision State

After the execution of an action x triggering a deterministic transition from pre-decision state s_k to post-decision state s_k^x, the arrival of random exogenous information, W_{k+1}, triggers a stochastic transition from s_k^x to s_{k+1}. In our case, $W_{k+1} = (\omega_{k+1}, \gamma_{k+1})$, where ω_{k+1} represents the arrival of a (possibly empty) set of new customer requests and γ_{k+1} corresponds to the appearance of a (possibly empty) set of new crowdshippers declaring their availability to perform deliveries.

Line 24 of Algorithm 1 reflects that we treat the base unit of time to be one minute so that decision epochs occur one minute apart. Line 25 initializes the request information for s_{k+1} from s_k^x. Line 26 updates the set of newly-arriving requests $\mathcal{U}_{k+1}$ with ω_{k+1}. Line 27 supplements the fleet of vehicles $\mathcal{M}_{k+1}$ with γ_{k+1}. Lines 28–31 update the status of the fleet by removing any vehicle whose

Algorithm 1: Dynamics of Markov Decision Process

Data: Pre-decision state $s_k = \left(t_k, \mathcal{R}_k^p, \mathcal{R}_k^o, \mathcal{U}_k, \mathcal{M}_k\right)$ and mechanism to identify an action x
Result: Post-decision state $s_k^x = \left(t_k^x, \mathcal{R}_k^{p,x}, \mathcal{R}_k^{o,x}, \mathcal{U}_k^x, \mathcal{M}_k^x\right)$

$s_k^x \leftarrow \texttt{DRACE}(s_k)$
$c(s_k, x) \leftarrow 0$
for each $m \in \mathcal{M}_k^x$ **do**
  if $(f_m == t_k^x)$ or $\left[(f_m < t_k^x) \text{ and } [(\theta_m^x(2) \neq \theta_m(2)) \text{ or } (w_m == t_k^x)]\right]$ **then**
    if $t_k^x + \tau(\theta_m^x(1), \theta_m^x(2), t_k^x) \geq e_{r(\theta_m^x(2))}$ **then**
      $f_m^x \leftarrow t_k^x + \tau(\theta_m^x(1), \theta_m^x(2), t_k^x)$
      $w_m^x \leftarrow f_m^x$
      $c(s_k, x) \leftarrow c(s_k, x) + \mu_1 \tau(\theta_m^x(1), \theta_m^x(2), t_k^x)$
      if $\theta_m^x(2) == o_r$ for some $r \in \mathcal{R}_k^{o,x}$ **then**
        $\mathcal{R}_k^{o,x} \leftarrow \mathcal{R}_k^{o,x} \setminus \{r\}$
        $\mathcal{R}_k^{p,x} \leftarrow \mathcal{R}_k^{p,x} \cup \{r\}$
        if $m \in \mathcal{G}_k^x$ **then**
          $c(s_k, x) \leftarrow c(s_k, x) + \rho$
      else if $\theta_m^x(2) == d_r$ for some $r \in \mathcal{R}_k^{p,x}$ **then**
        $c(s_k, x) \leftarrow c(s_k, x) + \mu_2 \max\{0, f_m^x - l_r\}$
      if $\theta_m^x(1) == d_r$ for some $r \in \mathcal{R}_k^{p,x}$ **then**
        $\mathcal{R}_k^{p,x} \leftarrow \mathcal{R}_k^{p,x} \setminus \{r\}$
      Update θ_m^x by removing $\theta_m^x(1)$
    else
      $w_m^x \leftarrow t_k^x + \max\left\{\eta\left(b_m^x - t_k^x\right), e_{r(\theta_m^x(2))} - t_k^x - \tau(\theta_m^x(1), \theta_m^x(2), t_k^x)\right\}$
  else
    $f_m^x \leftarrow f_m$, $w_m^x \leftarrow w_m$
end

Data: Post-decision state s_k^x and random information $W_{k+1} = (\omega_{k+1}, \gamma_{k+1})$
Result: Post-decision state s_{k+1}

$t_{k+1} \leftarrow t_k^x + 1$
$\mathcal{R}_{k+1} \leftarrow \mathcal{R}_k^x$
$\mathcal{U}_{k+1} \leftarrow \omega_{k+1}$
$\mathcal{M}_{k+1} \leftarrow \mathcal{M}_k^x \cup \gamma_{k+1}$
for each $m \in \mathcal{M}_{k+1}$ **do**
  if $b_m \leq t_{k+1}$ **then**
    $\mathcal{M}_{k+1} \leftarrow \mathcal{M}_{k+1} \setminus \{m\}$
end

availability has expired.

1.4 Solution Methodology

Let Π be the set of all Markovian deterministic policies, where a policy $\pi \in \Pi$ is a sequence of decision rules: $\pi = (X_0^\pi(s_0), X_1^\pi(s_1), \dots X_K^\pi(s_K))$ where each decision rule $X_k^\pi(s_k) : s_k \to \mathcal{X}(s_k)$ is a function that specifies the action choice when the process occupies state s_k and follows policy π, and K represents the epoch index at the end of the problem horizon. We seek a policy $\pi \in \Pi$ that minimizes the total expected cost, conditional on the initial state s_0:

$$\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{K} c(S_k, X_k^\pi(S_k))|s_0\right].$$

The Bellman equation recursively restates the cost structure as

$$V(s_k) = \min_{x \in \mathcal{X}(s_k)} \{c(s_k, x) + \mathbb{E}\left[V(s_{k+1})|s_k^x\right]\}$$

for $k = 1, \dots, K-1$.

Identifying an optimal policy is challenging due the three curses of dimensionality present in our problem: (1) the state space is multi-dimensional and can grow exponentially, (2) the action space consists of a correspondingly large number of options of assigning and sequencing requests on vehicles, and (3) the outcome space is vast due to the uncertain request arrivals and crowdshipper appearances. Four common strategies for creating policies are: lookahead approximation (LA), value function approximation (VFA), policy function approximation (PFA), and cost function approximation (CFA) (Powell et al. 2012). LAs can require substantial online computation to assess the future impact of an action, which preclude them from being viable options for real-time problems with large action spaces. VFAs approximate the second term of the Bellman equation (cost-to-go), but are limited by dimensionality, which can grow quite large in routing problems with multiple vehicles. PFAs are best

suited when it is possible to design a function (i.e., a policy) that captures the structure of the problem in order to provide decisions. There is no obvious function class to assess the choice of vehicle type (dedicated vehicle or crowdshipper) while also considering the impact of the routing decision on travel cost and lateness charges. Using calibrated parameters, CFAs modify the objective and/or constraints of a deterministic optimization formulation so that it better accounts for future uncertainty.

We apply a CFA which modifies the first term of the Bellman equation (the current epoch cost) to reflect the future impact of the current action. Given a calibrated parameter λ, we consider

$$V(s_k) \approx \min_{x \in \mathcal{X}_{(k)}} \{\bar{c}_k(s_k, x|\lambda)\} \tag{1.1}$$

for $k = 1, \ldots, K-1$. In the remainder of this section, we describe our solution approach that leverages a CFA to identify a policy.

1.4.1 Destroy-and-Repair Accounting for Capacity Expiration (DRACE)

To determine the action x for the system in state s_k at an epoch k, we apply a destroy-and-repair heuristic that assigns and sequences requests not associated with any vehicle. In the evaluation of different ways to assign and route the requests, the heuristic uses a CFA to account for the temporal availability of the vehicles.

Algorithm 2 outlines the procedure which considers the information in the pre-decision state s_k and implicitly identifies an action x by iteratively assigning and sequencing requests on vehicles.

Line 1 initializes the post-decision state by importing the information from the pre-decision state. In particular, we note that the transition from the pre-decision state to the post-decision state is instantaneous, $t_k^x = t_k$. In Lines 2–7, if a currently scheduled but outstanding request has a ready time within

Γ time units of the current time, t_k, we remove it from its current route. The magnitude of Γ affects the required computation time of Algorithm 2; we calibrate the value of Γ so the per-epoch execution of Algorithm 2 remains within one minute to facilitate real-time decision-making in our application.

To prioritize the most urgent requests, Line 8 creates an ordered set $\mathcal{I}$ by placing the current unscheduled requests, $\mathcal{U}_k \cup \{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\}$, in non-decreasing order of l_r. For each request $r \in \mathcal{I}$, Lines 9–19 iteratively determines a vehicle assignment. For the request r currently being considered, Lines 10–19 estimates the impact of assigning request r to vehicle m. Line 11 executes the `DELTA` procedure to estimate: (1) the increase in vehicle m's travel time due to the assignment of request r (Δ_{rm}^{travel}), (2) the increase in vehicle m's delivery lateness due to the assignment of request r (Δ_{rm}^{late}), and (3) the completion time of vehicle m's route (t_f).

If $r \in \{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\} \subset \mathcal{I}$, the `DELTA` procedure estimates Δ_{rm}^{travel}, Δ_{rm}^{late}, and t_f via a reconstruction routine that removes all requests from vehicle m's route and then iteratively reinserts request r and the other requests assigned to vehicle m. If $r \in \mathcal{U}_k \subset \mathcal{I}$, the `DELTA` procedure estimates Δ_{rm}^{travel}, Δ_{rm}^{late}, and t_f via a fast cheapest insertion routine that searches for the positions to place the pickup and delivery locations for request r. We invest less time for the insertion of newly-arriving requests because these requests are generally less urgent than the earlier-arriving outstanding requests.

Based on the `DELTA` procedure, at epoch k, if request r can be added to vehicle m without its route completion time extending beyond its availability window, then lines 10–13 compute c_{rm}, the cost of assigning request r to vehicle m, according to the cost function approximation:

$$c_{rm} = \mu_1 \Delta_{rm}^{travel} + \mu_2 \Delta_{rm}^{late} + \rho I_{\{m \in \mathcal{G}\}} + \lambda(b_m - t_k). \tag{1.2}$$

If vehicle m cannot accommodate request r without its route completion time extending beyond the end of its availability, then we set $c_{rm} = \infty$ in lines 14 and 15. Equation (2.2) consists of four summands capturing the effect of assigning

request r to vehicle m. The first summand corresponds to the product of the travel time multiplier μ_1 and the increase in vehicle m's travel time due to its assignment of request r. The second summand corresponds to the product of the lateness multiplier μ_2 and the increase in vehicle m's violation of request deadlines due to its assignment of request r. In the third summand, $I_{\{m \in \mathcal{G}\}} = 1$ if $m \in \mathcal{G}$ and 0 otherwise implies that the per-delivery fee ρ is charged if m is a crowdshipper and no fee otherwise. The fourth summand corresponds to the product of the CFA parameter λ and the amount of time vehicle m is available to service requests.

The first three summands in Equation (2.2) reflect the economic impact of assigning request r to vehicle m. The values of the parameters μ_1, μ_2, and ρ directly come from the economics of the problem context. The value of μ_1 represents the cost of each minute of vehicle travel time. The value of μ_2 represents the cost of each minute of violating a customer's deadline. The value of ρ represents the per-delivery fee remunerating crowdshippers.

The fourth summand in Equation (2.2) modifies the cost function by introducing a parameterized term that captures the opportunity cost of the limited time availability of the vehicles. While there is not a tangible cost associated with the slack time of a vehicle, accounting for it in this fourth summand rewards actions that assign requests to vehicles whose time availability will expire sooner. This allow greater flexibility in later assignment decisions and also gives the possibility of preserving the temporal capacity of other vehicles in the most critical moments. We determine the CFA parameter λ by tuning it to optimize performance in offline simulation experiments.

In Line 17, we identify vehicle $m^\star$ that achieves the smallest value of c_{rm} for request r. Line 18 executes the routing of request r on vehicle $m^\star$ by inserting o_r and d_r at their minimum cost positions. Then, the procedure continues with the request r' with the next smallest $l_{r'}$.

To relate Algorithm 2 to Equation (1.1), we first express the cost of an

action x inserting the set of unscheduled requests, $\mathcal{U}_k \cup \mathcal{R}_k^o$, onto the existing routes of a delivery fleet as

$$\bar{c}(s_k, x|\lambda) = \sum_{m \in \mathcal{M}_k} \left[\mu_1 \Delta_m^{travel,x} + \mu_2 \Delta_m^{late,x} + \rho q_m^x I_{\{m \in \mathcal{G}\}} + \lambda q_m^x (b_m - t_k)\right],$$

where $\Delta_m^{travel,x}$ is the increase in vehicle m's travel time due to the requests inserted onto its route by action x, $\Delta_m^{late,x}$ is the increase in lateness of vehicle m's deliveries due to the requests inserted onto its route by action x, and q_m^x is the number of requests inserted onto vehicle m by action x.

Determining an $\text{argmin}_x \bar{c}_k(s_k, x|\lambda)$ requires solving a deterministic pickup-and-delivery problem which is difficult (NP-hard). To obtain a solution within the one-minute inter-epoch time, DRACE considers a subset of unscheduled requests $\mathcal{U}_k \cup \{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\}$ and inserts them onto the best vehicle according to Equation (2.2) in increasing order of their delivery deadline. This results in an action $\bar{x}$ with cost

$$\bar{c}_k(s_k, \bar{x}|\lambda) = \sum_{r \in \mathcal{I}} \left[\mu_1 \Delta_{r,m^\star(r)}^{travel} + \mu_2 \Delta_{r,m^\star(r)}^{late} + \rho I_{\{m^\star(r) \in \mathcal{G}\}} + \lambda (b_{m^\star(r)} - t_k)\right],$$

where $m^\star(r)$ is the vehicle on which DRACE inserts request r.

1.4.2 Myopic Policy

To benchmark our DRACE approach, we consider a myopic policy based on the pre-decision state application of an adaptive large neighborhood search (ALNS) heuristic. There is a large body of literature demonstrating the effectiveness of ALNS for a wide range of vehicle routing problems. We refer the readers to the recent survey of Windras Mara et al. (2022) for an in–depth overview of the ALNS framework and its applications. In our work, we consider the ALNS of Van der Hagen et al. (2017) originally designed for a deterministic static pickup and delivery problem and modify it for real-time execution in our stochastic dynamic setting.

Algorithm 2: Destroy-and-Repair Accounting for Capacity Expiration: `DRACE`

Data: Pre-decision state s_k

Result: Post-decision state s_k^x resulting from an action x

$t_k^x \leftarrow t_k$, $\mathcal{R}_k^x \leftarrow \mathcal{R}_k$, $\mathcal{U}_k^x \leftarrow \mathcal{U}_k$, $\mathcal{M}_k^x \leftarrow \mathcal{M}_k$
for each $m \in \mathcal{M}_k^x$ **do**
  for $i = 1$ to $|\theta_m^x|$ **do**
    if $(\theta_m^x(i) == o_r)$ or $(\theta_m^x(i) == d_r)$ for some $r \in \{\mathcal{R}_k^{o,x} : e_r - t_k \leq \Gamma\}$ **then**
      Remove $\theta_m^x(i)$ from θ_m^x
  end
end
$\mathcal{I} \leftarrow \mathcal{U}_k \cup \{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\}$ ordered in increasing l_r
for each $r \in \mathcal{I}$ **do**
  for each $m \in \mathcal{M}_k^x$ **do**
    $(\Delta_{rm}^{travel}, \Delta_{rm}^{late}, t_f) \leftarrow \texttt{DELTA}(r, \theta_m^x)$
    if $t_f \leq b_m^x$ **then**
      $c_{rm} \leftarrow \mu_1 \Delta_{rm}^{travel} + \mu_2 \Delta_{rm}^{late} + \rho I_{\{m \in \mathcal{G}\}} + \lambda(b_m^x - t_k^x)$
    else
      $c_{rm} \leftarrow \infty$
  end
  $m^\star \leftarrow \arg\min_{m \in \mathcal{M}_k^x} \{c_{rm}\}$
  $\theta_{m^\star}^x \leftarrow \texttt{INSERT}(r, \theta_{m^\star}^x)$
end

At a decision epoch k, the ALNS initializes its solution by considering the set of vehicle routes corresponding to the pre-decision state. For each newly-arriving request $r \in \mathcal{U}_k$, the ALNS randomly selects a vehicle m for which insertion of r is feasible with respect to the vehicle's remaining time availability, $b_m - t_k$, and inserts o_r and d_r at the end of vehicle m's route, if it is feasible. In its random vehicle selection, the ALNS first considers all crowdshippers and if there is no feasible insertion on a crowdshipper route, then it considers the set of dedicated vehicles.

To facilitate the real time execution of ALNS, we restrict the set of requests that the ALNS considers in a manner similar to Algorithm 2. Specifically, at decision epoch k, the ALNS considers the removal and reinsertion of the set of requests that have a ready time with Γ time units of the current time, $\{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\}$. We calibrate the value of Γ and the iteration limit so that the ALNS completes execution within one minute.

During each iteration, the ALNS removes n requests from the set $\{\mathcal{R}_k^o : e_r - t_k \leq \Gamma\}$ and then reinserts these requests. The ALNS considers removing requests through *random removal* and the *Shaw removal*. In random removal, each request has an equal probability to be selected. Shaw removal determines the subset of requests for removal using the concept of *relatedness*. For our problem, the relatedness measure between two customer requests r_i and r_j is:

$$q(r_i, r_j) = \phi(t_{o_i o_j} + t_{d_i d_j}) + \chi(|e_i - e_j| + |l_i - l_j|) \tag{1.3}$$

In (1.3), ϕ and χ are constant parameters providing relative weighting of two terms measuring different dimensions of the compatibility of two requests. For a pair of requests r_i and r_j, $t_{o_i o_j} + t_{d_i d_j}$ corresponds to the sum of the average travel time between the pickup locations and the average travel time between the two delivery locations. While in general we consider time-dependent travel times in our problem setting, considering average travel times is sufficient for relatedness because it measures the relative compatibility of two requests' pickup and delivery locations rather than their deadlines. The second term in (1.3),

$|e_i - e_j| + |l_i - l_j|$, measures the temporal compatibility of the requests' ready times and deadlines. At an ALNS iteration, Shaw removal selects randomly one of the outstanding requests and then removes $\lceil n/2 \rceil$ others by identifying them according to the relatedness metric.

After requests have been removed from a solution (either via random removal or Shaw removal), we rebuild the solution using a minimum cost insertion heuristic. The cost calculation considers the sum of routing cost, penalty cost, and crowdshipper fees. More specifically, considering the removed requests ordered on the basis of the urgency of their delivery deadline, we assess the insertion cost of request on each vehicle. For each vehicle, the insertion heuristic finds the feasible minimum cost insertion position for a request pickup location and then for a request delivery location. To speed-up the computation, we evaluate only the first three positions in each route to re-insert a request pickup location, and only the three positions after the pickup insertion position in which the delivery location can be inserted. We then assign the request to the vehicle with the minimum insertion cost.

1.5 Description of Data Sets

In this section, we outline the construction of the data sets which we apply our solution methodology. In Section 1.5.1, we describe the structure of the customer request data in our various instance types. We outline the delivery capacity for our instances, particularly describing crowdshippers in Section 1.5.2. In Section 1.5.3, we explain how we model time- and location-dependent travel times in our instances.

1.5.1 Customer Request Data

As described by Arslan et al. (2019), our problem setting corresponds to a *few-to-many* service request pattern in that there are relatively few pickup lo-

cations (representing the places of business where pickups are originating) and many delivery locations (representing individual customers). To evaluate our solution approach and the effect of the various characteristics of our problem, we consider four collections of benchmark instances. The first collection of instances, which we refer to as the *Ulmer original* (UO) instances, comes directly from Ulmer et al. (2021) and considers 110 potential pickup (restaurant) locations and 32,000 potential delivery (customer) locations in Iowa City, Iowa. There are three classes of instances in which requests arrive according to homogeneous Poisson process over a seven-hour day at low (25.71 requests/hour), medium (34.29 requests/hour), and high (42.86 requests/hour) rates, respectively. Requests have a deterministic ready time of 10 minutes after request arrival and a soft deadline of 40 minutes after request arrival.

Based on the UO instances, we create a second collection of instances, which we refer to as the *nonstationary mixed* (NM) instances, that considers nonstationary customer demand and two types of requests: 50% short-deadline and 50% long-deadline. The short-deadline requests are ready 20 minutes after request arrival and have a soft deadline 60 minutes after request arrival. The long-deadline requests are ready 40 minutes after request arrival and have a soft deadline 120 minutes after request arrival. These instances consider same set of 32,000 potential delivery locations as the UO instances, but add 138 new pickup locations to represent long-deadline request origins so that total number of potential pickup locations is 248. There are three classes of instances in which requests arrive according to a nonhomogeneous Poisson process over a ten-hour day at low (22.5 requests/hour), medium (30 requests/hour), and high (37.5 requests/hour) rates, respectively. We provide the details on the hourly arrival rates of the short- and long-deadline requests for these three instance classes in 1.8.

By modifying the NM instances, we create a third collection of instances, which we refer to as the *n-to-1* or *many-to-one* (MTO) instances, that consider

the possibility of a single customer placing multiple requests, each with a different pickup location. Multiple requests from a single customer arrive at the same time and all requests in a n-to-1 bundle have long-deadlines. We generate the n-to-1 Bundle instances by letting each long-deadline customer have approximately a 90% probability of placing a single request, 7.5% probability of placing two long-deadline requests, and 2.5% probability of placing three long-deadline requests. For comparison purposes, we maintain the total number of daily request arrivals and the ratio of long-deadline to short-deadline requests to be the same as in the NM instances. The n-to-1 Bundle instances represent a situation in which a customer, either of their own volition or in response to an incentive offered on the e-platform, places requests from multiple businesses which then arrive to the dispatcher at the same time.

We create a fourth collection of instances, which we refer to as the *1-to-n* or *one-to-many* (OTM) instances, we modify the NM data instances to consider the possibility of multiple customers (with delivery locations geographically close to each other) place requests with a single business within a short amount of time. All requests in a 1-to-n arrive within 30 minutes of each other and have long deadlines to delivery locations within 2.414 kilometers of each other. We list the probability of a long-deadline request occurring in a 1-to-n bundle in 1.8. The 1-to-n Bundle instances represent a situation in which a long-deadline request from a customer triggers the circulation of promotions to other customers near the triggering customer's delivery location.

1.5.2 Delivery Capacity

For the NM, OTM, and MTO instances, we consider a delivery workforce of five dedicated vehicles and twenty-eight crowdshippers available during randomly-generated one- to four-hour time windows. We generate the appearance times of the twenty-eight crowdshippers according to a homogeneous Poisson process with a rate of three crowdshipper appearances per hour. Each crowdshipper ap-

pears to the system at a randomly-generated origin and destination (specifying where the crowdshipper needs to be located at the end of their availability time window). This destination prevents a request assignment that sends a crowdshipper to a distant delivery location near the end of their time window. We provide the availability information for the set of twenty-eight crowdshippers we consider in 1.8.

For the UO instances, we consider a delivery workforce consisting of three dedicated vehicles available over the entire time horizon and twenty-two crowdshippers available in randomly-generated one- to four-hour time windows. We specify this delivery capacity to be similar to the number of hours of delivery capacity considered by Ulmer et al. (2021) which considers a fleet of 15 dedicated vehicles. The availability information for these crowdshippers is the first 22 entries listed in 1.8.

1.5.3 Time-Dependent Travel Time

Traffic congestion depends on location and time of day. When possible, the effects of traffic congestion may be mitigated by avoiding being at the wrong place at the wrong time. One strategy to achieve this is to incorporate time–dependent travel in the routing model. We implement the time-dependent nature of travel using a speed profile as in Sun et al. (2018), and extend it to also vary by geographic location so that v_{rt} corresponds to the travel speed of a vehicle within region r at time t. For our instances, Table 1.2 lists the 32 travel speed values for v_{rt} corresponding to the four different time periods and eight different geographic regions of the Iowa City area. Figure 1.1 provides a map displaying the eight speed profile regions of the Iowa City area. We provide further details of our travel-time implementation in 1.8.

Table 1.2: Speed profile of v_{rt} values (km/min).

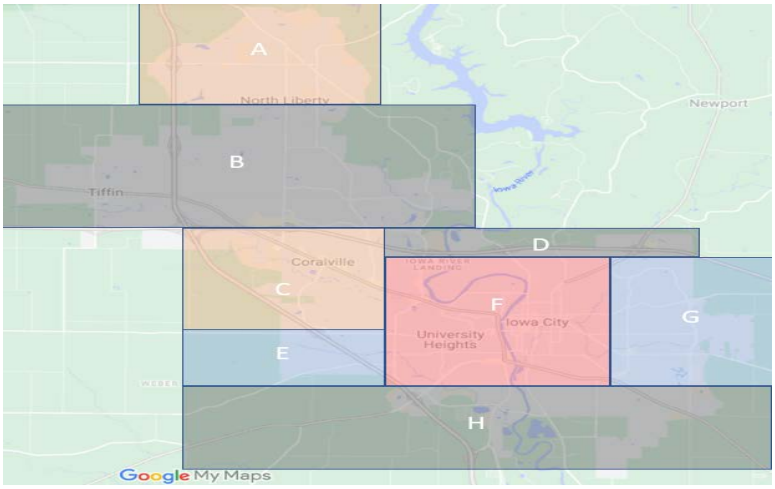
	Time Period (w)			
Region (r)	8:00 - 10:00	10:00 - 18:00	18:00 - 20:00	$>$ 20:00
A	0.25	0.40	0.25	0.40
B	0.50	0.67	0.50	0.67
C	0.25	0.40	0.25	0.40
D	0.50	0.67	0.50	0.67
E	0.33	0.53	0.33	0.53
F	0.16	0.26	0.16	0.26
G	0.33	0.53	0.33	0.53
H	0.50	0.67	0.50	0.67

1.6 Computational Results

There are four objectives of our computational study. First, we demonstrate the effectiveness of DRACE relative to a myopic ALNS approach. Second, we illustrate the impact of allowing vehicles to strategically wait at locations as executed in lines 3–23 of Algorithm 1 by comparing to vehicle scheduling with only the minimum operational waiting time. Third, we examine the effect of different demand patterns by comparing solution quality for the nonstationary mixed (NM), the many-to-one (MTO), and one-to-many (OTM) instances. Finally, we investigate the impact on solution quality of modeling time-dependent travel-time versus using average travel time.

In these computational results, we tune parameter values $\lambda = 0.05$ in Line 13 of Algorithm 2 and $\eta = 0.20$ in Line 20 of Algorithm 1 via offline simulations. We set the cost parameters as $\mu_1 = 1$, $\mu_2 = 5$, and $\rho = 2$. For the myopic ALNS, we calibrate $\phi = 9$ and $\chi = 3$ in Equation (1.3) via offline simulations. To complement the summaries we describe in this section, we provide extended computational results in 1.10.

Figure 1.1: Eight speed profile regions of Iowa City.



1.6.1 DRACE versus Myopic ALNS

For the four different instance types (NM, OTM, MTO, UO) and three different demand levels (low, medium, high), we conduct computational tests on 100 different daily instances. We compare DRACE to the myopic ALNS with respect to the average total cost (the sum of travel cost, late penalty, and crowdshipper fees). In addition, we compare the two approaches with respect to average minutes late per request to observe how they perform with respect to timely service. As a third measure of comparison, we also compute the percentage of requests served by crowdshippers for both approaches.

As Table 1.3 displays, DRACE outperforms the myopic ALNS with respect to average total cost and average lateness per request for all type/demand settings. While the average total cost and average lateness per request causes an increase for both approaches, the increase in cost for DRACE is much smaller than for the myopic ALNS.

A primary difference between DRACE and the myopic ALNS is DRACE's differentiation of delivery vehicles by their remaining temporal delivery capacity via the last term in Equation (2.2) when assigning requests to vehicles.

Comparing the percentage of requests served by crowdshippers in the last two columns of Table 1.3, we observe the CFA mechanism of Equation (2.2) guides DRACE to utilize the crowdshippers more than the myopic approach for all type/demand settings. For the NM instances, DRACE assigns 94% of requests to crowdshippers in the low demand case, 80% of requests to crowdshippers in the medium demand case, and 77% of requests to crowdshippers in the high demand case, with a similar pattern holding for the OTM and MTO instances. As the number of total requests increases with an increase in demand rate, DRACE inevitably assigns a smaller percentage of requests to crowdshippers because dedicated vehicles must serve more requests as the crowdshippers' capacity approaches its maximum utilization. However, the myopic ALNS assigns roughly the same percentage of requests to crowdshippers as demand rate increases because the crowdshippers still have ample capacity under the myopic approach's request assignment. For the UO instances, DRACE's percentage of requests served by crowdshippers also decreases as the demand rate increases although more gradually due to the shorter deadlines and slightly higher average demand rate of the low demand case (homogeneous rate of 25.71 requests/hour versus heterogeneous rate with an average of 22.5 requests/hr). For the UO instances, the myopic ALNS assigns a larger percentage of requests to crowdshippers as the demand rate increases to stem the lateness fees resulting in this capacity-stressed system.

Figure 1.2 displays box plots of the pairwise percent reduction in average cost per request achieved by DRACE versus the myopic approach as computed by $(cost_{myopic} - cost_{DRACE})/cost_{myopic}$. For the NM instances, DRACE achieves a median percent reduction of 33% in the low demand case, 61% in the medium demand case, and 78% in the high demand case, with a similar pattern holding for the OTM and MTO instances (see 1.9). The minimum percent reduction values are positive for all type/demand settings, demonstrating that DRACE achieves a lower total cost in all 400 instances. The shorter delivery

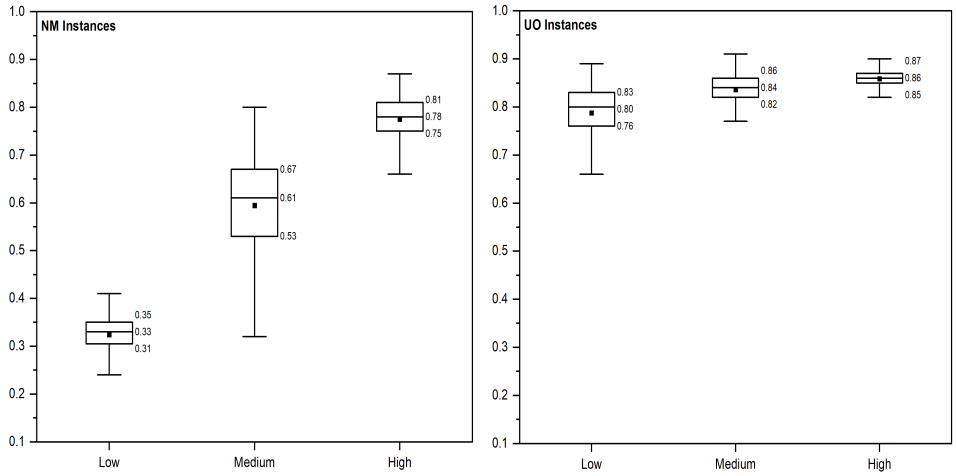
Table 1.3: Comparison of DRACE and myopic ALNS (My.) for four instance types and three demand levels in terms of computational times.

		Avg. Total Cost/Request		Avg. Lateness Per Request		% Requests Crowdserved	
Instance	Demand	My.	DRACE	My.	DRACE	My.	DRACE
NM	low	19	13	8	4	40	94
	medium	40	15	30	7	38	80
	high	74	16	51	8	38	77
OTM	low	21	13	9	4	40	93
	medium	36	18	27	8	39	80
	high	69	20	49	8	38	77
MTO	low	18	12	8	4	39	93
	medium	37	15	27	7	38	81
	high	71	16	49	8	38	78
UO	low	81	16	41	11	39	90
	medium	163	26	59	19	47	87
	high	230	33	69	22	53	85

Table 1.4: Comparison of DRACE and myopic ALNS (My.) for four instance types and three demand levels.

		Avg. Time per Epoch (sec)	
Instance	Demand	My.	DRACE
NM	low	48	50
	medium	27	14
	high	78	40
OTM	low	46	47
	medium	42	13
	high	78	40
MTO	low	44	49
	medium	27	14
	high	78	40
UO	low	21	10
	medium	60	5
	high	62	11

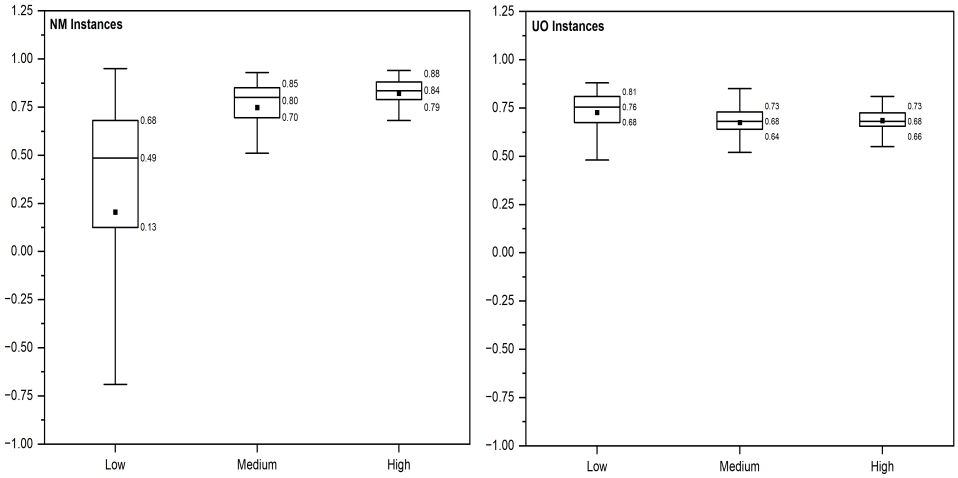
Figure 1.2: Percent reduction in average cost per request by DRACE versus myopic ALNS.



deadlines (30 minutes after request ready time and 40 minutes after request arrival) of the UO instances amplify DRACE's cost advantage over the myopic ALNS. For the UO instances, DRACE achieves a median percent reduction of 80% in the low demand case, 84% in the medium demand case, and 86% in the high demand case. Examining the variation in the distribution of the percent reduction values for the UO instances, we observe that as demand rate increases, DRACE's cost advantage becomes more consistent.

Figure 1.3 displays box plots of the pairwise percent reduction in average lateness per request achieved by DRACE versus the myopic approach as computed by $(lateness_{myopic} - lateness_{DRACE})/lateness_{myopic}$. For the NM instances, DRACE achieves a median percent reduction of 49% in the low demand case, 80% in the medium demand case, and 84% in the high demand case, with a similar pattern holding for the OTM and MTO instances (see 1.9). For the low demand case of the NM, OTM, and MTO instances, there are cases in which the myopic approach results in a smaller average lateness per request than DRACE. In these rare occasions, the CFA mechanism in Equation (2.2) guides DRACE to utilize crowdshipper capacity even though it results in more

Figure 1.3: Percent reduction in average lateness per request by DRACE versus myopic ALNS.



lateness and the conserved dedicated vehicle capacity is not worth this trade-off due to the low demand rate. However, as the demand rate increases in the NM, OTM, and MTO instances, there are no cases of DRACE resulting in more lateness than the myopic ALNS as the increased utilization of crowdshipper capacity becomes progressively more important. The shorter delivery deadlines (30 minutes after request ready time and 40 minutes after request arrival) of the UO instances amplify DRACE's service advantage over the myopic ALNS. For the UO instances, DRACE achieves a median percent reduction of 76% in the low demand case, 68% in the medium demand case, and 68% in the high demand case. For the UO instances, as demand rate increases DRACE's service advantage over the myopic ALNS decreases slightly and levels off. This occurs because the shorter deadlines already result in a capacity-stressed system in the low demand case and further increasing demand does not result in additional opportunities for DRACE's higher utilization of crowdshippers and conserved dedicated vehicle capacity to further increase the service gap between DRACE and the myopic ALNS.

1.6.2 Impact of Strategic Waiting

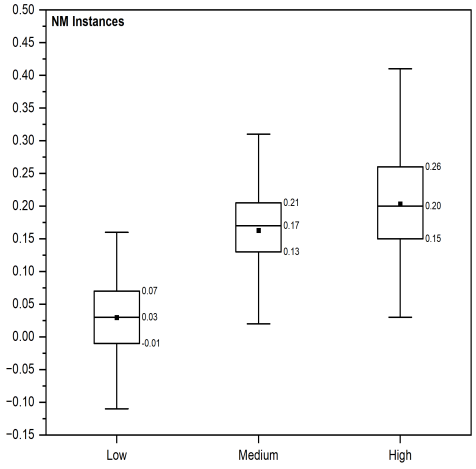
As we describe in Section 1.3.2, there may be a benefit for a vehicle to wait at a location and gain new information in the form of new request arrivals and new crowdshipper appearances. By waiting rather than departing for a request pickup, it may be possible to modify a vehicle's routing plan to reduce the total cost. Alternatively, waiting too long can waste a vehicle's temporal delivery capacity and impair its ability to serve requests in a timely manner.

Governed by the value of the η parameter, lines 19 and 20 of Algorithm 1 consider the possibility of allowing a vehicle to strategically wait at a location beyond the operational minimum. In this section, we examine the total cost of solution with strategic waiting corresponding to the tuned value of $\eta = 0.20$ versus the total cost of a solution with only operational waiting and no strategic waiting ($\eta = 0.00$). For the NM instances, Figure 1.4 displays the percent reduction in average cost per request when executing DRACE with $\eta = 0.20$ versus $\eta = 0.00$. The median percent reduction in average cost per request resulting from strategic waiting is 3% for the low demand case, 17% for the medium demand case, and 20% for the high demand case. As the demand rate increases, strategic waiting is increasingly beneficial because more information (more new request arrivals) is observed. For the low demand cases, there are several instances in which the routing with strategic waiting is more costly than routing without strategic waiting as the new information observed during the additional wait time does not merit the time spent waiting to acquire it.

1.6.3 Impact of Demand Management Mechanisms

In this section, we analyze the difference in the performance of DRACE across the NM, MTO, and OTM instance types. Recall that that MTO and OTM instances modify the NM instances via two different demand management mechanisms. For the MTO instances, "many-to-one " requests consisting of a single

Figure 1.4: Percent reduction in average cost per request with strategic waiting versus no strategic waiting.



customer placing requests with multiple businesses are more prevalent than in the base MN instances; approximately 5% of total requests arrive with one or more other requests from the same customer. For the OTM instances, "one-to-many " requests consisting of multiple different customers (co-located within specified radius) placing requests with the same pickup location within a thirty-minute period; approximately 11% of total requests arrive within one of these 1-to-n bundles.

Table 1.5 contains 95% confidence intervals on the pairwise differences in average total cost between the NM instances and MTO instances, and between the NM instances and MTO instances for all three demand levels. For the low demand case, the both the n-to-1 and 1-to-n bundling results in a statistically significant reduction in the average total cost. The n-to-1 bundling results in reduction between 0.8% to 5.4% of the average total cost in the low demand NM instances. The 1-to-n bundling results in reduction between 0.04% to 5.55% of the average total cost in the low demand NM instances.

As the demand rate increases, Table 1.5 shows that the increasingly stressed

Table 1.5: 95% confidence intervals on the effect of n-to-1 and 1-to-n request bundling on total daily cost.

	Demand		
Instance Comparison	Low	Medium	High
NM - MTO	[24, 151]	[-120, 124]	[-92, 334]
NM - OTM	[1, 157]	[-152, 84]	[-110, 328]

delivery capacity is unable to consistently leverage the n-to-1 and 1-to-n bundling, thereby eliminating any benefit of these demand management mechanisms. Therefore, we conclude that these demand management mechanisms are only effective at lowering costs when delivery capacity is sufficiently sized to take advantage of the tested density of bundled requests. We also note that increasing the density of bundled requests should decrease the delivery capacity required to exploit this request arrival pattern.

1.6.4 Impact of Modeling Time-Dependent Travel Time

In this section, we examine the benefit of explicitly accounting for the time-dependent travel time in our instances (described in Section 1.5.3). We compare the cost of the routing policies that incorporate time-dependent travel time in Table 1.2 when determining an action for a current state to the cost of the routing policies that assume an average travel time of $0.4\bar{3}$ km/min when determining an action for a current state. For the NM instances, Figure 1.5 plots the cost of both the time-dependent travel time (TD TT) policies and the average travel time (Avg TT) policies. Figure 1.5 breaks down the total cost into two components: the routing cost (which includes fees paid to crowdshippers) and the lateness charge. As Figure 1.5 shows, policies that consider time-dependent travel times achieve smaller total costs than their average travel time counterparts for all type/demand settings. Policies that consider time-dependent travel times reduce total cost by an average of 21% over all type/demand settings.

The reduction in total cost by considering time-dependent travel times relies primarily on the reducing lateness charge versus reducing routing cost. TD TT policies reduce lateness charges by an average of 63% but only reduce routing costs by an average of 5% over all type/demand settings relative to Avg TT policies.

For the high demand NM instances, considering time-dependent travel time actually increases the average routing cost by 6% compared to the Avg TT policies. However, the TD TT policies reduce the average lateness charge by 42% in the high demand NM instances, resulting in a decrease of 14% in total cost relative to the Avg TT policies.

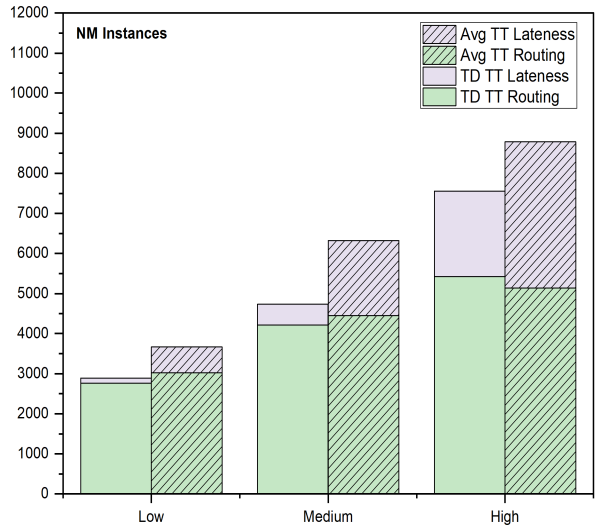
Overall, the results of Figure 1.5 demonstrate that accurately capturing time-dependent travel times in our dynamic pickup-and-delivery problem plays a key role in efficiently assigning requests to vehicles. This finding is important given the computational complications required to account for time-dependent travel times, particularly in a real time setting such in which decisions must be executed on a minute-by-minute basis.

1.7 Conclusion

In this study, we consider a dynamic pickup-and-delivery problem in which delivery capacity is supplied through a fleet of dedicated vehicles and crowdshippers. The appearance time of crowdshippers is unknown a priori, but their availability duration is declared upon appearance. To maintain practical relevance, we incorporate time-dependent and location-dependent travel times to execute request assignment and scheduling actions within a one-minute inter-epoch time.

To form our real time policies to assign and schedule requests, we design a destroy-and-repair heuristic (DRACE) that accounts for the capacity expiration of the delivery vehicles. Our computational results demonstrate that

Figure 1.5: Comparing costs of policies using time-dependent travel time versus cost of policies using average travel time.



DRACE achieves a substantial percent reduction in total cost relative to a myopic procedure and that this advantage increases as the demand rate increases. Further, we show the impact of inserting strategic wait time into the scheduling of the pickup-and-delivery routes and that the benefit of our wait time mechanism increases as the demand rate increases. We also investigate the impact of two demand management mechanisms that facilitate many-to-one and one-to-many request bundles, respectively. Our computational results demonstrate that both of these demand management mechanisms can improve the efficiency of servicing requests, but only when there exists sufficient delivery capacity relative to the demand rate. Finally, we justify modeling time-dependent travel times by showing that this explicit modeling reduces the total cost by 21% over all our instances relative to modeling average travel times.

Our model and solution approach are general enough to accommodate various types of committed and uncommitted crowdshippers (Savelsbergh and Ulmer 2022) including: gig workers committing to availability time windows

known a priori to the dispatcher, (ii) in-store customers who declare themselves as eligible to deliver packages of other customers (Dayarian and Savelsbergh 2020), and (iii) commuters (Arslan et al. 2019). Future research remains to compare and contrast these various types of crowdshippers in a workforce scheduling context (Ulmer and Savelsbergh 2020).

1.8 Data Set Details

In this appendix, we provided details regarding the data sets upon which we apply our solution approach.

In the NM instances, which consider a request mix of 50% short deadlines and 50% long deadlines, the arrival rate of short-deadline requests is non-stationary over the day while the arrival rate of the long-deadline requests is stationary. This captures the difference in demand pattern between requests for perishable items that commonly compose short-deadline requests and requests for non-perishable items that compose long-deadline requests. Table 1.6 specifies the hourly arrival rates of the short- and long-deadline requests for these three demand classes.

Table 1.6: Hourly rates for nonhomogenous Poisson process in NM instances.

	Demand Class					
	Low		Medium		High	
Hour	Short	Long	Short	Long	Short	Long
1	3.75	11.25	5	15	6.25	18.75
2	11.25	11.25	15	15	18.75	18.75
3	18.75	11.25	25	15	31.25	18.75
4	15.00	11.25	20	15	25.00	18.75
5	11.25	11.25	15	15	18.75	18.75
6	3.75	11.25	5	15	6.25	18.75
7	3.75	11.25	5	15	6.25	18.75
8	11.25	11.25	15	15	18.75	18.75
9	18.75	11.25	25	15	31.25	18.75
10	15.00	11.25	20	15	25.00	18.75

For the 1-to-n Bundle instances, Table 1.7 lists the probability of long-deadline 1-to-n bundles occurring for values of n from 1 to 6. Both the yield percentage of a promotion and the customer density affect the distribution of

Table 1.7: Probabilities of a long-deadline request occurring in a 1-to-n bundles.

n	Probability
1	0.782368
2	0.192354
3	0.023308
4	0.001856
5	0.000109
6	0.000005

1-to-n bundles. Our instances correspond to neighborhoods of 70 customers with a yield percentage of 0.35%.

1.8.1 Delivery Capacity

Table 1.8 contains information on the crowdshipper appearance pattern we use in our computational experiments.

1.8.2 Time-Dependent Travel Time

For travel beginning at location i at time t and ending at location j within the same time period w while staying within the same region r, the speed profile of Table 1.2 applies directly and the travel time between i and j is $\tau(i, j, t) = d_{ij}/v_{rw}$, where d_{ij} is the distance (km) between location i and location j. For travel between locations i and j in two different regions r_i and r_j, and/or across time periods, we describe how we approximate the speed profile in 1.8. Finally, we note it is possible to incorporate other time-dependent or region-dependent factors within the $\tau(i, j, t)$ term, such as expected delays at a pickup location i due to a request not being ready at its quoted ready time or expected delays at delivery location j due to congestion or difficulty locating an address in dense residential areas.

Table 1.8: Crowdshipper g availability information.

			Origin		Destination	
g	a_g	b_g	Latitude	Longitude	Latitude	Longitude
1	1	120	41.63562541	-91.51196350	41.65909800	-91.55525400
2	60	180	41.63562541	-91.51196354	41.69834515	-91.58892941
3	70	310	41.64128623	-91.56508335	41.65806850	-91.53102480
4	90	270	41.64885944	-91.55320966	41.64506561	-91.52355511
5	115	295	41.71443676	-91.58289955	41.66152363	-91.47081150
6	115	355	41.67312935	-91.57551051	41.72164463	-91.59286545
7	130	250	41.65001141	-91.46857959	41.72164463	-91.59286545
8	135	315	41.63917026	-91.51290389	41.76164463	-91.69286545
9	140	320	41.66699952	-91.48110701	41.63562541	-91.51196354
10	145	385	41.68153099	-91.57331503	41.64128623	-91.56508335
11	160	340	41.67950253	-91.57390402	41.71443676	-91.58289955
12	170	350	41.63511630	-91.51468220	41.67312935	-91.57551051
13	190	430	41.65263384	-91.58594751	41.65001141	-91.46857959
14	220	400	41.65345535	-91.52692116	41.66699952	-91.48110701
15	250	370	41.64186230	-91.56777907	41.63917026	-91.51290389
16	255	495	41.65787087	-91.46407211	41.64885944	-91.55320966
17	300	420	41.70314675	-91.60940027	41.63583000	-91.51710000
18	310	390	41.69834515	-91.58892941	41.66309000	-91.57927000
19	330	450	41.69986134	-91.56992240	41.65371000	-91.49574000
20	360	580	41.65778645	-91.56992240	41.65529000	-91.53254000
21	375	535	41.69835544	-91.58876611	41.65430000	-91.54275000
22	390	540	41.70713817	-91.58440115	41.66103000	-91.54609000
23	420	620	41.61927800	-91.53541100	41.65157319	-91.48818436
24	480	630	41.64975200	-91.51395990	41.66605613	-91.51510378
25	525	660	41.66704200	-91.53342870	41.68642773	-91.51032070
26	555	705	41.64885944	-91.55320966	41.63202532	-91.50501068
27	570	720	41.64199910	-91.52728740	41.69894765	-91.50420625
28	590	750	41.65911430	-91.54342830	41.64894115	-91.58800303

For travel beginning at location i at time t in time period w and ending at location j such that the travel crosses multiple regions, $r_i \neq r_j$, we approximate the speed profile by weighting the speed profiles of each region intersected by this travel by the size of the respective region. Formally, the travel speed between region r_i and r_j within time period w is approximated by

$$v_{r_i:r_j w} = \sum_{k \in A(r_i, r_j)} \frac{\alpha_{r_k}}{\sum_{k \in A(r_i, r_j)} \alpha_{r_k}} v_{r_k w} \quad \forall w, \tag{1.4}$$

where $A(r_i, r_j)$ is the set of regions intersected by traveling between a location i in region r_i to a location j in region r_j and α_r is the area of region r. For our computational experiments, we apply Equation (2.1) to compute the speed profiles for all $\binom{8}{2} = 28$ possible combinations of inter-region travel.

Let W be the number of time periods in a speed profile. Let the time interval $[l_w, u_w]$ define the time period w. For travel beginning at location i at time t in time period w and ending at location j in time period $w' \neq w$, we compute travel time as

$$\tau(i, j, t) = \sum_{k=w}^{W} \min \left\{ u_k - \max\{l_k, t\}, \frac{d_{ij} - \sum_{s=w}^{k-1} (u_s - \max\{l_s, t\}) v_{rs}}{v_{rk}} \right\}. \tag{1.5}$$

If travel from location i to location j intersects multiple regions, then v_{rk} is replaced by $v_{r_i:r_j w}$ in Equation (1.5).

1.9 Additional Figures from Computational Results

Figure 1.6: Percent reduction in average cost per request by DRACE versus myopic ALNS.

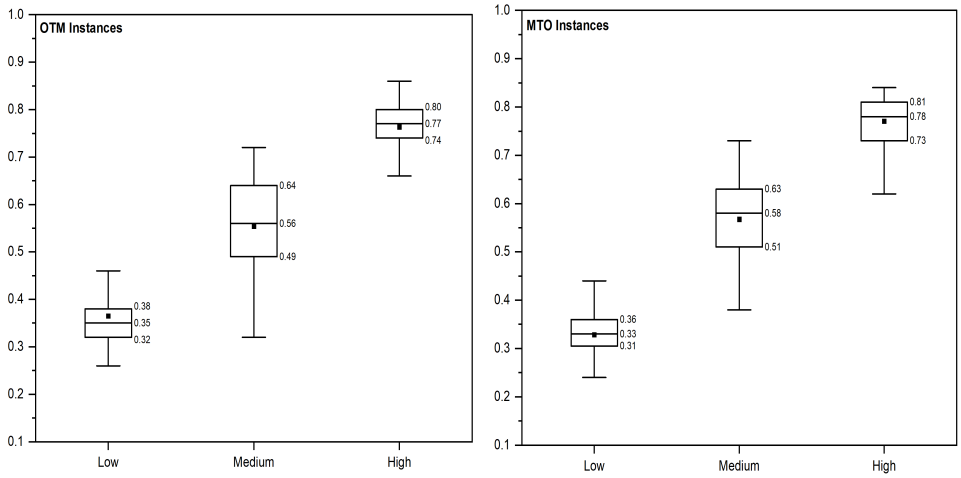


Figure 1.7: Percent reduction in average lateness per request by DRACE versus myopic ALNS.

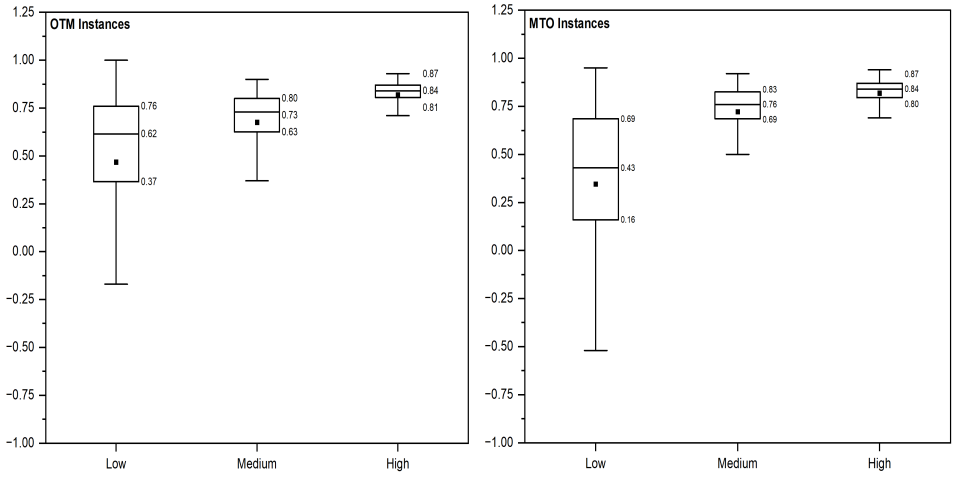
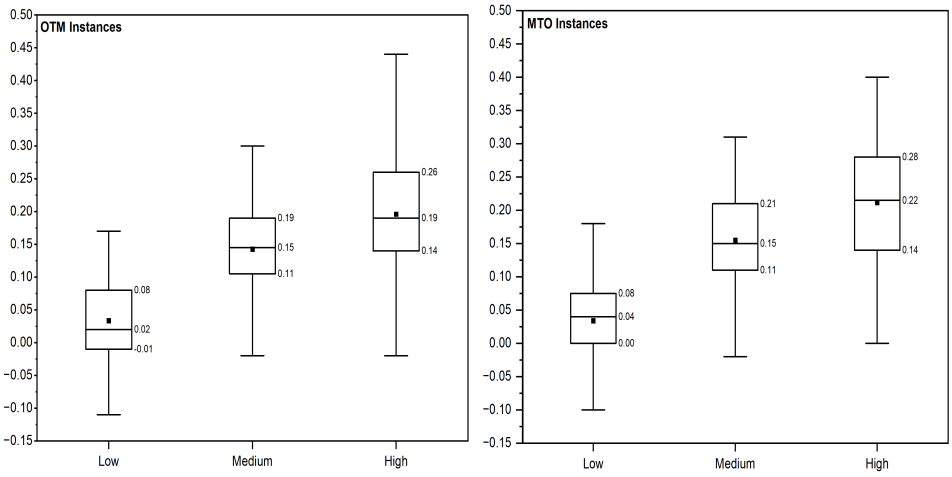


Figure 1.8: Percent reduction in average cost per request with strategic waiting versus no strategic waiting.



1.10 Extended Computational Results

This section comprehensively summarizes all our computational experiments. For each value in the following tables, we calculate the statistics on the output of 100 instances of each type/demand setting. The measure *Total cost* refers to the sum of routing costs and lateness charges for the requests served in a daily instance. *Routing cost* is the sum of travel-related costs and crowdshipper fees for the requests served in a daily instance. *Lateness charge* is the total penalized cost for late deliveries. *Delayed requests* refers to the number of requests served after their deadline. *Total delay* is the total amount of minutes of delay for deliveries over a daily instance. *Crowd service* refers to the number of requests served by the crowdsourced fleet during a day. *Dedicated service* is the number of requests served by the dedicated fleet during a day.

Low NM	KPI	Average	Median	St. Dev.	Min	Max
Myopic	Total cost	4867	4166	375	3166	5496
	Routing cost	4722	4058	271	3165	4670
	Lateness charge	145	52	263	1	1859
	Delayed requests	3	2	3	0	25
	Total delay	8	7	0	0	54
	Crowd service	103	89	10	69	119
	Dedicated service	154	133	62	107	152
DRACE	Total cost	2819	2818	223	2309	3710
	Routing cost	2686	2706	170	2304	3086
	Lateness charge	133	110	108	5	715
	Delayed requests	6	6	3	1	15
	Total delay	4	4	0	0	10
	Crowd service	206	205	14	170	242
	Dedicated service	15	15	5	3	28

Medium NM	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	11803	11086	3476	5563	20575
	Routing cost	4771	4775	239	4289	5391
	Lateness charge	7032	6529	3331	1217	15470
Myopic	Delayed requests	45	45	9	24	67
	Total delay	30	29	0	0	56
	Crowd service	111	113	8	84	128
	Dedicated service	183	185	13	146	208
	Total cost	4461	4480	410	3578	5883
	Routing cost	3950	3957	271	3473	4503
	Lateness charge	511	485	241	75	1380
DRACE	Delayed requests	15	15	5	6	31
	Total delay	7	6	0	0	18
	Crowd service	238	238	12	207	267
	Dedicated service	57	57	11	33	78

High NM	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	27804	26345	7329	13001	48971
	Routing cost	5728	5722	7078	5006	6523
	Lateness charge	22076	20789	7078	7723	42449
Myopic	Delayed requests	84	81	16	55	137
	Total delay	51	51	0	0	70
	Crowd service	141	141	9	122	170
	Dedicated service	230	229	14	199	267
	Total cost	5943	5828	739	4601	7819
	Routing cost	4783	4801	219	4236	5251
	Lateness charge	1160	1123	617	300	2975
DRACE	Delayed requests	26	26	8	10	49
	Total delay	8	8	0	0	17
	Crowd service	285	284	15	241	334
	Dedicated service	87	87	11	57	120

Low OTM	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	4558	4075	1530	3245	5064
	Routing cost	4084	4075	309	3245	5064
	Lateness charge	474	70	1347	0	8428
Myopic	Delayed requests	6	2	10	0	51
	Total delay	9	8	0	0	33
	Crowd service	89	132	9	67	118
	Dedicated service	132	133	9	109	153
	Total cost	2740	2759	224	2230	3442
	Routing cost	2644	2661	175	2205	3122
	Lateness charge	96	70	89	0	585
DRACE	Delayed requests	5	5	3	0	17
	Total delay	4	3	0	0	17
	Crowd service	206	205	14	169	242
	Dedicated service	16	15	5	6	30

Medium OTM	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	10633	10169	2810	5922	17948
	Routing cost	4808	4772	260	4203	5524
	Lateness charge	5825	5261	2630	1503	12561
Myopic	Delayed requests	41	40	9	26	67
	Total delay	27	27	0	0	43
	Crowd service	114	115	8	87	132
	Dedicated service	180	180	13	147	205
	Total cost	4496	4453	442	3646	6310
	Routing cost	4007	3998	240	3517	4570
	Lateness charge	489	423	289	85	1860
DRACE	Delayed requests	13	13	5	3	26
	Total delay	8	7	0	0	19
	Crowd service	236	236	12	211	266
	Dedicated service	58	57	11	33	83

High OTM	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	25870	24486	6653	12217	45908
	Routing cost	5733	5716	321	4528	6595
	Lateness charge	20137	18754	6442	6937	39465
Myopic	Delayed requests	80	77	15	51	125
	Total delay	49	50	0	0	65
	Crowd service	143	142	9	125	172
	Dedicated service	229	228	14	198	265
	Total cost	5834	5711	835	4469	10831
	Routing cost	4772	4763	231	4251	5367
	Lateness charge	1062	915	708	195	5585
DRACE	Delayed requests	24	24	8	9	58
	Total delay	8	8	0	0	19
	Crowd service	285	284	13	259	323
	Dedicated service	87	85	12	62	119

Low MTO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	4070	4109	299	3283	5345
	Routing cost	3973	3972	251	3281	4540
	Lateness charge	97	50	134	2	854
Myopic	Delayed requests	2	2	2	0	9
	Total delay	8	6	0	0	47
	Crowd service	87	87	10	63	116
	Dedicated service	135	135	8	111	154
	Total cost	2731	2750	237	2170	3360
	Routing cost	2611	2620	181	2150	3034
	Lateness charge	120	100	99	10	485
DRACE	Delayed requests	6	6	3	1	14
	Total delay	4	3	0	0	12
	Crowd service	206	205	14	172	240
	Dedicated service	15	14	5	4	30

Medium MTO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	10885	10750	2948	5458	19588
	Routing cost	4716	4739	257	4161	5244
	Lateness charge	6169	6158	2782	4161	5244
Myopic	Delayed requests	43	43	9	22	67
	Total delay	27	27	0	0	44
	Crowd service	112	112	7	95	129
	Dedicated service	182	185	13	147	208
	Total cost	4459	4461	466	3578	6353
	Routing cost	3909	3933	253	3468	4484
	Lateness charge	550	538	289	75	2150
DRACE	Delayed requests	16	16	5	6	31
	Total delay	7	7	0	0	18
	Crowd service	238	238	12	212	265
	Dedicated service	56	55	11	33	80

High MTO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	26724	25561	7116	13242	47522
	Routing cost	5823	5646	1102	4956	14959
	Lateness charge	20901	19891	6924	8287	40990
Myopic	Delayed requests	83	80	16	52	137
	Total delay	49	50	0	0	64
	Crowd service	141	140	9	116	174
	Dedicated service	231	230	13	198	264
	Total cost	5822	5571	798	4569	9495
	Routing cost	4665	4635	238	3955	5347
	Lateness charge	1157	1025	658	305	4265
DRACE	Delayed requests	26	26	8	10	54
	Total delay	8	8	0	0	16
	Crowd service	288	285	15	258	330
	Dedicated service	84	83	12	57	119

Low UO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	14670	13541	5296	6590	28656
	Routing cost	2414	2383	203	2019	2874
	Lateness charge	12256	11099	5122	4481	25783
Myopic	Delayed requests	58	55	14	35	91
	Total delay	41	40	8	24	58
	Crowd service	69	68	10	52	94
	Dedicated service	109	109	6	96	121
	Total cost	2895	2783	603	2109	4855
	Routing cost	1981	1979	116	1679	2335
	Lateness charge	914	793	549	230	2880
DRACE	Delayed requests	16	16	5	7	33
	Total delay	11	10	5	4	36
	Crowd service	161	160	12	137	189
	Dedicated service	19	18	4	10	32

Medium UO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	39423	38173	9540	19465	83466
	Routing cost	3733	3318	2503	2555	21346
	Lateness charge	35690	34510	9905	12214	79236
Myopic	Delayed requests	120	118	21	68	199
	Total delay	59	59	8	25	80
	Crowd service	114	114	14	77	157
	Dedicated service	127	127	6	112	142
	Total cost	6390	6180	1555	3395	11221
	Routing cost	2510	2507	140	2159	2823
	Lateness charge	3881	3655	1499	985	8510
DRACE	Delayed requests	41	38	10	20	69
	Total delay	19	19	9	9	30
	Crowd service	211	210	16	161	253
	Dedicated service	32	32	6	20	52

High UO	KPI	Average	Median	St. Dev.	Min	Max
	Total cost	66743	66188	9261	45013	93327
	Routing cost	4710	4041	6899	3356	72976
	Lateness charge	62033	62036	10999	355	88825
Myopic	Delayed requests	180	181	15	135	216
	Total delay	69	68	5	59	88
	Crowd service	152	152	11	127	181
	Dedicated service	137	138	4	120	146
	Total cost	9838	9672	2263	4397	16284
	Routing cost	2896	2906	155	2416	3308
	Lateness charge	6942	6745	2218	1580	13390
DRACE	Delayed requests	63	62	12	38	101
	Total delay	22	22	8	8	34
	Crowd service	256	256	12	227	285
	Dedicated service	45	45	7	30	68

Chapter 2

Demand and Capacity Management in a Stochastic Dynamic Pickup-and-Delivery Problem with Crowdsourced resources

Joint work with: ***Demetrio Laganà and Jeffrey Ohlmann***
Accepted on ***"Optimization in Green Sustainability and Ecological Transition"*****, Airo Springer Series 2023**

Abstract: In this study we examine the situation of a consortium of brick-and-mortar businesses operating a e-portal to collect customer orders and provide same-day delivery as a dynamic and stochastic pickup-and-delivery problem in which deliveries are performed with a mixed fleet of vehicles composed of dedicated vehicles and crowdsourced resources. Forms of delivery capacity vary with respect to the level of information and control that the dispatcher has

on the availability and behavior of the courier. Taking into account the specific nature of the different types of delivery capacity, we present an approximate dynamic programming approach to manage the same-day delivery. We present computational results for different scenarios of demand. In this way we would like to extend the computational results presented in Chapter 1, proving that the proposed approach could be generalized to different forms of crowdsourced resources.

2.1 Motivation

Delivery services are prominent within the grocery and restaurant domain (e.g., Grubhub, Chomp, Uber Eats, and Instacart), focusing on the delivery of food within a near-immediate deadline or within a narrowly-specified time window later in the day. While the demand pattern and consumer behavior for non-food items differ than that of perishable food items, similar online service platforms (Kanga, Renren Kuaidi, Deliv, Trunkrs) have been developed for electronics, hardware, clothing, small appliances, housewares, health and beauty products, and sporting goods. To compete with the distribution power of companies like Amazon.com, there is motivation for local businesses to join a collaborative delivery platform, as highlighted by Mattioli (2021).

Pairing an e-shopping platform with a high level of customer service (professional shopper consultation, favorable return policy, etc.) provides an opportunity for local brick-and-mortar companies to facilitate "shop local " initiatives and stave off a "retail apocalypse." The global COVID-19 pandemic has altered consumer behavior (perhaps permanently) and local brick-and-mortar stores may need to serve as showrooms (for in-person shopping) and/or dark stores (warehouses), with employees serving as salespeople and order-pickers.

Similar to Schrotenboer et al. (2021), we cast the situation of a consortium of brick-and-mortar businesses operating an e-portal to collect customer orders

and provide same-day delivery as a dynamic and stochastic pickup-and-delivery problem. Using this framework, we investigate the impact of various modes of delivery capacity on the ability to provide quality service. In particular, we focus on the usage of a mixed fleet of vehicle composed of dedicated vehicles and crowdsourced resources.

The use of crowdshipping in last mile logistics is growing dramatically as a means of providing fast delivery while also managing costs by reducing (or eliminating) the required number of full-time workers to operate dedicated vehicles. There are other advantages related to the use of crowdshipping directly correlated with the concept of sustainability. Usually the crowdshipping is done by small vehicles (passenger cars, bicycles, etc.) which have lower carbon emissions than commercial delivery vehicles. Furthermore, if the crowdshipper is a driver adding deliveries to extant commute, then crowdsourcing can reduce traffic congestion.

The use of crowdshippers also has drawbacks as this delivery alternative reduces the dispatcher's control. Because crowdshippers are not employees, the dispatcher may not know with certainty their availability and behavior. A recent survey by Savelsbergh and Ulmer (2022) highlights the opportunities that crowdsourced couriers offer and defines different types of crowdsourced resources based on the delivery platform's level of control and amount of information it possesses about the crowdshipper.

The remainder of this paper is organized as follows: in Section 2.2 the problem is formally described. In Section 2.3 we model the problem as a Markov Decision Process. In Section 2.4 we present our solution approach. In Section 2.5 we describe the used instances and we discuss the obtained results. In Section 2.6 we provide some remarks and the orientation of the future research work.

2.2 Problem Description

We assume that a dispatcher receives the customer requests stochastically throughout the day and assigns these requests to available vehicles. Each customer request p must be picked up at location o_p and delivered to location d_p. Associated with each customer request p is the earliest time (e_p) it may be picked up and a soft deadline (l_p) representing the latest time it should be delivered. If a vehicle handling customer request p arrives at location o_p before time e_p, the vehicle waits until e_p. If the vehicle handling customer request p delivers the request at location d_p after l_p, it incurs a lateness charge.

To model the non-stationary effects of traffic congestion, we consider time-dependent travel times. Specifically, if a vehicle departs location i at time t, the travel time from location i to location j is given by $\tau_{ij}(t)$. We assume that any service time at location i is included in the travel time $\tau_{ij}(t)$. By including the service time at location i in the travel time $\tau_{ij}(t)$, we are able to model time-dependent factors such as delays at a pickup location i due to a request not being ready, or delays at delivery location i due to congestion in dense residential areas.

In addition, we consider time-dependent travel times that depend on geographic location. We divide the entire service area into sub-regions characterized by a distinct speed profile dependent on the nature of the roads and traffic in the sub-region. For a region r, a speed profile defines the speed values $\{v_{r1}, v_{r2}, \ldots, v_{rW}\}$ for a set of W time periods partitioning the operating day. In each time period the speed is considered a constant value, and this value changes across the different time periods. In the case of travel between locations i and j that crosses more than one area, we approximate the speed profile by weighting the speed profiles of each region intersected by this travel by the size of the respective region. Formally, the travel speed between region

r_i and r_j within time period w is approximated by

$$v_{r_i:r_j w} = \sum_{k \in A(r_i, r_j)} \frac{\alpha_{r_k}}{\sum_{k \in A(r_i, r_j)} \alpha_{r_k}} v_{r_k w} \quad \forall w, \tag{2.1}$$

where $A(r_i, r_j)$ is the set of regions intersected by traveling between a location i in region r_i to a location j in region r_j and α_r is the area of region r.

For travel subject to time- and location-dependent travel times, the objective is to minimize the cost of fulfilling customer requests with a fleet of dedicated vehicles, $\mathcal{M}$, complemented with a fleet of crowdshippers, $\mathcal{L}$. To measure the cost of providing service, we minimize the sum of three components: (1) the total travel cost of the dedicated vehicles and crowdshippers (as function of travel time), (2) the total lateness charge (as a function of the number of minutes that customer requests are delivered after their soft deadlines), and (3) the total per-delivery fees paid to crowdshippers. We note that the cost of using a crowdshipper for a delivery includes both a fixed per-delivery fee (that may vary based on type of crowdshipper) and a variable travel-based cost to reflect for differences in the travel demands of requests due to their varying pickup-and-delivery locations.

The functional difference between the fleet of dedicated vehicles, $\mathcal{M}$, and the fleet of crowdshippers, $\mathcal{L}$, is amount of control and information about the availability of the delivery capacity that the dispatcher possesses. For our application, we measure delivery capacity in terms of the amount of time that a vehicle has available to serve customer requests. Because the physical size of the items composing a request is generally small and there are a relative few number of requests picked up before being delivered due to the request deadlines, we do not consider spatial storage capacity of the vehicles.

Using the terminology of Savelsbergh and Ulmer (2022), each dedicated vehicle $m \in \mathcal{M}$ is a *fully committed* courier. That is, the dispatcher has full control over a dedicated vehicle and certain knowledge of its availability. We assume that each dedicated vehicle $m \in \mathcal{M}$ originates at a depot and is

available over the entire day (from time 0 to time T).

In contrast, the dispatcher has varying levels of control and knowledge about the delivery capacity provided by different types of crowdshippers. In our treatment, we assume the dispatcher does not have a priori knowledge when crowdshippers will appear on the platform. Upon linking to the platform at time a_l, the crowdshipper l declares their location n_l^a and their availability to execute pickups and deliveries subject to requirement that they can arrive at location n_l^b by time $b_l > a_l$. The destination requirement prevents the dispatcher from assigning a request that would result in a crowdshipper making a delivery in a geographically distant location near the end of its declared time window. In this manner, we model gig workers who may commit to one- to four-hour shifts as well as commuters willing to make relatively short diversions (e.g., 30 minutes) on their planned route to/from their home, workplace, etc.

2.3 Markov Decision Process

We model the sequential decision-making process of this pickup-and-delivery problem with uncertain arrivals of customer requests and the uncertain appearance of crowdshippers as a Markov decision process (MDP). In a MDP, a decision epoch k represents the moment when the decision maker executes an action based on the state of the system, S_k. We treat the base unit of time to be one minute so that decision epochs occur one minute apart. That is, if the time of decision epoch k is t_k then the time of decision epoch $k+1$ is $t_{k+1} = t_k + 1$. We assume the platform receives requests on a daily basis from time 0 to specified time $\tilde{T}$, but the delivery of requests continues until a (random) time $T \geq \tilde{T}$ when all requests have been served.

The (pre-decision) state variable S_k describes the information available to the decision-maker at epoch k. Specifically, S_k includes the current time of day, information related to the set of unserved customer requests, information

regarding each vehicle's location and current planned route, and the temporal availability of each of the vehicles in the fleet.

Based on the state S_k at a decision epoch k, the decision-maker selects an action, which consists of the assignment and sequencing of outstanding requests to vehicles, where a request p is *outstanding* if a vehicle has not visited its pickup location o_p and begun its service of this request. In addition to the assignment of requests, an action also specifies how long an empty and idle vehicle will wait at its current location. A vehicle is *empty* if it does not contain any requests, he already performed all the assigned services i.e., requests that the vehicle has picked up but not delivered. A vehicle is *idle* if it is at a location, i.e., not in transit between locations. Allowing empty and idle vehicles to wait at their current locations facilitates the improved bundling of requests as more requests can be observed before routing decisions are finalized. On the other hand, too much waiting risks wasting a vehicle's time idling, which is especially a concern for crowdshippers which may have limited time windows of availability.

The selected action incurs an immediate cost and the combination of state S_k and selected action a results in a deterministic transition to a post-decision state S_k^a. Random information, in the form of customer requests and appearance of additional crowdshippers, then arrives resulting in a stochastic transition to the pre-decision state of the next decision epoch, S_{k+1}. The objective is to minimize the cost of fulfilling customer requests over an operating day. We assume that all the requests must be served during each day and no requests can be rejected. Contrary to assumptions made in similar problems in the literature, we assume that an assignment once made can be subsequently modified if more favorable service conditions arise thanks to the revelation of new requests or new vehicles, up to the moment in which the pickup of the request is not carried out.

2.4 Solution Approach

The multidimensional state, action, and outcome spaces afflict the problem with the curse of dimensionality. To guide the action selection, we employ a parametric cost function approximation (CFA) approach that modifies the current period cost term in the Bellman equation. Explicitly, the cost of assigning of request p to a vehicle $v \in \mathcal{M} \cup \mathcal{L}$ is computed by

$$c(p, v) = \mu_1 \Delta_{travelingTime} + \mu_2 \Delta_{lateness} + \rho I_{\{v \in \mathcal{M} \cup \mathcal{L}\}} + \lambda(b_v - t_k) \tag{2.2}$$

The first term in (2.2) computes the increase in traveling time that results from assigning request p to vehicle v (multiplied by a cost multiplier μ_1). The second term in (2.2) computes the increase in delivery lateness that results from assigning request p to vehicle v (multiplied by a cost multiplier μ_2). The third term in (2.2) corresponds to the fixed per-request fee ρ that is paid if the vehicle v is a crowdshipper, i.e., $I_{\{v \in \mathcal{M} \cup \mathcal{L}\}} = 1$ if $v \in \mathcal{L}$ and 0 otherwise. The fourth term in (2.2) computes the amount of time that vehicle v has remaining to make deliveries by computing the difference between the end of its availability time window (b_v) and the time of the epoch (t_k), multiplied by the CFA parameter λ which is tuned via offline simulation experiments.

The motivation behind the CFA is to utilize the relatively limited crowdshipper availability and make the system more flexible to manage future requests. Guided by the $c(p, v)$ values computed at the pre-decision state s_k, our approach iteratively inserts the outstanding requests to the available vehicles, considering the requests in the order of their urgency. The action at decision epoch k for each vehicle v is implicitly defined by the first element of the remaining segment of their route plan.

The waiting time of empty and idle vehicles at their current location is computed by considering a *mininum wait time* and *maximum wait time*. We compute the minimum wait time by noting that it is not convenient for an empty vehicle to depart from its current position and arrive to the first sched-

uled pickup location before the release date of the corresponding request. According to this logic, we compute the minimum wait time as the difference between the release date of the scheduled request and the arrival time of the vehicle to the pickup location if the vehicle would leave immediately. We define the minimum waiting time as this difference if it is positive and zero otherwise. If minimum waiting time is greater than zero, we also compute the maximum waiting time for newly-appeared crowdshippers or when a dedicated vehicle returns to the depot and should start a new route. We compute maximum waiting time as a fixed percentage of the total remaining temporal capacity of the vehicle (β). This β value is fixed and determined through computational experiments. By employing waiting time for vehicles and allowing the reassignment of any request not already picked up, we facilitate efficient bundling of requests.

To benchmark our approximate dynamic programming solution approach, we develop a myopic approach that only considers the current state information in determining an action and ignores any future implications of the action choice. Our myopic approach is based on the adaptive large neighborhood search (ALNS) heuristic of Van der Hagen et al. (2017).

2.5 Computational Results

To evaluate our solution approach and the effect of the various characteristics of our problem, we create a set of instances by modifying the benchmark data set from Ulmer et al. (2021). The geographical coordinates of pickup-and-delivery points are extracted from the map of Iowa City (IA, USA). In our base set of instances, there is a 50%/50% mix of requests with short deadlines (60 minutes) and long deadlines (120 minutes) which arrive according to a non-homogenous Poisson process over a ten-hour working day. We consider three different demand levels: low (average of 225 daily requests), medium (average

of 300 daily requests), and high (average of 375 daily requests).

We consider demand management mechanisms that reflect two different discount mechanisms. In the first mechanism, when a customer places an order with a business, it triggers discount offers to that customer from other businesses in an effort to solicit more orders from the same customer. This first mechanism results an increasing prevalence of " many-to-one " orders consisting of a single customer placing orders with multiple businesses. We generate a n-to-1 set of instances that reflect the demand management mechanism that when a customer places a long-deadline order with a business, this customer also places orders with a random number of other long-deadline orders with other businesses.

In a second demand management mechanism, when a customer places an order with a business, it triggers discount offers from this same business to other customers geographically near the originating customer. This second mechanism results an increasing prevalence of "one-to-many " orders consisting of a multiple customer placing orders with a single business. We generate a 1-to-n set of instances that reflect the demand management mechanism that when a customer places a long-deadline order with a business, a random number of customers (with locations close to the original customer) place long-deadline orders with the same business.

For the crowdsourced vehicles we assume they follow an homogeneous Poisson process over time leading to uniformly distributed crowdshippers' appearances during the whole working day in which the requests arrive. The origin and destination of each vehicle are randomly chosen from all potential known locations in the Iowa City area. The time window of each vehicle is randomly chosen in a set of possible time windows that varies from one hour to four hours.

For a base capacity setting of 5 dedicated vehicles and 28 crowdsourced vehicles with uncertain appearance times and varying availability durations, we

conduct a wide range of computational experiments. In Table 2.1, we provide the average total cost for each delivered request (*Per Delivery Cost*) and the average computational time (in seconds) used by the algorithm to execute an action (*Per Epoch Time*). The average values were calculated over 100 realizations of each instance type and for each demand level. The results show that our CFA approach outperforms the myopic approach across all demand settings and request patterns. It is worth noting that *Per Delivery Cost* increases as the demand level increases for the fixed delivery capacity we consider, but the increase is much less for the CFA approach than for the myopic approach.

The parameter values used in the objective function are $\mu_1 = 1$ \$/min, $\mu_2 = 5$ \$/min, $\rho = 2$ \$/req and $\lambda = 0.05$ \$/min.

If we compare the CFA approach with the myopic approach in the case of *Base* Instances we can observe a percentage reduction in terms of *Per Delivery Cost* equal to $30,3\%$ for small demand level, $52,6\%$ for medium demand level and $72,8\%$ for large demand level. A similar trend is obtained in the case of n-to-1 and 1-to-n Instances. For the first set the reduction is equal to $30,2\%$ for small demand level, 52% for medium demand level and $74,6\%$ in the case of large demand level. In 1-to-n Instances we have a decreasing of $32,5\%$ for small demand cases, 48% for medium demand level and $71,6\%$ for large demand cases.

Comparing the computation time in Table 2.1, the CFA approach outperforms the myopic approach, especially for medium and large demand levels. Comparing the average computation time per epoch for the CFA approach across demand levels, we observe that computation time is higher in the small demand case than for the medium and high demand cases. This occurs because the CFA applies rules-of-thumb that cap the number of requests to be modified at an epoch in order to keep the execution time less than one minute. In contrast, for the myopic approach we specify a certain number of iterations as a stopping criterion for each demand level so that the execution time comparable

Table 2.1: Average time per epoch and total cost values.

		Per Delivery Cost		Per Epoch Time	
Instance Type	Demand	Myopic	CFA	Myopic	CFA
Base	low	18.53	12.92	48	50
	medium	37.66	17.85	27	14
	high	74.14	20.11	78	40
n-to-1	low	17.97	12.54	44	49
	medium	37.87	18.14	27	14
	high	78.49	19.95	65	36
1-to-n	low	18.23	12.31	46	47
	medium	33.41	17.36	42	13
	high	68.99	19.57	78	40
Ulmer	low	81.5	16.1	20.6	9.7
	medium	164.3	26.6	60	5.25
	high	222.5	32.8	62	10.92

to that of the CFA.

We also conduct computational tests on the original instances from Ulmer et al. Ulmer et al. (2021), called *Ulmer* in Table 2.1, where the requests arrive according to a homogeneous Poisson process over a six-hour working day and all the requests have short-deadlines (40 minutes). Also in this case we consider three different demand levels: low (average of 180 daily requests), medium (average of 240 daily requests), and high (average of 300 daily requests). For this six-hour working day, we use capacity settings of 3 dedicated vehicles and 22 crowdsourced vehicles with uncertain appearance times and varying availability durations. Table 2.1 shows that our CFA approach outperforms also in this case the myopic approach across all demand patterns.

For the *Ulmer* instances, a comparison of the CFA approach to the myopic approach shows a reduction in *Per Delivery Cost* of $80,2\%$ for low demand level, $83,8\%$ for medium demand level and $85,3\%$ for high demand level. Relative

to the results of *Base*, n-to-1, and 1-to-n instances, we observe from the *Ulmer* instances' results that the improvement of the CFA over the myopic approach is even more extreme. Compared to the other three types of instances, the *Ulmer* instances consider requests with shorter deadlines and less delivery capacity to service these requests. Thus, as the system becomes more stressed with respect to the demand-to-capacity ratio, the myopic approach suffers more than the CFA approach.

To better understand how the characteristics of the various types of instances affect the CFA and myopic approaches, Table 2.2 provides a percentage breakdown of the total cost into its three component cost terms. For the CFA approach, for all instance types, as the demand increases, the penalty cost's share of the total cost increases while the routing cost's and per-delivery crowdshipper fees' share of the total cost decreases. As demand increases while delivery capacity remains the same, more deliveries are made after the requests' deadlines. Further, because we do not allow crowdshippers to make deliveries beyond their stated availability time window, as demand increases a larger percentage of requests are assigned to the dedicated vehicles as these vehicles will serve requests until all demand is satisfied.

While the *Ulmer* instances contain the same pattern as demand increases (the penalty cost percentage increases while the routing cost percentage and per-delivery crowdshipper fee percentage decrease), Table 2.2 reflects the more stressed nature of these instances. For each of the demand levels, the penalty cost percentage in the *Ulmer* instances is substantially higher than in the other three instance types. For the medium and high demand cases, the penalty costs compose a majority of the total cost, demonstrating the difficulty in meeting the short deadlines as demand increases. While penalty costs increase for both the myopic and CFA approaches, the myopic approach suffers far more from increasing demand.

We observe from Table 2.2 that the CFA approach always incurs a higher

Table 2.2: Percentage breakdown of total cost across the three component cost terms.

		% Routing		% Penalty		% Per-Delivery Fee	
Instance	Demand	Myopic	CFA	Myopic	CFA	Myopic	CFA
	low	92.78	80.66	2.98	4.72	4.24	14.62
Base	med	38.53	77.9	59.58	11.45	1.89	10.65
	high	19.58	70.89	79.4	19.52	1.02	9.59
	low	87.05	81.5	8.96	3.5	3.99	15
1-to-n	med	43.06	78.61	54.78	10.87	2.16	10.52
	high	21.05	72.03	77.84	18.22	1.11	9.75
	low	93.37	80.48	2.38	4.4	4.25	15.12
n-to-1	med	41.27	76.99	56.67	12.33	2.06	10.68
	high	20.73	70.25	78.21	19.87	1.06	9.88
	low	15.5	57.27	83.54	31.57	0.96	11.16
Ulmer	med	8.89	32.68	90.53	60.74	0.58	6.58
	high	6.6	24.24	92.95	70.56	0.45	5.2

percentage of its total costs from the per-delivery crowdshipper fees than the myopic approach. This results from the CFA doing a better job of utilizing the crowdshipper delivery capacity than the myopic approach.

2.6 Conclusions

In this work we present a pickup-and-delivery problem where requests are unknown a priori and arrive stochastically during the day. They are served by a mixed fleet composed by some dedicated vehicles and by some crowdsourced resources unknown a priori that reveal himself stochastically during the day too. We modeled this problem as a Markov Decision Process and we propose a solution method based on a Cost Function Approximation approach to control the assignment of the requests to the vehicles. For what concerns the routing phase we apply a quick constructive heuristic to build the routes. We tested

our approach on different set of instances and on different demand levels, and in all setting our results show that our approach appropriately exploits the capacity of the vehicles, preferring their use to that of dedicated vehicles and guarantee an high level of service to the customer reducing the lateness on the delivery respect to a myopic approach based on an ALNS.

Regarding the management of delivery capacity, we are conducting computational experiments to assess the impact of various fleet mixing consisting of dedicated vehicles, crowdsourced vehicles with uncertainty in both the appearance time and availability duration, as well as the shared economy methods of commuting individuals performing deliveries on commute detours as in Arslan et al. (2019), and in-store customers performing deliveries to online customers as Dayarian and Savelsbergh (2020).

Chapter 3

Analysis of Crowdshipping Alternatives in Stochastic Dynamic Pickup-and-Delivery

Joint work with: ***Demetrio Laganà*****,** ***Jeffrey Ohlmann*** **and** ***Francesca Vocaturo***

Currently editing for submission to an international journal

Abstract: The rapid evolution of collaborative consumption, shared economy and gig work has significantly affected brick-and-mortar business. In this context, a new delivery model not involving stabilized staff has emerged, i.e. crowdshipping. It involves employment of common individuals to deliver goods and management of an e-platform to interact with these individuals commonly referred to as crowdshippers. We focus on an e-platform that aggregates local businesses' products, usually sold in physical stores, and provides delivery services to e-shoppers through dedicated vehicles and crowdshippers. In our case, the crowdshippers are in-store customers that are available to deliver online orders on their way home (consequently, in short time). To decide how to serve these orders with the available delivery capacity, a pickup-and-delivery

problem has to be solved. We model this problem through a Markov decision process with the aim of representing effectively uncertainty in online orders arrivals and in-store customers appearances. Then, we employ cost function approximation within a heuristic framework. The results of computational experiments empirically show the value of this kind of crowdshippers.

3.1 Introduction

Thanks to greater technological advances, last-mile logistics companies are increasingly using crowdsourcing in their delivery service portfolios to provide customers with a better last-mile delivery experience. This phenomenon is particularly evident in large urban areas. Therefore, it is essential to conduct a comprehensive analysis of the factors affecting logistics platforms that provide pickup-and-delivery services by using external resources. This examination is crucial for ensuring the sustainability and competitiveness of the last-mile delivery service. Efficient last-mile logistic services are essential in satisfying dynamic requests with low costs and high service levels. To enhance service quality and efficiency, a logistic service system must be well-equipped with tools and resources such as automated order processing, real-time tracking, and optimized routing strategies. In doing so, service providers can minimize delivery time and improve customer satisfaction. This is accomplished using a combination of owned fleet and external vehicles and drivers. Depending on the level of commitment to the platform provider, drivers can be classified as either dedicated or crowd-sourced. Savelsbergh and Ulmer (2022) provide a practical framework for making this distinction. In the following we will refer to dedicated drivers (or vehicles) as those of a fleet whose service is solely committed to managing requests through a same-day logistics platform, while crowdshippers are those providing an external additional service within a limited time window and without involving an expensive detour from their way

home. Recent literature has focused on same-day delivery services, wherein crowdshippers are modelled as a deterministic resource available at any time of the day. However, little attention is paid to the uncertainty affecting the scheduling of crowdshippers' online availability and to their limited availability for services. We refer the reader to Stoia et al. (2023) for an overview of research on uncertain optimisation problems with crowdshippers.

Recent studies focus on aspects that impact on the unpredictable behavior in the crowd sources. For instance, Nguyen et al. (2023) provide an analysis on the economic and social factors that might affect the behavior of crowdshippers. They study the intentions of crowdshippers to continue participating in last-mile delivery service in Vietnam. Based on the findings of this study, they advise businesses to enhance their benefits and decrease costs in jobs to sustain and further increase participation in last-mile delivery. Le and Ukkusuri (2019) present a study in the USA on the behaviours, expectations, and characteristics of customers, as well as potential crowdshippers. The results of their study could be utilised to enhance the logistics services, attract potential crowdshippers, and devise business plans that correspond to the expectations of customers and crowdshippers. However, despite studies examining the unique operational characteristics of crowdsourced delivery methods in online retail, there remains a research gap in understanding how same-day delivery systems respond to the uncertainty that arises from both service requests and crowdsourcing availability. Our work aims to partially address this gap starting from the study of Stoia et al. (2023) which investigates a pickup-and-delivery problem with time-dependent travel and crowdshipping.

Specifically, we focus on the delivery service connected with a consortium of local businesses. We assume that customers situated in various locations place orders through a shared e-platform and that the platform is accountable for order fulfilment. An order is characterized by a pickup location (a company belonging to the consortium), a drop-off location (the consumer's home or

another specified location), a ready time specifying when pick-up can start, and a soft deadline outlining the consumer's timeliness expectations. As orders are received unpredictably during the day, the manager of the e-platform must determine how to fulfil them. Two types of resources are used to facilitate deliveries: dedicated vehicles and crowdshippers. Crowdsourced resources can manifest in various forms, including some level of commitment. For example, some individuals may guarantee that their service is available at a particular time each day, but they may not ensure their availability for the entire day. The behaviors of the crowdshippers greatly impact the service's effectiveness; therefore, it is crucial to examine various types of crowdsourced resources.

In this study, we consider dedicated vehicles with known availability for day-long shifts. Instead, crowdshippers' appearance times are unknown. The main difference with the work of Stoia et al. (2023) is the typology of individuals that perform the service. Specifically, Stoia et al. (2023) consider crowdshippers with any starting location (e.g., work office or home). Here we consider in-store customers (see Dayarian and Savelsbergh (2020)). An in-store customer can be defined as an individual going to one of the store of the consortium for own shopping, but who is willing to fulfill orders of online shoppers on the way home (or, more in general, to the next destination). Therefore, the starting location of a crowdshipper in this study always coincides with an pickup location. In addition, for the nature of our crowdshippers, we assume that the availability duration (declared at appearance in the store) is shorter than the one assumed by Stoia et al. (2023). The rationale behind this study differs as well.

In fact, our aim is to examine the economic effects of using a crowdsourced vehicle fleet under uncertain conditions on the quality of service provided. Our preliminary analyses aim to determine the best setting in terms of number of crowdshippers required and their time windows'capacity to reduce the number of dedicated vehicles. We aim to demonstrate that using a mixed vehicle fleet is preferable, as compared to a standard fleet only composed of dedicated ve-

hicles, under certain conditions. The impact can be measured by comparing the marginal costs of a fleet consisting only of dedicated vehicles to that of a mixed fleet including both dedicated vehicles and crowdsourced resources. Additionally, the difference in delay times in comparable situations provides another indicator.

The remainder of this article is organized as follows. A formal description of the problem is given in Section 3.2. Section 3.3 illustrates the solution methodology. In Section 3.4 data sets are described, while in Section 3.5 the results of a preliminary experimental phase are discussed. Conclusions follow in Section 3.6.

3.2 Problem Description and Model Formulation

The problem we refer in our study is a stochastic dynamic pickup-and-delivery problem with time-dependent travel and in-store shoppers as crowdsourced capacity. It is a variant of the problem reported in Chapter 1.
Also in this case we assume that an e-platform receives the customer requests randomly throughout the day. A customer request r corresponds to a tuple of information: the pickup location (o_r), the delivery location (d_r), the time which the request will be ready (e_r), and a soft deadline representing the latest time it should delivered (l_r). If the vehicle handling customer request r arrives at location o_r before time e_r, the vehicle must wait until e_r before executing the request pick up. If the vehicle handling customer request r delivers the request at location d_r after l_r, it incurs a lateness charge.

To provide delivery service, we assume there is a fleet of dedicated vehicles, $\mathcal{V} = \{1, \ldots, V\}$, and a set of crowdshippers, $\mathcal{G} = \{1, \ldots, G\}$. Each dedicated vehicle $v \in \mathcal{V}$ is available over a known shift starting and ending at the depot, $n_v^a = n_v^b = 0$ for $v \in \mathcal{V}$. For example, a dedicated vehicle v available for the entire day would have a shift from time $a_v = 0$ to time $b_v = \hat{T} > T$, where

Table 3.1: Overview of notation.

$\mathcal{T}$	time horizon of request arrivals, $t \in \mathcal{T} = [0, 1, \ldots, T]$
$\mathcal{R}_k^p$	set of in-process requests at epoch k
$\mathcal{R}_k^o$	set of outstanding requests at epoch k
$\mathcal{R}_k$	set of active requests at epoch k, $\mathcal{R}_k = \mathcal{R}_k^p \cup \mathcal{R}_k^o$
$\mathcal{U}_k$	set of newly-arriving requests to the system at epoch k
o_r	pickup location of request r
$\mathcal{O}$	set of pickup locations (stores)
d_r	delivery location of request r
$\mathcal{D}$	set of delivery locations (customers)
e_r	ready time of request r
l_r	deadline of request r
$\mathcal{V}_k$	set of dedicated vehicles available at epoch k
$\mathcal{G}_k$	set of in-store shoppers available at epoch k
$\mathcal{M}_k$	entire pool of vehicles available at epoch k, $\mathcal{M}_k = \mathcal{V}_k \cup \mathcal{G}_k$
n_m^a	starting location of vehicle $m \in \mathcal{O}$
n_m^b	ending location of vehicle m
a_m	appearance time of vehicle m at location n_m^a
b_m	time by which vehicle m must arrive at location n_m^b
θ_m	routing plan of vehicle m
f_m	arrival time of vehicle m at the first location in θ_m
w_m	departure time of vehicle m from the first location in θ_m
s_k	pre-decision state
s_k^x	post-decision state
$\mathcal{X}(s_k)$	set of possible actions in s_k, $x \in \mathcal{X}(s_k)$
$c(s_k, x)$	cost incurred by action x when system in state s_k at epoch k
ρ	per-delivery fee (\$)
μ_1	travel time multiplier (\$ per minute)
μ_2	lateness multiplier (\$ per minute)
ω_k	set of new customer requests arriving at epoch k
γ_k	set of new crowdshippers appearing at epoch k
λ	cost multiplier for a vehicle's remaining delivery time
η	wait time multiplier for a vehicle's remaining delivery time
$\tau(i, j, t)$	travel time between locations i and j when departing at time t

$\hat{T}$ is sufficiently large enough to ensure delivery of all requests. Each in-store shopping $g \in \mathcal{G}$ is available to service requests during a time window $[a_g, b_g]$ that is unknown to the dispatcher a priori. Upon appearing to the platform at time a_g and declaring their location n_g^a, a crowdshipper g communicates their availability to service requests subject to requirement that they can arrive at location n_g^b by time b_g. To model the situation in which an in-store customer serves as a crowdshipper, we assume that the appearance location is a store location, $n_g^a \in \mathcal{O}$.

The objective is to minimize the cost of fulfilling customer requests over an operating day comparing the case of a fleet composed by dedicated vehicles and the case composed by dedicated vehicles and in-store shoppers. To measure the cost of providing service, we minimize the sum of three components in the case of mixed-fleet: (1) the total travel cost of the dedicated vehicles and crowdshippers (as function of travel time), (2) the total lateness charge (as a function of the amount of time customer requests are delivered after their soft deadlines), and (3) the total per-delivery fees paid to crowdshippers. Note that we assume a fixed per-delivery fee of ρ for all requests served by a crowdshipper and include the travel cost of the crowdshippers in the objective function to represent the variable fees paid by the e-platform to the crowdshippers to account for differences in the travel demands of requests due to their relative pickup-and-delivery locations. In the case of a dedicated fleet of vehicles obviously we take into account just the total travel cost and the total lateness charge.

To represent the stochastic nature of the arrival of customer requests and the appearance of crowdshippers, we model the problem as a Markov decision process (MDP) over finite, discrete-time horizon. For an in-depth description of the model, see Section 1.3.

3.3 Solution Methodology

The applied solution methodology is the same of the one proposed in Chapter 1, where we describe in detail how DRACE works. For an in-depth and detailed reading of the method, please refer to Section 1.4.

3.4 Description of Data Sets

In this section, we outline the construction of the data sets which we apply our solution methodology. In Section 3.4.1, we describe the structure of the customer request data in our various instance types. We outline the delivery capacity for our instances, particularly describing crowdshippers in Section 3.4.2. In Section 3.4.3, we explain how we model time- and location-dependent travel times in our instances.

3.4.1 Customer Request Data

In this paper, to assess the impact that different vehicle fleets can have in terms of cost-effectiveness and guaranteed service level, the instances related to the arrival of customer requests we refer to are the non-mixed stationary instances presented by Stoia et al. (2023) in Section 1.5.1 in which, however, the possibility of having short and long deadline requests is relaxed. In this case we tested the instances considering 100% of the requests as non-urgent. The long-deadline requests are ready 40 minutes after request arrival and have a soft deadline 120 minutes after request arrival and the considered arrival rate in which requests arrive according to a nonhomogeneous Poisson process over a ten-hour day is 22.5 requests/hour.

3.4.2 Delivery Capacity

In this work we compare different delivery workforces. We consider the case of a fleet composed by just dedicated vehicles available over the entire time horizon and we carry out tests by varying the number of available vehicles, and a mixed fleet of vehicles composed by dedicated vehicles and crowdshippers. For what concerns the crowdshippers we consider that their appearance times realizes according to a non-homogeneous Poisson process. Each crowdshipper appears to the system at a randomly-generated origin and destination. In this case, differently from the paper of Stoia et al. (2023) we consider crowdshippers who are in-store shoppers as in Arslan et al. (2019), in fact their origin location corresponds to one of the 248 pickup locations of the instances presented by Stoia et al. (2023). We also consider the possibility of having different time windows availability (TW). Furthermore, considering the focus of the present work, we would like to exploit the possibility of varying the available crowdsourced capacity considering a probability value P that corresponds to the ratio of the crowdsourced vehicles respect to the demand level that should be served in a day.

3.4.3 Time-Dependent Travel Time

Also in this work we consider the opportunity of having time-dependent travel times instead of average and deterministic travel times, due to the fact that we proved its effectiveness in Section 1.6.4. We modeled time dependency in the same way we did in Chapter 1 in Section 1.5.3.

3.5 Computational Results

The objective of this study is to examine the economic benefits of employing a mixed-fleet of vehicles consisting of both owned and crowdsourced capacity, as opposed to a dedicated fleet of vehicles. Additionally, we aim to determine

the optimal size of the mixed fleet, based on its various features. First, we compute the realized cost of meeting daily requests using a range of different-sized dedicated vehicle fleets. This initial assessment establishes the minimum number of dedicated vehicles necessary for ensuring feasible request fulfillment, as well as the maximum number of valuable dedicated vehicles. Second, we calculate the total cost of a mixed fleet of vehicles, comprised of both dedicated vehicles and crowdsourcing resources. For the crowdsourced resources, we consider time windows with an availability of 20, 30, and 60 minutes, added to the travel time required to reach the origin and destination of each vehicle. Additionally, we evaluate the probability value, P, with values from the set $\{5\%, 10\%, 20\%, 30\%\}$. The parameters used in DRACE to evaluate the total cost have been the same as those used in Chapter 1. In this study, we do not account for any fixed costs associated with owning and maintaining dedicated vehicles. Instead, the analysis only considers the operational expenses of various fleets.

3.5.1 Dedicated vehicles VS Mixed-Fleet of vehicles

Preliminary tests were carried out on a standard fleet consisting of dedicated vehicles, where we varied its size to determine the optimal fleet size that minimises the total cost per day calculated by DRACE. The tests we conducted are based on the average values calculated from 10 different demand instances. The results obtained indicate that a fleet of 4 dedicated vehicles is insufficient to meet the demand for all 10 instances related to different working day demand scenarios. This emerges from Table 3.2, which highlights the infeasibility for Instances 1, 3, and 6. The minimum number of dedicated vehicles that guarantees feasibility in the fulfillment of the demand is 5, but the fleet that minimizes the total fulfillment cost per day is composed by 7 dedicated vehicles as it is shown in Figure 3.1. This value corresponds also to the minimization of the average total delay per day. The average costs per day in the

mixed fleet case were calculated using five different occasional vehicle arrival paths and accounting for four dedicated vehicles. In this scenario, Tables 3.3, 3.4, and 3.5 display the marginal costs of adding another dedicated vehicle to the fleet instead of using the mixed fleet. Each value has been computed as follows:

$$MarginalCost = AvgCost_{mixedfleet} - AvgCost_{dedicatedfleet} \tag{3.1}$$

Looking at Table 3.3, it can be inferred that using a fleet consisting of only 5 dedicated vehicles is more advantageous than using a mixed fleet comprising of 4 dedicated vehicles and a varying number of in-store customers. This is only feasible when there are a limited number of crowdsourced vehicles and each has a limited time window of 20 minutes, and also if the cost of owning and maintaining a fixed vehicle is consistently below 255 per day. For all other possible combinations in this table, negative marginal costs arise, indicating that using crowdsourced capacity is valuable and preferable. The value increases with the number of in-store customers and their time availability, even though the lowest (negative) marginal cost arises in the case where $TW = 20$ minutes and $P = 30\%$. However, when we consider the same value of P, but greater values of TW, the negative marginal cost reduces. The explanation for this is related to the fact that in these last two cases, we are considering a capacity that is not necessarily exploited due to the demand level we must serve. In Tables 3.4 and 3.5, it is evident that the benefit of an additional dedicated vehicle is highest when the number of in-store customers is low ($P = 5\%$). Nevertheless, this benefit reduces as the availability of time windows increases. For values of $P = 20\%$ or $P = 30\%$, marginal cost is consistently negative. Given that 7 is the optimal number of dedicated vehicles for minimizing total costs in a standard fleet consisting of dedicated vehicles, these negative marginal costs indicate that we could improve upon the best case by reducing the dedicated fleet to 4 and considering the opportunity to have a sufficient number of crowdsourced resources, even if their time availability is limited. Similar considerations can

Table 3.2: Total collected cost ($\$/day$) and total collected delay (min/day) for a dedicated fleet of 4 vehicles.

Instance	Tot cost	Tot min delay
1	Unfeasible	
2	53049	10192
3	Unfeasible	
4	26177	4828
5	14451	2373
6	Unfeasible	
7	44640	8494
8	36705	6800
9	4168	371
10	5485	610

also be made when referring to the average daily minutes of delay gathered on request deliveries, reported in Tables 3.6, 3.7 and 3.8, instead of using the marginal total cost as a comparison term.

Figure 3.1: Avg Total Cost per day and Avg Delay per day for different fleet sizes of dedicated vehicles

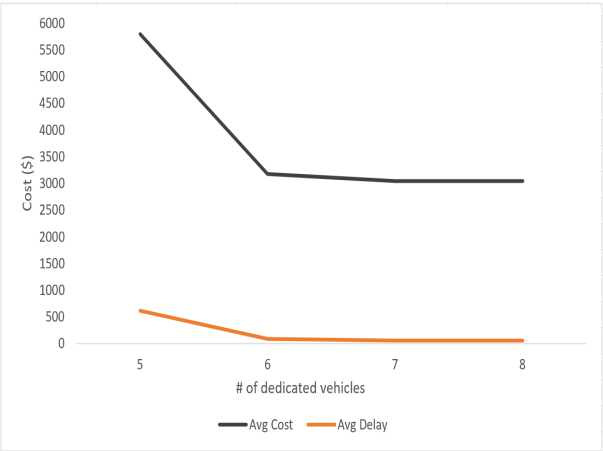


Table 3.3: Marginal cost per day ($) of considering a dedicated fleet composed by 5 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	255	-906	-1250
10 %	-2504	-2669	-2829
20 %	-3433	-3366	-3187
30 %	-3626	-3496	-3373

Table 3.4: Marginal cost per day ($) of considering a dedicated fleet composed by 6 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	2880	1720	1376
10 %	122	-43	-203
20 %	-807	-741	-561
30 %	-1000	-870	-747

Table 3.5: Marginal cost per day ($) of considering a dedicated fleet composed by 7 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	3011	1851	1506
10 %	252	87	-72
20 %	-677	-610	-431
30 %	-870	-740	-617

Table 3.6: Marginal reduction in delay per day (min) considering a dedicated fleet composed by 5 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	119	-107	-205
10 %	-408	-457	-540
20 %	-577	-589	-607
30 %	-595	-604	-612

Table 3.7: Marginal reduction in delay per day (min) considering a dedicated fleet composed by 6 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	650	424	326
10 %	123	74	-9
20 %	-46	-58	-76
30 %	-64	-73	-81

Table 3.8: Marginal reduction in delay per day (min) considering a dedicated fleet composed by 7 dedicated vehicles and a mixed fleet composed by 4 dedicated vehicles and a variable crowdsourced capacity.

	TW (min)		
P	20	30	60
5 %	679	453	355
10 %	151	103	20
20 %	-17	-29	-47
30 %	-35	-44	-52

3.5.2 Mixed-Fleet Analysis

The second objective of this analysis is to explore the sensitivity of the aspects related to occasional vehicle usage, vehicle number (P), and time availability (TW) towards ensuring economic savings.

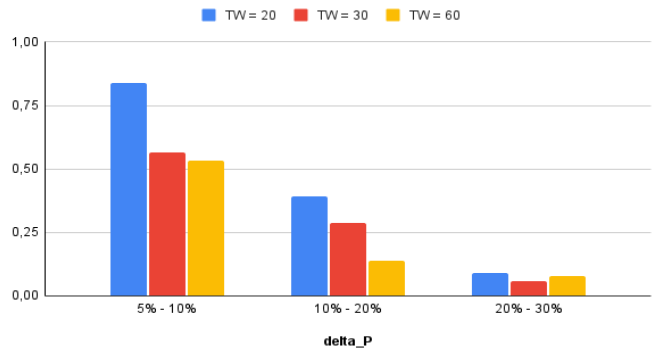
Figure 3.2 illustrates the percentage reduction in total cost as we increase the value of P for various configurations of time window availability.

The values presented are computed by fixing the TW for each pair P_i, P_j:

$$cost_{\Delta P_{ij}} = (cost_{max\{P_i,P_j\}} - cost_{min\{P_i,P_j\}})/cost_{min\{P_i,P_j\}} \tag{3.2}$$

The greatest saving is achieved by increasing from $P = 5\%$ to $P = 10\%$ for narrower time windows ($TW = 20\ min$), equating to 84%. This outcome is justifiable since we are examining the most stressed delivery system (due to limited in-store customers and restricted time window availability). Increasing the values of P reduces the advantage, as demonstrated across all time windows. However, the impact is minimal due to the low arrival rate of requests per hour in these tests. Values of P at 20% or 30% have negligible effect, as the extra capacity is unnecessary.

Figure 3.2: Percent reduction in Avg Total Cost per day varying the value of P.

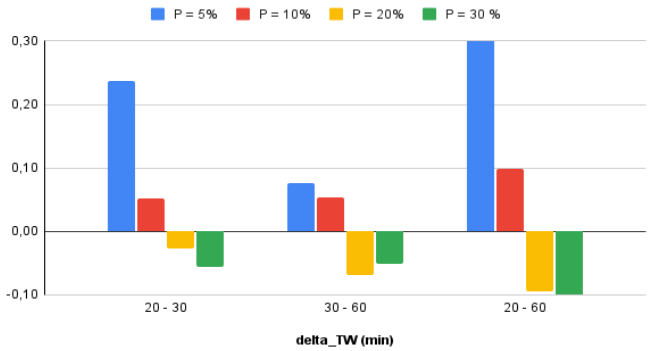


In Figure 3.3, we investigate the percentage decrease in average total cost per day as we increase the availability of time windows for in-store customers, for each value of P.
The values are calculated based on a formula similar to Formula 3.2. For each pair of TW_i and TW_j, we determine the percentage reduction in costs as:

$$cost_{\Delta TW_{ij}} = (cost_{max\{TW_i,TW_j\}} - cost_{min\{TW_i,TW_j\}})/cost_{min\{TW_i,TW_j\}} \quad (3.3)$$

It is evident from the comparisons that longer time windows offer value for small P values. When comparing the scenarios of $TW = 20$ and $TW = 60$ for $P = 5\%$, we observed a reduction of total costs by 33%. However, this advantage decreases as P value increases, and the lowest value is achieved for $P = 30\%$ and a variation of TW from 20 minutes to 60 minutes. There are two reasons that may explain this phenomenon. First, the rate of request arrivals is low, therefore having a lot of capacity may not be beneficial if it is not needed. Second, there is a structural issue related to DRACE's use of the same parameter setting as in the paper Stoia et al. (2023), where the availabilities of crowdshippers were longer.

Figure 3.3: Percent reduction in Avg Total Cost per day varying the value of TW.



3.6 Conclusions

In this research, we have presented an initial analysis that seeks to illustrate the benefits of employing a combination of dedicated and crowdsourced vehicles for the given problem. We specifically focus on the use of such a mixed fleet for in-store customers. The preliminary results are reassuring, and the study is noteworthy since, to the best of our knowledge, there are no similar works in the literature. In the future, we will extend the analysis performed on smaller instances to the other large and medium instances detailed in Stoia et al. (2023), alongside a generalisation of the considered crowd-shipping methods, as differentiated by Savelsbergh and Ulmer (2022), to deliver a comprehensive analysis.

Conclusions

This thesis work focuses on the study of pickup-and-delivery problems affected by uncertainty and addressing them through the use of innovative paradigms based on the sharing economy. The importance of last-mile delivery firms leveraging alternative delivery methods to stay competitive and overcome the challenges posed by the logistics industry's daily volume increases has become evident.

The emergence of e-commerce due to technological advances in recent years has led to this significant growth. It is convenient to purchase a variety of products from home and anticipate swift delivery. This inclination aligns with the expectations of consumers with high demands. Nonetheless, it can lead to the extinction of small independent shops, which are unable to offer equivalent provisions as e-commerce big companies. To address this issue, several survival strategies can be employed, one of which involves the formation of a consortium. This strategy involves delegating the delivery of products to final customers to a third-party logistics provider via an online platform, rather than managing it directly. The logistics provider receives orders from the final customers and takes care of pickup-and-delivery.

Instead of only having a fleet of dedicated vehicles, the platform for a consortium of local businesses can also take advantage of crowdsourced vehicles, i.e., non-employees who offer their willingness to make a few deliveries in exchange for a fair remuneration. In addition to the uncertainty in the timing and location of customer requests, this introduces uncertainty in the deliv-

ery capacity of crowdshippers. When a crowdshipper is available and for how long may be uncertain to the delivery platform. Further, we also study time-dependent travel times (e.g., resulting from traffic patterns) and their impact on our pickup-and-delivery setting.

In Chapter 1, we formally define and model of the problem.To accommodate the uncertain nature of the information throughout the day, we utilised a Markov decision process framework. We apply a destroy-and-repair heuristic, incorporating a parameterized cost function approximation (CFA) to address the fundamental challenge of the varying and uncertain fleet capacity. To assess the proposed approach's efficacy in cost and service level, we benchmark it with a myopic policy based on adaptive large neighborhood search (ALNS), a traditional approach for tackling deterministic pickup-and-delivery issues. Our destroy-and-repair approach outperforms the myopic ALNS with respect to all metrics.

To evaluate our CFA-based destroy-and-repair heuristic, we develop a collection of data sets in which requests with varying deadline lengths arrive according to a nonhomogeneous Poisson process reflecting real-world cases. In addition, we create a data sets that examine the impact of demand management techniques that affect the arrival pattern of customer requests. The consideration of these data sets allows us to gain a better understanding on the effect of promotional marketing efforts on delivery efficiency.

Chapter 2 refers to a brief article providing a bridge to Chapter 3. Chapter 2 generalizes the proposed approach of Chapter 1 to various kinds of crowdshipping, such as in-store customers and individual commuters, and examines the potential of these alternative capacities. In Chapter 3, we conduct an economic analysis of the value of using a mixed fleet of vehicles compared to using a fleet consisting exclusively of dedicated vehicles. We examine a specific type of crowdshipper, which can be pictured as an in-store customer who accepts an offer to conduct convenient deliveries upon their departure from the store. In

contrast to the generic crowdshippers of Chapter 1, in-store customers have an origin in one of the pickup locations and typically have limited time availability. Our tests determine the level of in-store customer participation to reduce the fleet of dedicated vehicles, lower operating, and maintain a higher standard of service.

As part of our future work, we will conduct an economic analysis to determine the value of crowdsourced vehicles for delivery capacity and explore methods to encourage their participation in the service. This analysis will address an existing gap in the literature related to the challenge of modeling and predicting the behavior of vehicles without formal commitments to the service provider.

Bibliography

Aliaa Alnaggar, Fatma Gzara, and James H. Bookbinder. Crowdsourced delivery: A review of platforms and academic literature. *Omega*, 98, 2021a.

Aliaa Alnaggar, Fatma Gzara, and James H. Bookbinder. Crowdsourced delivery: A review of platforms and academic literature. *Omega*, 98, 2021b.

Claudia Archetti, Francesca Guerriero, and Giusy Macrina. The online vehicle routing problem with occasional drivers. *Computers & Operations Research*, 127, 2021.

Alp M Arslan, Niels Agatz, Leo Kroon, and Rob Zuidwijk. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 53(1):222–235, 2019.

Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231, 2021.

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

Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

Dimitri P. Bertsekas. Dynamic programming and optimal control. *Athena Scientific*, 1, 2005.

BusineesWire. New research finds 65 Available online at: `https://www.businesswire.com/`.

Xinwei Chen, Marlin W Ulmer, and Barrett W Thomas. Deep Q-learning for same-day delivery with vehicles and drones. *European Journal of Operational Research*, 298(3):939–952, 2022.

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

Gianpaolo Ghiani, Andrea Manni, and Emanuele Manni. A scalable anticipatory policy for the dynamic pickup and delivery problem. *Computers & Operations Research*, 147, 2022.

Tho V. Le and Satish V. Ukkusuri. Crowd-shipping services for last mile delivery: Analysis from American survey data. *Transportation Research Interdisciplinary Perspectives*, 1:100008, 2019.

Simona Mancini and Margaretha Gansterer. Bundle generation for last-mile delivery with occasional drivers. *Omega*, 108, 2022.

Diana Mattioli. Not being Amazon is a selling point for these e-commerce players. *The Wall Street Journal*, (June 16), 2021. https://www.wsj.com/articles/not-being-amazon-is-a-selling-point-for-these-companies-11623835802.

Snežana Mitrović-Minić and Gilbert Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655, 2004.

Snežana Mitrović-Minić, Ramesh Krishnamurti, and Gilbert Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685, 2004.

Nguyet Nguyen, Thi Hoang Ha Tran, Thi Thuy Duong Luu, and Tuan Duong Vu. Crowdshippers' intentions to continue participating in last-mile delivery: A study in Vietnam. *The Asian Journal of Shipping and Logistics*, 39(3):48–56, 2023.

Warren B. Powell, Hugo P. Simao, and Belgacem Bouzaiene-Ayari. Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1:237—-284, 2012.

Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

Ulrike Ritzinger, Jakob Puchinger, and Richard F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

SAP. Last mile logistics: Solutions for a changing world. Available online at: `https://www.sap.com/`.

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

M. Schilde, K.F. Doerner, and R.F. Hartl. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, 38(12):1719–1730, 2011.

Michael Schilde, Karl F Doerner, and Richard F Hartl. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18–30, 2014.

Albert H Schrotenboer, Michiel A.J. Uit het Broek, Paul Buijs, and Marlin W Ulmer. Fighting the e-commerce giants: Efficient routing and effective consolidation for local delivery platforms. Technical report, arXiv preprint arXiv:2108.12608, 2021.

Marco Silva, João Pedro Pedroso, and Ana Viana. Deep reinforcement learning for stochastic last-mile delivery with crowdshipping. *EURO Journal on Transportation and Logistics*, 12, 2023.

André LS Souza, Marcella Bernardo, Puca HV Penna, Jürgen Pannek, and Marcone JF Souza. Bi-objective optimization model for the heterogeneous dynamic dial-a-ride problem with no rejects. *Optimization Letters*, 16(1):355–374, 2022.

Zachary Steever, Mark Karwan, and Chase Murray. Dynamic courier routing for a food delivery service. *Computers & Operations Research*, 107:173–188, 2019.

Sara Stoia, Demetrio Laganà, and Jeffrey W. Ohlmann. Dynamic pickup-and-delivery for collaborative platforms with time-dependent travel and crowdshipping. *Submitted article*, 2023.

Peng Sun, Lucas P Veelenturf, Mike Hewitt, and Tom Van Woensel. The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 116:1–24, 2018.

Marlin W Ulmer and Martin Savelsbergh. Workforce scheduling in the era of crowd-sourced delivery. *Transportation Science*, 54(4):1113–1133, 2020.

Marlin W Ulmer and Barrett W Thomas. Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, 72(4):475–505, 2018.

Marlin W Ulmer, Barrett W Thomas, Ann Melissa Campbell, and Nicholas Woyak. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, 55(1):75–100, 2021.

Liana Van der Hagen, T. R. Visser, and R. Spliet. The pickup and delivery problem with time windows: an adaptive large neighborhood search heuristic. Technical report, Erasmus University Rotterdam, 2017.

Thibaut Vidal, Rafael Martinelli, Tuan Anh Pham, and Minh Hoàng Hà. Arc routing with time-dependent travel times and paths. *Transportation Science*, 55(3):706–724, 2021.

Stefan Vonolfen and Michael Affenzeller. Distribution of waiting time for dynamic pickup and delivery problems. *Annals of Operations Research*, 236:359–382, 2016.

Setyo T. Windras Mara, Rachmadi Norcahyo, Panca Jodiawan, Luluk Lusiantoro, and Achmad P. Rifai. A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, 146, 2022.

Zhihai Xiang, Chengbin Chu, and Haoxun Chen. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, 185(2):534–551, 2008.

List of Figures

List of Tables

Acknowledgements

My sincerest gratitude goes to the professors who supported and supervised me throughout my doctoral journey: Prof. Laganà, Prof. Ohlmann and Prof. Vocaturo. Your continued support, advice, and willingness to listen to me and to answer my questions have been invaluable. Thank you for inspiring me with your passion, for sharing your knowledge and for fanning the flame of curiosity in me. Your contribution is not only limited to my academic education, but has also shaped the way I approach research and life. I hope that this small gesture of appreciation may in some way reflect the admiration and the respect I feel for you. Thank you again for all that you have done for me.

A special thanks also go to the reviewers of the work, whose valuable suggestions contributed to its improvement.

Ringraziamenti

La mia più sincera gratitudine va ai docenti che mi hanno affiancato e supervisionato durante il mio percorso: il Prof. Laganà, il Prof. Ohlmann e la Prof.ssa Vocaturo.
Il Vostro continuo sostegno, i vostri consigli e il vostro supporto, la vostra disponibilità nell'ascoltarmi e nel rispondere alle mie domande sono stati inestimabili. Grazie per avermi ispirato con la vostra passione, per aver condiviso il vostro sapere e alimentato in me la fiammella della curiosità. Il vostro contributo non si limita solo alla mia formazione accademica, ma ha anche plasmato il mio modo di affrontare la ricerca e la vita. Spero che questo piccolo gesto di stima possa in qualche modo riflettere l'ammirazione e il rispetto che provo nei vostri confronti. Grazie ancora per tutto quello che avete fatto per me.

Un sentito e doveroso ringraziamento va anche ai revisori del lavoro di tesi che, grazie ai loro preziosi suggerimenti, hanno contribuito a migliorarlo.