Optimization under Uncertainty for E-retail Distribution: From Suppliers to the Last Mile

by

Aliaa Alnaggar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Management Sciences

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Dr. Ming Hu Distinguished Professor of Business Operations and Analytics, Rotman School of Management, University of Toronto
Supervisor(s):	Dr. Fatma Gzara Professor, Management Sciences, University of Waterloo
	Dr. James Bookbinder Professor, Management Sciences, University of Waterloo
Internal Member:	Dr. Houra Mahmoudzadeh Assistant Professor, Management Sciences, University of Waterloo
	Dr. Saeed Ghadimi Assistant Professor, Management Sciences, University of Waterloo
Internal-External Member:	Dr. Liping Fu Professor, Civil and Environmental Engineering, University of Waterloo

Author's Declaration

Statement of Contributions

I am the sole author of Chapters 1, 5, and 6.

Chapters 2 and 3 of this thesis are published in two research papers. Chapter 2 is published in Alnaggar et al. (2020), and Chapter 3 is published in Alnaggar et al. (2019). Both of these publications were co-authored by myself and my supervisors, Dr. Fatma Gzara and Dr. James Bookbinder.

Chapter 4 of this thesis is a submitted paper under review, co-authored by myself and my supervisors, Dr. Fatma Gzara and Dr. James Bookbinder.

Abstract

This thesis examines problems faced in the distribution management of e-retailers, in different stages of the supply chain, while accounting for sources of uncertainty. The first problem studies distribution planning, under stochastic customer demand, in a transshipment network. To decide on a transportation schedule that minimizes transportation, inventory and outsourcing costs, the problem is formulated as a two-stage stochastic programming model with recourse. Computational experiments demonstrate the cost-effectiveness of distribution plans generated while considering uncertainty, and provide insights on conditions under which the proposed model achieves significant cost savings.

We then focus our attention on a later phase in the supply chain: last-mile same-day delivery. We specifically study *crowdsourced delivery*, a new delivery system where freelance drivers deliver packages to customers with their own cars. We provide a comprehensive review of this system in terms of academic literature and industry practice. We present a classification of industry platforms based on their matching mechanisms, target markets, and compensation schemes. We also identify new challenges that this delivery system brings about, and highlight open research questions. We then investigate two important research questions faced by crowdsourced delivery platforms.

The second problem in this thesis examines the question of balancing driver capacity and demand in crowdsourced delivery systems when there is randomness in supply and demand. We propose models and test the use of heatmaps as a balancing tool for directing drivers to regions with shortage, with an increased likelihood, but not a guarantee, of a revenue-producing order match. We develop an MDP model to sequentially select matching and heatmap decisions that maximize demand fulfillment. The model is solved using a stochastic look-ahead policy, based on approximate dynamic programming. Computational experiments on a real-world dataset demonstrate the value of heatmaps, and factors that impact the effectiveness of heatmaps in improving demand fulfillment.

The third problem studies the integration of driver welfare considerations within a platform's dynamic matching decisions. This addresses the common criticism of the lack of protection for workers in the sharing economy, by proposing compensation guarantees to drivers, while maintaining the work hour flexibility of the sharing economy. We propose and model three types of compensation guarantees, either utilization-based or wage-based. We formulate an MDP model, then utilize value function approximation to efficiently solve the problem. Computational experiments are presented to assess the proposed solution approach and evaluate the impact of the different types of guarantees on both the platform and the drivers.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Professor Fatma Gzara and Professor Jim Bookbinder. Their expertise, guidance, support and encouragement have greatly improved the quality of my research and made my PhD journey a truly enjoyable and fulfilling experience. I am honored to have worked with them for several years; they have been both a professional and personal inspiration to me.

I would also like to thank the members of my examination committee Professor Ming Hu, Professor Liping Fu, Professor Houra Mahmoudzadeh, and Professor Saeed Ghadimi for their helpful feedback on my thesis.

My PhD experience wouldn't have been the same without the wonderful friends I met at the Management Sciences department. I appreciate the many discussions we had and the beautiful memories we created. I would especially like to thank Cynthia, Remziye, Hanan, Haoran, Dani, Kiefer, Rebecca, Paulo, Gita and Gohram.

Lastly and most importantly, I would like to thank my family for their endless love and encouragement. Thank you to my amazing parents who inspired my love of knowledge and taught me to chase my dreams and persevere. Thank you to my loving, kind and supportive husband, Islam Barghash, who is always there for me, and who has been my source of strength in tough times. Finally, thank you to my two wonderful children, Sami and Jenna, who brighten my day, everyday, and who have been incredibly patient and understanding during the pandemic.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

E-retailing is a global market that has experienced enormous growth in the last decade. Today, many retailers that were once only offering bricks-and-mortar stores are adopting omnichannel models by expanding their services to online shopping. This growth comes with higher expectations from customers; they are increasingly expecting shorter delivery times without a hefty delivery charge. To create efficient and responsive distribution plans, e-retailers need to account for the different sources of uncertainty that affect their distribution activities throughout the various phases of the supply chain. This thesis examines three distribution management problems, faced by e-retailers, that aim to optimize their distribution planning activities under uncertainty.

Chapter 2 studies distribution planning between suppliers and distribution centers, where we consider a transshipment network under stochastic customer demand, applicable to many three-tier supply chain networks. We study this problem from the perspective of a third-party logistics provider (3PL) that is outsourced to handle the logistics needs of its clients; the 3PL uses a consolidation center to achieve transportation cost savings. We formulate a two-stage stochastic programming model with recourse that aims to minimize the sum of transportation cost, expected inventory holding cost and expected outsourcing cost. The recourse variables ensure that the problem is feasible regardless of the realization of demand, by allowing the option of using a spot market carrier if demand exceeds capacity. We use Sample Average Approximation (SAA) to solve the problem and show that it results in reasonable optimality gaps for problem instances of different sizes. We conduct extensive testing to evaluate the benefits of the proposed stochastic model compared to its deterministic counterpart. Computational experiments provide managerial insight into the robustness and cost-effectiveness of the distribution plans of the proposed stochastic model, and the conditions under which the model achieves significant distribution cost savings.

Chapter 3 onward, we shift our attention to a later phase in the supply chain, namely, last-mile same-day delivery from distribution centers or stores to end customers. We particularly study an emergent delivery system, crowdsourced delivery, which relies on individuals completing last-mile delivery tasks with their own cars. In Chapter 3, we analyze the current industry status of this delivery system and provide a classification of available platforms based on their matching mechanisms, target markets and compensation schemes. We review the operations research (OR) literature addressing this topic and assess the applicability of assumptions to real-world applications. We also compare the management decisions within crowdsourced delivery systems to well-studied OR problems in the literature, and pinpoint new challenges that arise in the context of crowdsourced delivery. In the following two chapters, we investigate two important managerial questions faced by crowdsourced delivery platforms.

Chapter 4 studies the problem of balancing driver capacity and demand in crowdsourced delivery systems, when there is randomness in both driver supply and demand. Since crowdsourced drivers are independent contractors, their movement is not directly managed. We investigate the use of heatmaps as a balancing tool for directing drivers to regions with shortage, with an increased likelihood, but not a guarantee, of a revenue-producing assignment. This creates a generalized framework for managing the movement of crowdsourced drivers, without a direct assumption on their compensation scheme. We develop a Markov decision process (MDP) model to sequentially select matching and heatmap decisions that maximize demand fulfillment. The model is solved using a stochastic lookahead policy, based on approximate dynamic programming. We also propose a simple policy and an upper-bound problem that assumes drivers are directly managed. We find that optimized heatmaps induce driver repositioning to areas of shortage and improve demand fulfillment up to the level where drivers are managed directly, when the number of drivers is higher than demand. The effectiveness of heatmap is most notable when the network is imbalanced, where the demand inflow to some nodes is significantly higher/lower than the outflow.

Chapter 5 examines the integration of driver welfare considerations in a platform's dynamic matching decisions. Crowdsourced delivery and other sharing economy platforms typically compensate workers per task and provide little guarantees for their earning amount while they are ready to work. We study the problem of designing dynamic matching policies, in a crowdsourced delivery system, that guarantee a particular level of utilization or earning for active workers, while maintaining the inherent work hour flexibility promoted by the sharing economy. We propose, model, and test three types of guarantees, that are either utilization-based or wage-based. To capture the dynamic and stochastic nature of crowdsourced delivery operations, we propose an MDP model. We utilize approximate dynamic programming techniques to efficiently obtain good solutions, given the high dimensionality of the solution space. In particular, we use value function approximation to obtain good estimates of the value of post-decision states, using forward simulation. We conduct extensive computational testing to assess the performance of the proposed solution methodology. We also compare the different types of guarantee policies, and assess their impact on the platform and the drivers, relative to the base policy, which models the no-guarantee case.

Chapter 6 concludes the thesis and discusses opportunities for future research.

Chapter 2

Distribution Planning with Random Demand and Recourse in a Transshipment Network[1]

2.1 Introduction

In many supply chain networks, third party logistics providers (3PLs) are employed to handle the distribution needs within the supply chain. The 3PL faces the challenging task of coordinating these distribution activities between suppliers and customers, possibly through the use of intermediate facilities, so as to create a lean cost-efficient supply chain, while ensuring timely customer deliveries. Third party logistics is a fast growing market; in 2016, it had an estimated worldwide market size of 802.2 billion US dollars, 38% of which is in the Asia Pacific region, 25% in North America, 21.5% in Europe (Langley, 2017). With this growth comes increased competition which further necessitates that the 3PL create leaner logistics solutions, in order to survive in a growingly contested market. In recent years, there has been an increasing trend in businesses outsourcing their transportation needs to 3PL's to focus on their core business competencies. The various players within the supply chain expect the 3PL to accommodate shipping quantities that may fluctuate depending on customer demand. This creates a compelling need for a 3PL to operate more efficiently, with imperfect information, to secure profitability, while providing competitive shipping rates for clients and building customer loyalty.

[1]This chapter is based substantially on a published article in the European Journal on Transportation and Logistics. Alnaggar, A., Gzara, F., and Bookbinder, J.H., 2020. "Distribution planning with random demand and recourse in a transshipment network." EURO Journal on Transportation and Logistics 9, no. 1 (2020): 100007.

Many variations of freight distribution coordination with intermediate facilities have been investigated by researchers. However, very limited work addresses such problems with stochastic customer demand. In their literature surveys, both SteadieSeifi et al. (2014) and Guastaroba et al. (2016) acknowledge the need for more research that considers stochasticity in freight transportation planning. In addition, from an industrially-practical point of view, when customer demand arrives in real-time, accounting for demand variation at the distribution planning phase will enable the creation of efficient distribution plans that more accurately anticipate actual distribution costs.

We study the problem of a 3PL that is coordinating transportation needs between suppliers and customers when customer demand is stochastic. That coordination considers the release time of shipments from suppliers, the delivery due dates of customers, the different transportation options that could be used, as well as the holding cost at the consolidation center. In our problem setting, the 3PL does not operate its own fleet, but rather chooses the best available multi-modal transportation services for its clients. The 3PL determines a suitable shipping schedule, arranging for the pickup at suppliers when shipments are ready, i.e., after their release time.

For a given supplier, orders of multiple customers are consolidated in fewer high-volume loads and sent to the consolidation center, operated by the 3PL, through one or more transportation options. A transportation option between a supplier and the consolidation center is referred to as an *inbound transportation option*. We define an inbound transportation option as a combination of a transportation mode (or multiple modes), a capacity, an arrival time at the consolidation center, and a cost associated with the service. At the consolidation center, the 3PL combines orders from multiple suppliers to the same customer and delivers them through one or more transportation options, such that customer delivery deadlines are satisfied. A transportation option from the consolidation center to a customer is referred to as an *outbound transportation option*, and is defined as a combination of transportation mode, capacity, dispatch time from the consolidation center, and cost.

Most transportation service prices are not simply based on the weight and volume of the shipment. Prices also depend on when the service is taking place (e.g., peak seasons, holidays), the mode of transportation used, as well as other factors such as the particular route taken. Thus, inbound and outbound transportation cost may not be monotonically increasing or decreasing with lead time. We adopt this general definition, where inbound and outbound transportation options have nonlinear discrete cost functions.

The distribution service provided by the 3PL is for a predefined number of periods, rather than a one-time service. However, the choice of transportation options, for inbound and outbound shipments, is contractual, and is kept for the full length of the planning horizon. Once customer demand is known, if the actual demand cannot be fulfilled with the particular choice of inbound and outbound options made at the beginning of the planning horizon, a spot market carrier may be used to ship the additional demand at a higher cost. The goal of the 3PL is to select transportation options that minimize the expected transportation cost of the network plus the expected holding cost at the consolidation center, while ensuring that customer demand is fulfilled by the due date.

One motivating example of the problem comes from a 3PL that manages the distribution planning of an e-retailer. The latter operates multiple distribution centers and orders its products from a number of global suppliers. Each supplier provides different types of products that the e-retailer sells. To manage their inventory, each distribution center periodically places a replenishment order, which varies depending on end-customer's demand. In fulfilling those orders, the 3PL uses a consolidation center to save on transportation cost between suppliers and the e-retailer's distribution centers. The 3PL needs to choose a minimal-cost transportation plan with specific transportation modes, capacity and arrival/dispatch times at the consolidation center, for inbound and outbound shipments, respectively, to carry on the regular transportation needs between suppliers and distribution centers. For simplicity and to make our problem applicable to other application areas, we will refer to the third-tier of the supply chain (which are the distribution centers in this example) simply as customers.

The main contributions of this research are threefold. Firstly, we address the need for considering randomness in freight distribution planning with intermediate facilities by proposing a two-stage stochastic programming model that accounts for stochastic customer demand at the planning phase. Our proposed model addresses tactical decisions, i.e., the choice of transportation options, and minimizes the sum of transportation-choice costs plus expected operational costs. Secondly, modeling this problem from the perspective of a third party logistics provider, even without demand uncertainty, has received very little attention. This work aims to fill that gap. Thirdly, we conduct a thorough analysis on the benefits and limitations of our proposed model and present managerial insights on the conditions under which our model achieves significant distribution cost savings.

The rest of this chapter is arranged as follows. In Section 2.2 we provide a review of relevant literature. In Section 2.3, we detail the problem setting and assumptions, and formulate the proposed stochastic model. We discuss the solution methodology used in solving the problem in Section 2.4. We then discuss our numerical testing and analysis, and compare the performance of our stochastic model to its deterministic counterpart in Section 2.5. Finally, we outline some concluding remarks and future directions in Section 2.6.

2.2 Literature Review

References on freight consolidation have considered distinct goals and the viewpoints of different decision makers. Relevant literature is in three main categories: freight/shipment consolidation, freight transportation with intermediate facilities, and freight forwarder/3PL operations. We also discuss important publications that explicitly incorporate stochasticity.

In the past three decades, considerable research has been done on shipment consolidation (SCL). This classical problem mainly aims to find the optimal dispatch policy, from the perspective of a shipper, that determines for how long to consolidate shipments, and when to dispatch the aggregate load. Early research laid the foundation of this topic (Masters, 1980). Later, Higginson and Bookbinder (1994), Çetinkaya and Bookbinder (2003), Mutlu et al. (2010), and Bookbinder et al. (2011) used simulation and stochastic modeling to compare different dispatch policies and determine optimal ones under various settings and considering additional costs, such as inventory cost.

The preceding references explicitly analyze SCL policies, but other researchers have integrated those decisions within wider-scope supply chain network decisions. Freight transportation problems with intermediate facilities were reviewed by Guastaroba et al. (2016). The authors suggested three classes of such problems, the second of which: intermediate facilities in transshipment problems, is the closest to our problem setting, since the consolidation center acts as a transshipment node. Our problem extends the cited references in Guastaroba et al. (2016) by considering a stochastic model rather than a deterministic one. Another article by SteadieSeifi et al. (2014) surveys the literature on multi-modal freight transportation planning. Our proposed model fits under their category of tactical planning, i.e., choice of transport services, associated modes and capacities, and allocating customer orders to the services selected.

Croxton et al. (2003), Berman and Wang (2006) and Song et al. (2008) each studied distribution coordination with consolidation center(s) or merge-in-transit centers. Each paper developed different models to determine the best distribution plan that minimizes transportation plus inventory costs. Croxton et al. (2003) assume that suppliers provide components, which are shipped to a merge-in-transit center, assembled, and dispatched to the customer as a finished product. Song et al. (2008) suppose that suppliers also furnish components, but the customer assembles the product after receiving all parts as one consolidated load. Berman and Wang (2006) assume that each supplier provides a number of products, which are sent to customers via a cross-dock.

Both Croxton et al. (2003) and Berman and Wang (2006) assume that freight is moved via a pre-determined transportation arrangement, so the choice of carriers is not studied.

Song et al. (2008), however, assume that the decision maker (a 3PL) selects from a large number of possible carriers, each with a given dispatch time and cost. We adopt this latter assumption: a typical 3PL chooses the modes and capacities from a number of potential transportation service providers.

The three aforementioned papers had nonlinear cost functions. Transportation costs follow a nonlinear discrete cost function in Song et al. (2008). Similar to those authors, we adopt a general cost function that can capture the various factors affecting transportation cost.

In the context of freight forwarding, most publications assume the relevant company operates its own fleet, proposing different models that extend the classical vehicle routing problem, or the pickup and delivery problem with time windows (Krajewska and Kopfer, 2009, Wang et al., 2014, Bock, 2010). Models that study 3PL coordination issues are closely related to freight forwarders problems; transportation in supply chains is typically outsourced to both 3PLs and freight forwarders. However, 3PLs may coordinate additional distribution activities, like warehousing and managing inventory. Song et al. (2008) study the scheduling problem faced by a 3PL who is arranging shipments between suppliers and customers in an international distribution network through the use of a consolidation center. Cai et al. (2013) analyze the outsourcing of fresh products to a 3PL, where the products could deteriorate during the transportation process, and derive the optimal decisions for supply chain members. Qin et al. (2014) consider the freight consolidation and containerization problem from the perspective of a 3PL that wants to determine the optimal allocation of shipments to international shipping containers and the routing of those containers.

Of some relevance to our work is the extensive family of problems on service network design (SND), as surveyed by Crainic (2000) and Wieberneit (2008). SND decisions relate to the network structure, i.e., selection of routes where service is conducted, and also the movement of freight on the network. Our problem, however, assumes an already-established network, where only the modal choice and scheduling of the freight movements, on predefined routes, is of interest. Furthermore, SND problems often take the carrier's perspective, whereas our view is that of a 3PL that also manages a consolidation center, hence inventory holding cost need be included. Guastaroba et al. (2016) argue that most papers on SND with intermediate facilities concern applications at a national or regional level with a single transport mode. Contrarily, our problem is applicable to global distribution networks, with multiple transportation modes.

All previously reviewed papers assume deterministic customer demand. Limited work addresses similar stochastic demand problems. Guastaroba et al. (2016) recognize that intermediate facilities in stochastic transshipment problems have received no attention, and highlight this for future research. To the best of our knowledge, the only related papers that consider randomness, but without transshipment, are Kılıç and Tuzkaya (2015) and some stochastic service network design papers, surveyed below.

Kılıç and Tuzkaya (2015) investigate a two-echelon distribution network design problem between distribution centers and wholesalers when demand is uncertain. The authors use two-stage stochastic mixed integer programming, where the first stage selects location of distribution centers; the second stage addresses transportation and inventory decisions, as well as unmet demand. In contrast to that article, our work addresses transportation needs in an already-established network.

Several papers have examined the benefit of considering demand randomness in designing service networks. Lium et al. (2009) study demand stochasticity in SND by formulating a two-stage stochastic programming model that chooses the routes and frequency of service in the first stage, and decides on the allocation of commodities to established routes or outsourcing a portion of demand in the second stage. Bai et al. (2014) later extend this model to allow possible rerouting of vehicles, to reduce the amount of outsourcing needed when demand is high. Both our research and Bai et al. (2014) consider outsourcing demand when it exceeds available first-stage capacity. However, since the 3PL in our case does not operate its own vehicle fleet, rerouting is not an option. Moreover, we examine the trade-off between choice of first-stage transportation options and inventory holding cost, a dimension not studied in stochastic network design problems.

Other publications (Hoff et al., 2010 and Crainic et al., 2014) focused on creating efficient solution methodologies for solving realistic instances of stochastic SND problems. Furthermore, more recent work by Wang et al. (2016) examined the value of deterministic solutions, in terms of their quality and upgradeability, in a stochastic environment. Another publication by Wang and Wallace (2016) studied the effect of considering spot markets at the design stage of creating a transportation plan under uncertain demand. The article showed that in most situations, accounting for spot markets when designing a service network reduces total cost.

In the following section, we describe our problem setting, assumptions and formulation.

2.3 Problem Description and Formulation

We propose a two-stage stochastic programming model with recourse, to formulate the Stochastic Distribution Planning with Consolidation (SDPC) problem faced by a 3PL that is coordinating shipments between suppliers, $i \in I$, and customers, $j \in J$, whose demands, D_{ij}, are uncertain. Given customers' demand distributions, delivery due dates and supplier release times, the 3PL needs to select the transportation options for shipments inbound to and outbound from the consolidation center, at the beginning of the planning horizon. Similar to Song et al. (2008), we adopt general, possibly nonlinear, cost functions for inbound and outbound transportation options, $f(x_{iq})$ and $g(y_{jl})$, respectively. Note that these cost functions may differ for distinct inbound and outbound transportation options, $q \in Q_i$ and $l \in L_j$, respectively. Exploiting a general cost function enables consideration of different transportation modes or multi-modal transportation options with varying capacity levels, with those differences reflected in the cost structure.

In our problem setting, the chosen transportation options and their associated capacities are fixed for the whole planning horizon. Once demand is realized, if total demand from a supplier (to a customer) exceeds the capacity of inbound (outbound) transportation option(s) reserved for that supplier (customer), a spot market carrier is used. A spot market may also be used if inventory cost savings outweigh the increased spot market cost. We note that the preceding additional cost of utilizing a spot market carrier is incurred by the 3PL. This cost is composed of two parts, (a) the estimated spot market per unit price premium between the origin and destination, and (b) a disutility factor that represents the 3PL's disutility to transport shipments through a spot market carrier. Such a disutility factor can be set to zero, if shipping price is the only consideration for the 3PL to make shipments through a spot market carrier. However, the 3PL may not favor shipping through a spot market carrier, e.g., due to unpredictable price fluctuations in that market, or limited availability of spot market carriers in peak seasons. Thus, the disutility factor is meant to adjust the level of favorability in using a spot market carrier by the specific 3PL.

The first stage of the two-stage stochastic program tackles the selection of inbound and outbound transportation options to be reserved for the duration of the planning period. The second stage allocates orders to chosen first-stage options, or to spot market carriers. The two stages are optimized simultaneously so as to minimize the sum of transportation cost, expected inventory holding cost and expected spot market carrier shipping cost.

Let x_{iq} be a binary variable indicating whether inbound transportation option $q \in Q_i$ is reserved for supplier $i \in I$. Similarly, the binary variable y_{jl} shows whether outbound transportation option $l \in L_j$ is reserved for customer $j \in J$. Let S be the set of possible scenarios or demand realizations. u^s_{ijq} and w^s_{ijl} are binary variables that express whether shipment (i, j) is shipped through reserved inbound and outbound transportation options q and l, respectively, in scenario $s \in S$. Similarly, μ^s_{ij} and λ^s_{ij} are binary variables that indicate whether shipment (i, j) is moved via an inbound and outbound spot market carrier, respectively, in scenario $s \in S$. Table 2.1 outlines the complete list of notation. We formulate the problem as shown below and refer to the model as the *stochastic distribution planning with consolidation - flow based formulation* (SDPC-FF).

Note that we assume that if shipment (i, j) is transported by a spot market carrier, no holding cost is incurred at the consolidation center, since the shipping time will be chosen such that the interval the load is held at the consolidation center is negligible. Therefore, the holding cost is incurred only if a shipment (i, j) is shipped through reserved inbound and outbound transportation options, i.e., for a specific shipment (i, j), if both variables u^s_{ijq} and w^s_{ijl} have a value of 1. This requirement results in the nonlinearity of the holding cost component of objective function (2.4).

$$[\textbf{SDPC-FF}] \ \min \ \sum_{i \in I} \sum_{q \in Q} f(x_{iq}) + \sum_{j \in J} \sum_{l \in L} g(y_{jl}) + \zeta(\boldsymbol{x_{iq}, y_{jl}}) \tag{2.1}$$

$$\text{subject to} \ \ x_{iq}, y_{jl} \in \{0,1\}, \quad i \in I, q \in Q, j \in J, l \in L \tag{2.2}$$

where

$$\zeta(\boldsymbol{x_{iq}, y_{jl}}) = \mathbb{E}_{\xi} \zeta_s(\boldsymbol{x_{iq}, y_{jl}, \xi}) \tag{2.3}$$

$$\begin{aligned}\zeta_s(\boldsymbol{x_{iq}, y_{jl}, d^s}) = \min \Bigg\{ & \sum_{i \in I} \sum_{j \in J(i)} d^s_{ij} h_i (\sum_{l \in L} \tau_{jl} w^s_{ijl} (1 - \mu^s_{ij}) \\ & - \sum_{q \in Q} \tau_{iq} u^s_{ijq} (1 - \lambda^s_{ij})) + \sum_{i \in I} \sum_{j \in J(i)} d^s_{ij} (\pi_i \mu^s_{ij} + \pi_j \lambda^s_{ij}) \Bigg\}\end{aligned} \tag{2.4}$$

subject to

$$\sum_{q \in Q} u^s_{ijq} + \mu^s_{ij} = 1, \qquad i \in I, j \in J(i) \tag{2.5}$$

$$\sum_{l \in L} w^s_{ijl} + \lambda^s_{ij} = 1, \qquad j \in J, i \in I(j) \tag{2.6}$$

$$\sum_{q \in Q} \tau_{iq} u^s_{ijq} \leq \sum_{l \in L} \tau_{jl} w^s_{ijl} + \bar{\tau}_{iq} \lambda^s_{ij}, \qquad j \in J, i \in I(j) \tag{2.7}$$

$$\sum_{j \in J(i)} d^s_{ij} u^s_{ijq} \leq C_{iq} x_{iq}, \qquad i \in I, q \in Q \tag{2.8}$$

$$\sum_{i \in I(j)} d^s_{ij} w^s_{ijl} \leq C_{jl} y_{jl}, \qquad j \in J, l \in L \tag{2.9}$$

$$\begin{aligned} & u^s_{ijq}, \mu^s_{ij} \in \{0,1\}, && i \in I, q \in Q, j \in J(i) \\ & w^s_{ijl}, \lambda^s_{ij} \in \{0,1\}, && j \in J, i \in I(j), l \in L \end{aligned} \tag{2.10}$$

The objective function (2.1) minimizes the total inbound and outbound transportation cost, $f(x_{iq})$ and $g(y_{jl})$, respectively, plus the recourse function: the expected value of the second-stage problem. For a particular demand realization $\boldsymbol{d^s}$ of the random vector $\boldsymbol{\xi}$, objective (2.4) minimizes the sum of the holding cost and the cost of shipping through a spot market carrier. Constraints (2.5) and (2.6) guarantee that the model allocates each order to exactly one inbound shipment, and exactly one outbound shipment, respectively, whether the shipment is through a reserved first-stage transportation option or a spot market carrier. Constraints (2.7) make sure that the outbound dispatch time of an order is greater than its inbound arrival time. Constraints (2.8) and (2.9) ensure that the total demand allocated to a transportation option, for a given supplier and customer, respectively, does not exceed the capacity of that option. Finally, Constraints (2.10) impose the binary requirement on all variables.

In order to avoid the nonlinearity in objective function (2.4), we propose an equivalent linear path-based formulation that replaces the flow variables with path variables. We refer to this formulation as the *stochastic distribution planning with consolidation - path based formulation* (SDPC - PF), and we detail it next.

Linear path formulation

We define a set of feasible paths for shipment (i, j) from supplier i to customer j through the consolidation center as P_{ij}, where a feasible path $p_{ijql} \in P_{ij}$ represents a pair of inbound and outbound transportation options (q, l) that is feasible with regard to arrival/dispatch times for shipment (i, j). In other words, shipment (i, j) has a feasible path p_{ijql} if inbound option q arrives at the consolidation center before outbound option l is dispatched. Shipment (i, j) also has feasible paths through each of its inbound options q and through an outbound spot market, and similarly, there are feasible paths along each outbound option l and an inbound spot market. For a shipment (i, j), inbound and outbound shipping via a spot market carrier is also a feasible path. For brevity, we refer to a feasible path as $p \in P_{ij}$. We define the following additional notation. a_{iqp} and b_{jlp} are binary parameters indicating if options q and l are on path $p \in P_{ij}$. c^s_{ijp} is the cost of sending shipment (i, j) on path p in scenario s, where

$$c^s_{ijp} = \begin{cases} d^s_{ij} h_i(\tau_{jl} - \tau_{iq}), & \text{if no spot market carrier is used on path } p \in P_{ij} \\ \pi_i d^s_{ij}, & \text{if a spot market carrier is used only for inbound shipping on path } p \in P_{ij} \\ \pi_j d^s_{ij}, & \text{if a spot market carrier is used only for outbound shipping on path } p \in P_{ij} \\ (\pi_i + \pi_j) d^s_{ij}, & \text{if a spot market carrier is used for both inbound and outbound shipping on} \\ & \text{path } p \in P_{ij} \end{cases}$$

We also define decision variables β^s_{ijp} as binary variables that indicate whether or not shipment (i, j) traverses path $p \in P_{ij}$ in scenario s. Figure 2.1 shows a visual representation of the network with both the flow and path variables. The path based formulation, SDPC-PF, can then be expressed as follows:

$$[\textbf{SDPC-PF}] \ \min \sum_{i\in I}\sum_{q\in Q} f(x_{iq}) + \sum_{j\in J}\sum_{l\in L} g(y_{jl}) + \zeta(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}) \tag{2.11}$$

$$\text{subject to} \quad x_{iq}, y_{jl} \in \{0,1\}, \quad i \in I, q \in Q, j \in J, l \in L \tag{2.12}$$

where

$$\zeta(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}) = \mathbb{E}_\xi \zeta_s(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}, \boldsymbol{\xi}) \tag{2.13}$$

$$\zeta_s(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}, \boldsymbol{d^s}) = \min \sum_{i\in I}\sum_{j\in J(i)}\sum_{p\in P_{ij}} c^s_{ijp}\beta^s_{ijp} \tag{2.14}$$

subject to

$$\sum_{p\in P_{ij}} \beta^s_{ijp} = 1 \qquad \forall i \in I, j \in J(i) \tag{2.15}$$

$$\sum_{p\in P_{ij}} a_{iqp}\beta^s_{ijp} \le x_{iq} \qquad \forall i \in I, j \in J(i), q \in Q \tag{2.16}$$

$$\sum_{p\in P_{ij}} b_{jlp}\beta^s_{ijp} \le y_{jl} \qquad \forall j \in J, i \in I(j), l \in L \tag{2.17}$$

$$\sum_{p\in P_{ij}}\sum_{j\in J(i)} a_{iqp}d^s_{ij}\beta^s_{ijp} \le C_{iq} \qquad \forall i \in I, q \in Q \tag{2.18}$$

$$\sum_{p\in P_{ij}}\sum_{i\in I(j)} b_{jlp}d^s_{ij}\beta^s_{ijp} \le C_{jl} \qquad \forall j \in J, l \in L \tag{2.19}$$

$$\beta^s_{ijp} \in \{0,1\}, \qquad i \in I, j \in J(i), p \in P_{ij} \tag{2.20}$$

The objective function (2.11) minimizes the total transportation cost plus the expected value of the second-stage problem. For a specific realization $\boldsymbol{d^s}$ of the random vector $\boldsymbol{\xi}$, objective (2.14) minimizes the total allocation cost of shipments to feasible paths. Constraints (2.15) ensure that exactly one path is chosen for each shipment (i, j) in the network. Constraints (2.16) and (2.17) guarantee that shipment (i, j) traverses a path only if both the inbound transportation option of supplier i (x_{iq}) and the outbound transportation option of customer j (y_{jl}) have a value of 1. Constraints (2.18) and (2.19) require that the total demand that traverses a given path does not exceed the capacity of the inbound or outbound transportation options of that path. Finally, Constraints (2.20) impose the binary requirement on the variables.

Note that constraints (2.18) and (2.19) may be modified by multiplying their left-hand-

Table 2.1. List of notations used in SDPC-FF.

Notation	Meaning
Parameters	
I	The set of suppliers
J	The set of customers
S	The set of demand realizations or scenarios
$I(j)$	The set of suppliers from which customer j orders
$J(i)$	The set of customers that order from supplier i
Q_i	The set of inbound transportation options for supplier i
L_j	The set of outbound transportation options for customer j
C_{iq}	The capacity of inbound transportation option q of supplier i
C_{jl}	The capacity of outbound transportation option l of customer j
τ_{iq}	The arrival time of inbound transportation option q of supplier i at the CC
X_i	The set of all arrival times of inbound transportation options of supplier $i \in I$
τ_{jl}	The dispatch time of outbound transportation option l of customer j from the CC
Y_j	The set of all dispatch times of outbound transportation options of customer $j \in J$
$\bar{\tau}_{iq}$	The latest arrival time of all inbound transportation options $q \in Q_i$ of a given supplier i, i.e., $\max_{q \in Q_i}\{\tau_{iq}\}$
d^s_{ij}	The demand of customer j from supplier i in scenario s
h_i	The holding cost of supplier i's shipment at the CC
π_i	Inbound spot market carrier cost per unit from supplier i to CC
π_j	Outbound spot market carrier cost per unit from CC to customer j
$f(x_{iq})$	Inbound transportation cost of transportation option q for supplier i
$g(y_{jl})$	Outbound transportation cost of transportation option l for customer j
Decision Variables	
x_{iq}	Binary variable, equals 1 if inbound transportation option q for supplier i is reserved
y_{jl}	Binary variable, equals 1 if outbound transportation option l for customer j is reserved
u^s_{ijq}	Binary variable, equals 1 if shipment (i,j) is transported from supplier i to CC through option q in scenario s
w^s_{ijl}	Binary variable, equals 1 if shipment (i,j) is transported from CC to customer j through option l in scenario s
μ^s_{ij}	Binary variable, equals 1 if shipment (i,j) is transported from supplier i to CC by a spot market carrier, in scenario s
λ^s_{ij}	Binary variable, equals 1 if shipment (i,j) is transported from CC to customer j by a spot market carrier, in scenario s

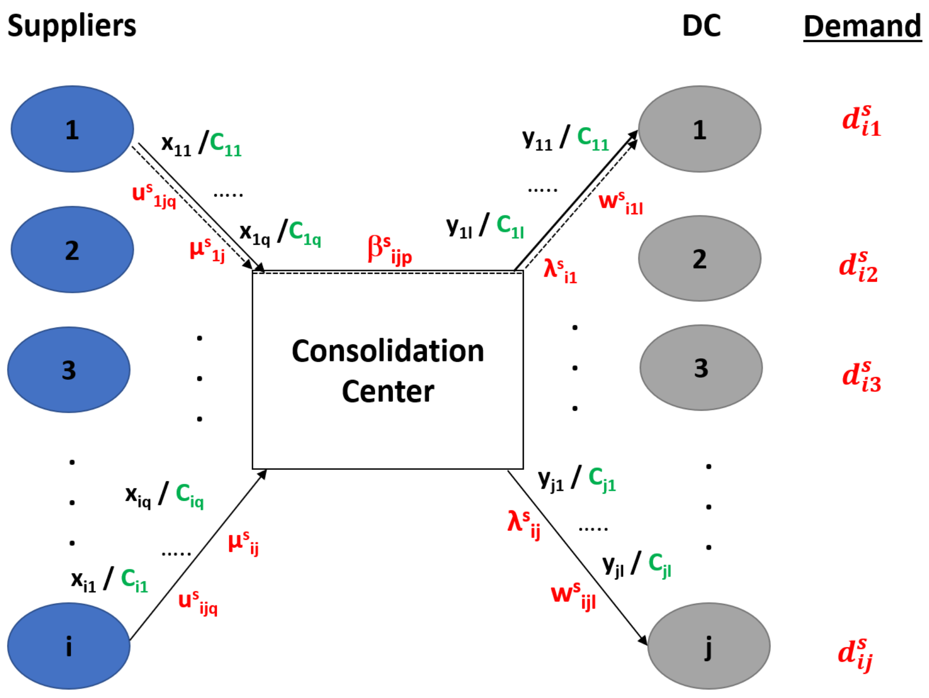


Figure 2.1. Visual representation of the network with the main decision variables and parameters.

side by x_{iq} and y_{jl}, respectively. However, empirical testing showed that such modification caused a slight increase in computational time for some instances. Therefore, we refrain from using this adjustment in our computational testing.

We use Sample Average Approximation (SAA) to solve SDPC-PF. The main advantage of this technique is that it provides a statistical estimate of the optimality gap of the *true* stochastic optimization problem, which is discretized by a very large scenario tree. In contrast, solving the problem directly with a commercial solver with 50 or more scenarios is computationally intensive for reasonable size problems, as it results in a large number of path variables. Solving the problem directly also gives little information on the quality of the solutions obtained, relative to the *true* stochastic problem. We therefore use SAA to measure the quality of the resulting distribution plans by utilizing the optimality gap estimate as a quality metric, and also to keep the problem size manageable and obtain good solutions in a reasonable amount of time.

2.4 Solution Methodology: Sample Average Approximation

Sample Average Approximation is a Monte Carlo simulation-based solution technique for solving two-stage or multi-stage stochastic optimization problems (Mak and Wood, 1999, Kleywegt et al., 2002). In this technique, the objective function of the stochastic model is approximated by a sample average estimate obtained from a random finite set of samples. The problem is then solved, with the approximate objective function and a set of scenarios S^N, as a deterministic optimization problem either directly or using other solution techniques. The process is repeated M times with different samples, and each time results in a candidate solution. To assess the quality of the candidate solutions, statistical estimates of their optimality gaps can be obtained.

SAA solves the *true* problem with a reasonable level of accuracy provided some conditions are met (Kleywegt et al., 2002, Shapiro and Philpott, 2007). Those conditions, and justifications on how SDPC-PF meets them, are as follows:

1. It is possible to generate a sample realization of the random vector ζ. For our proposed problem, this can be done by sampling from each (i, j) demand distribution.
2. The SAA problem can be solved efficiently with a moderate sample size. We will show in Section 2.5 that we can solve SDPC-PF in a reasonable amount of time, for most test instances, with a sample size of $N = 10$.
3. The function $\zeta_s(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}, \boldsymbol{d^s})$ can be easily computed for given $\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}$ and $\boldsymbol{d^s}$. That is, for a given first stage solution and a given realization of demand, the optimal objective function (2.14) can be easily evaluated by solving the model in Equations (2.14) to (2.20).
4. The true problem has relatively complete recourse, i.e., any solution to the first stage problem is feasible to the second stage because it can be *corrected.* In SDPC-PF, this is done through the assumption that a spot market carrier is always available when demand cannot be fulfilled with reserved first stage variables. Thus, any choice of transportation plan would result in a feasible second stage problem, because shipping via the spot market is a feasible path for all shipments (i, j).

We now detail how SAA is used to solve SDPC-PF. Applying SAA, the objective function of the second stage problem of SDPC-PF is approximated as:

$$\zeta(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}) = \frac{1}{N} \sum_{s \in S^N} \sum_{i \in I} \sum_{j \in J(i)} \sum_{p \in P_{ij}} c^s_{ijp} \beta^s_{ijp} \tag{2.21}$$

where S^N is the set of scenarios of size N sampled in a given SAA problem. The full SAA model is expressed as follows:

$$\begin{aligned}
&[\textbf{SAA Model}]\ \min \sum_{i\in I}\sum_{q\in Q} f(x_{iq}) + \sum_{j\in J}\sum_{l\in L} g(y_{jl}) + \zeta(\boldsymbol{x_{iq}}, \boldsymbol{y_{jl}}) \qquad (2.22)\\
&\text{subject to}\\
&\text{constraints } (2.15)-(2.20), (2.21) \qquad \forall s \in S^N
\end{aligned}$$

To assess the quality of the SAA solution, statistical estimates of lower and upper bounds on the objective function value of the original stochastic problem may be obtained, as well as estimates of the variances of these bounds. This is achieved by solving the SAA model M times, where each time a set of independent samples of size N is generated. This results in M candidate solutions, $\boldsymbol{x^1}, \ldots, \boldsymbol{x^M}$, where $\boldsymbol{x^m}$ is the vector notation of the solution of the first stage variables, $\boldsymbol{\bar{x}_{iq}}, \boldsymbol{\bar{y}_{jl}}$ for candidate solution $m \in \{1, \ldots, M\}$, with objective function values $\eta^1, \ldots, \eta^M$.

To estimate the lower bound of the true objective function value, we first compute the mean ($\bar{\eta}$) and the variance ($\hat{\sigma}^2_{N,M}$) of the objective function values $\eta^1, \ldots, \eta^M$ as:

$$\bar{\eta} = \frac{1}{M}\sum_{m=1}^{M} \eta^m \qquad (2.23)$$

$$\hat{\sigma}^2_{N,M} = \frac{1}{M(M-1)}\sum_{m=1}^{M} (\eta^m - \bar{\eta})^2 \qquad (2.24)$$

The lower bound is expressed as:

$$LB = \bar{\eta} - t_{\alpha,v}\hat{\sigma}_{N,M} \qquad (2.25)$$

where $t_{\alpha,v}$ is the α-critical value of the t-distribution with v degrees of freedom, $v = M - 1$.

Kleywegt et al. (2002) note there is a trade-off between SAA solution quality and computational requirements as the size N changes. With a larger N, the objective function value of the SAA problem gets closer to the true objective value, but the computational requirement increases significantly. Similarly, as the number of replications M increases, a better lower bound can be obtained with a smaller standard deviation $\hat{\sigma}^2_{N,M}$. However, the algorithm may become computationally inefficient. The exact values of N and M used in our computational testing are explained in Section 2.5.

The upper bound on the true objective function value of each candidate solution is obtained by evaluating the solution with a very large scenario tree of size N' that is assumed to represent the true distribution of demand. Since each scenario $s \in \{1, \ldots, N'\}$ is an i.i.d. random sample, the problem of evaluating a candidate solution decomposes into N' subproblems. The size of the scenario tree N' is much larger than the size of the scenario tree maintained in each SAA run, N. We denote the objective function value of a given subproblem s as $\phi(\boldsymbol{x^m}, s)$, which is computed as shown in Equation (2.26). Note that because each subproblem is solved separately, N' can be very large without causing a significant computational burden. The estimate of the true objective value of the second stage problem, denoted as $\phi(\boldsymbol{x^m})$, is computed as shown in (2.27).

$$\phi_s(\boldsymbol{x^m}, s) = \sum_{i \in I} \sum_{j \in J(i)} \sum_{p \in P_{ij}} c_{ijp}\beta_{ijp} \qquad \forall s \in \{1, \ldots, N'\} \tag{2.26}$$

$$\phi(\boldsymbol{x^m}) = \frac{1}{N'} \sum_{s=1}^{N'} \phi_s(\boldsymbol{x^m}, s) \tag{2.27}$$

The value of the true objective function, $\bar{\eta}^m$, for candidate solution $\boldsymbol{x^m}$, and its variance, $\sigma^2_{N'}(\boldsymbol{x^m})$, are computed as shown in Equations (2.28) and (2.29), respectively,

$$\bar{\eta}^m = \sum_{i \in I} \sum_{q \in Q} f(\bar{\boldsymbol{x}}_{\boldsymbol{iq}}) + \sum_{j \in J} \sum_{l \in L} g(\bar{\boldsymbol{y}}_{\boldsymbol{jl}}) + \phi(\boldsymbol{x^m}), \tag{2.28}$$

$$\sigma^2_{N'}(\boldsymbol{x^m}) = \frac{1}{N'(N'-1)} \sum_{s=1}^{N'} [\phi_s(\boldsymbol{x^m}, s) - \phi(\boldsymbol{x^m})]^2. \tag{2.29}$$

Finally, the upper bound of a candidate solution, z_U^m is computed as:

$$\eta_U^m = \bar{\eta}^m + z_\alpha \sigma_{N'}(\boldsymbol{x^m}) \tag{2.30}$$

where z_α is the α-critical value of the standard normal distribution. The upper bound of the algorithm is the smallest η_U^m ,$\forall m \in \{1, \ldots, M\}$, as shown in Equation (2.31). The final solution of SAA, $\boldsymbol{x^*}$, is the candidate solution that results in the smallest optimality gap $(\eta_U^m - \eta_L)$ for all candidate solutions $m \in \{1, \ldots, M\}$, which corresponds to the solution with the smallest upper bound η_U^m, as shown in (2.32).

$$UB = \min_{m \in \{1, \ldots, M\}} \eta_U^m \tag{2.31}$$

$$\boldsymbol{x^*} = \underset{m \in \{1, \ldots, M\}}{\text{argmin}} \, (\eta_U^m) \tag{2.32}$$

2.5 Computational Experiments and Analysis

In this section, we conduct extensive computational testing to assess the effectiveness of SAA in solving the SDPC-PF and to evaluate the benefit of accounting for uncertainty in modeling SDPC. We solve problem instances of various sizes and different experimental settings using the SAA algorithm. We then compare the solution of SDPC to its deterministic counterpart with average demand values. We refer to the deterministic problem as the *deterministic distribution planning with consolidation* (DDPC) and we describe its formulation in Appendix A.2. We compare the stochastic and deterministic solutions and objective values to evaluate the benefit of accounting for uncertainty, through computing the *value of stochastic solution*.

We briefly describe the data generation method used and discuss the different data sets used in testing and analysis, and how they compare and contrast, in Section 2.5.1. Detailed explanation of data generation is provided in Appendix A.1. We further elaborate on our computational testing by detailing the setting used for the SAA algorithm as well as some key performance measures in Section 2.5.2. Finally, we report and discuss the results of our computational testing in Section 2.5.3. The SAA algorithm was implemented in Python 2.7 on an Intel(R) Core(TM) i7 CPU, 2.90 GHz, 16.00 GB of RAM. The optimization problems were solved by CPLEX 12.8.

2.5.1 Data Generation and Data Sets

We generate the parameters of the test instances partly following the method outlined by Song et al. (2008), since their proposed model also studies a distribution planning problem from the perspective of a 3PL. We randomly generate the additional parameters used in our SDPC-PF. More particularly, we use Song et al.'s method to generate the network of suppliers and customers and the sets of arrival times and dispatch times of inbound and outbound options, X_i, Y_j, for suppliers and customers, respectively. We modify their proposed cost function of inbound and outbound options $f(x_{iq}), g(y_{jl})$, by incorporating capacity. We also scale down their holding cost h_i to make it a cost per unit, rather than per shipment. We then generate the following additional parameters: capacities C_{iq} and C_{jl} for inbound and outbound options, demand distributions for each (i, j) shipment, and spot market inbound and outbound transportation cost, π_i and π_j, respectively. The detailed data generation method is outlined in Appendix A.1.

Based on this method, we generate 10 data sets of different sizes. Each set is composed of 5 instances that share the same network data but differ in the demand distributions and the transportation options available for suppliers and customers. Table 2.2 outlines the problem size of each data set in terms of the numbers of suppliers and customers, and number of (i, j) shipments. The disutility factor of the spot market carrier is set to r = 4. Recall that transportation cost through a spot market carrier is a variable per unit cost. The 3PL thus pays for exactly the shipping amount needed and has more flexibility in shipping time, as opposed to the reserved transportation options, which justifies the cost difference. The effect of the disutility factor on the expected outsourcing amount and the benefit of using SDPC-PF are analyzed in Section 2.5.2

Table 2.2. Sizes of the data sets used in the computational experiments.

Data Set No.	No. of Suppliers	No. of Customers	No. of Shipments
Set 1	5	5	20
Set 2	5	10	20
Set 3	5	10	40
Set 4	10	10	50
Set 5	10	20	50
Set 6	10	20	100
Set 7	10	30	100
Set 8	10	30	200
Set 9	20	20	100
Set 10	20	20	200

We now better examine how inventory holding times at the consolidation center and the number of inbound and outbound transportation options may influence the benefit of using the SDPC-PF. For each set outlined above, we develop four experimental settings. Each setting considers three arrival/dispatch times for each supplier i and customer j. We assume that each supplier and customer have a slow option, an average-speed option and a fast option. The arrival times $\tau_{iq} \in X_i$ of an inbound fast option, average-speed option, and slow option for a given supplier i are generated uniformly in the ranges U[100,235], U[235,370], and U[370,500], respectively. Similarly, the dispatch times $\tau_{jl} \in Y_j$ of outbound options that are fast, of average speed option, and slow for a given customer j are generated uniformly in the respective ranges U[370,500], U[235,370], and U[100,235].

The four experimental settings differ in the following way:

- **(A)** For each of the three arrival and dispatch times of inbound and outbound options, two levels of capacity are considered, creating a total of six transportation options per supplier and customer. The two capacity levels are $\gamma = 1.00$ and $\gamma = 1.15$. In this experimental setting, arrival/dispatch times of inbound and outbound options are generated independently. For example, an average-speed option of supplier i does not necessarily arrive before the dispatch time of the average-speed option of customer j, given that $i \in I(j)$. This results in higher average wait times at the consolidation center, and therefore greater inventory cost.

- **(B)** For each of the three arrival and dispatch times of inbound and outbound options, the same two levels of capacity are considered $\gamma = 1.00$ and $\gamma = 1.15$, creating six transportation options per supplier and customer. However, under this setting, arrival times $\tau_{iq} \in X_i$ of suppliers $i \in I$ are synchronized with the dispatch times $\tau_{jl} \in Y_j$ of customers $j \in J(i)$ for specific speed levels. That is, for a given supplier i and customer $j \in J(i)$, supplier i's fast transportation option is guaranteed to arrive before customer j's slow option is dispatched. The same synchronization is done for different speed levels, such that average-speed and slow supplier options arrive before the dispatch time of average-speed and fast customer options, respectively. This creates instances of lower average holding times at the consolidation center.

- **(C)** Arrival and dispatch times are generated independently, similar to **(A)**, but three capacity levels ($\gamma = 1.00, \gamma = 1.15$, and $\gamma = 1.3$) are considered for each time, thus creating a total of nine options per supplier and customer. We are interested to know how having an additional capacity level may change the solution, and also how increasing the number of transportation options may affect the efficiency of the SAA algorithm, when average holding times at the consolidation center are high.

- **(D)** Arrival and dispatch times are synchronized, similar to **(B)**, and three capacity levels are considered for each time, $\gamma = 1.00, \gamma = 1.15$, and $\gamma = 1.3$, creating a total of nine options per supplier and customer. Similar to **(C)**, we wish to understand the impact of an additional capacity level on the solution of SDPC, and the efficiency of SAA with the increased number of transportation options, when average holding times are low.

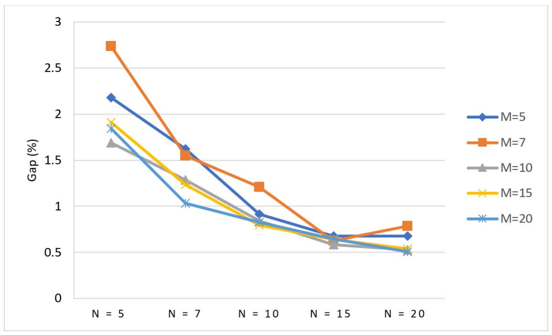
2.5.2 Experiments

SAA settings

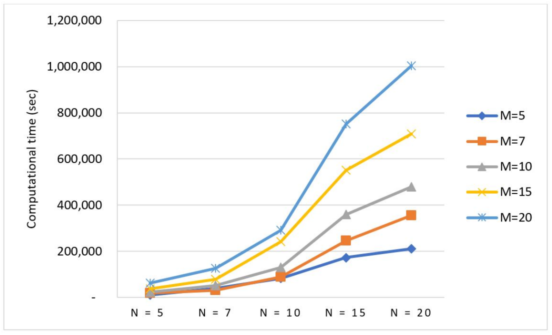
We use SAA to solve the SDPC-PF, as outlined in Section 2.4. We define $N = 10$ scenarios to estimate the expected second stage cost. This is then repeated for $M = 10$ SAA problems so as to estimate a lower bound on the *true* expected cost.

Each scenario includes a realization of demand for each shipment (i, j). We sample these realizations from the demand distribution data (Section A.1.2). Each SAA problem is solved using CPLEX 12.8, with a maximum time limit of 1200 seconds (20 minutes). The lower bound is computed as in Equation (2.25), with $t_{\alpha=5,v=9} = 1.833$, for 95% confidence interval and 9 (N-1) degrees of freedom.

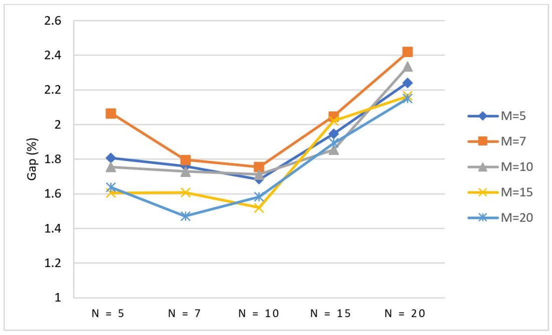
Our choice of N and M was based on empirical results so as to achieve a reasonable trade-off between gap and computational time. Figure 2.2 shows the gap and the computational time of two instance sets, 6A and 8A. Though the computational time when $N = 10$ and $M = 10$ is higher than when both or either N and M take lower values, the benefit is apparent in the reduced estimated optimality gap. Set 8A is more computationally demanding; notice how increasing the value of N and M actually causes an increase in the estimated gap since the optimality of each SAA run is not achieved in the imposed time limit.



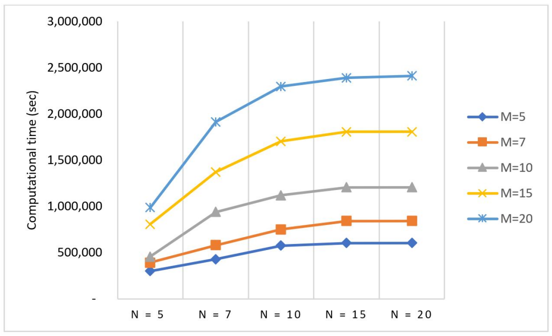
(a) Gap estimate for different values of N and M - Set 6A.



(b) Total computational time of the M SAA runs, each with N scenarios - Set 6A.



(c) Gap estimate for different values of N and M - Set 6A.



(d) Total computational time of the M SAA runs, each with N scenarios - Set 8A.

Figure 2.2. Trade-off between gap and computational time for different values of N and M.

To obtain the upper bound on expected cost of the *true* problem, we consider all individual solutions, $\mathbf{x^m}$, of the M runs, and evaluate them using a scenario tree of $N' = 1000$ scenarios. For each of the $\mathbf{x^m}$ solutions, we calculate the expected second stage cost of the solution and compute $\bar{\eta}^m$ as shown in Equation (2.28). We then compute the upper bound η_U^m as in Equation (2.30), with $z_{\alpha=5} = 1.64$, for a 95% confidence interval. The estimated upper bound of the algorithm is $\min_{m=\{1,\ldots,M\}} \eta_U^m$, as shown in Equation (2.31).

Performance measures

To assess the advantage of taking demand uncertainty into account at the modeling phase, we compare the SAA solution for each problem instance to the solution of its deterministic counterpart DDPC-PF, shown in Section A.2. Particularly, we solve the deterministic problem to obtain the distribution plan when the mean demand is used for each shipment (i, j). We then evaluate the deterministic distribution plan using the same 1000 scenario tree used to obtain the upper bounds of the SAA solutions. This shows the expected cost savings achieved when distribution plans are constructed using SDPC as opposed to DDPC. This value is referred to in the literature as *the value of stochastic solution* (Birge and Louveaux, 2011). We report this value as $\frac{(\bar{\eta}_{det} - \bar{\eta}_{stoch})}{\bar{\eta}_{stoch}}$, where $\bar{\eta}_{det}$ and $\bar{\eta}_{stoch}$ are the objective values of the deterministic and the stochastic solutions, respectively, when the second stage problem is assessed on the full $N' = 1000$ scenario tree, computed as shown in Equation (2.28). Note that $\bar{\eta}_{stoch}$ is the objective value of the best SAA run, with the lowest optimality gap.

To further highlight the potential benefits of our model, we report the *expected outsourcing* that each distribution plan requires. That is, the expected shipment amount, as a percentage of total expected demand, that travels via spot market carriers rather than a reserved first stage transportation option in the 1000-scenario tree demand distribution. This provides a measure of risk associated with the transportation plan of a given solution by emphasizing the extent to which first stage reserved transportation options $\bar{\boldsymbol{x}}_{\boldsymbol{iq}}, \bar{\boldsymbol{y}}_{\boldsymbol{jl}}$ are capable of satisfying demand. We also report the *expected utilization* of reserved transportation options for each instance, seeking a possible relationship between expected utilization of options and expected outsourcing.

2.5.3 Computational Results

SAA results

The SAA algorithm was applied to the different data sets of Section 2.5.1 for experimental settings A, B, C and D. Results are shown in Tables 2.3 and 2.4. Note that each row reports the average over 5 instances of the specified size and setting. For example, row 1A in Table 2.3 shows the average results of 5 instances of Set 1 (5 suppliers, 5 customers, 20 shipments), experimental setting A. For additional clarity, we also provide detailed results for one problem instance in Appendix A.3.

The first column of Tables 2.3 and 2.4 specifies the data set number. The relative gap of the algorithm is reported in the second column and the number of feasible paths used to form the model is shown in the third, to demonstrate the problem size. Column 4 exhibits the *value of stochastic solution*, a measure of the benefit of using our proposed stochastic model over its deterministic counterpart. Columns 5 and 6 report the *expected outsourcing* for solutions of the stochastic and the deterministic models, respectively. Columns 7 and 8 show the *expected utilization* of reserved inbound and outbound transportation options for the stochastic and deterministic models. For insight on the fluctuation of cost over different scenarios, we report the relative standard deviation of the lower and upper bounds, as a percentage of their respective means, in columns 9 and 10. Finally, columns 11 to 14 respectively report the computational time (in seconds) of the full algorithm, the time to solve the 10 SAA runs, the time to compute upper bounds, and the time to solve the deterministic counterpart (DDPC).

SAA performance

The results clearly demonstrate that the optimal distribution plans of the SDPC are more cost efficient than the plans of DDPC, once the actual demand is realized. Instances across the different data sets and experimental settings show that the SDPC yields significant expected cost savings, as outlined by the *value of stochastic solution*. Observe, however, that the advantage of the SDPC, compared to DDPC with nominal values, is less notable for denser problem instances, when a greater number of customer orders are consolidated in a single load. This can be observed in the average *value of stochastic solution* of Sets 2 and 3, 5 and 6, 7 and 8, 9 and 10 across all experimental settings. For each of these pairs of sets, the network size is the same, i.e., the same number of suppliers and customers, but the number of shipments doubles. For example, the average value of stochastic solution drops from 39.11% in Set 2A to 11.79% in Set 3A. The same trend can be observed when comparing Sets 9A and 10A; the value of stochastic solution drops from 16.57% to 3.00%. This implies that the SDPC problem is more beneficial in sparse networks as opposed to denser ones. This observation can be explained by the fact that as the number of combined (i, j) shipments in an inbound or outbound transportation option increases, the mean of the consolidated shipment approaches the *true* mean, and therefore the deterministic solution, with mean demand, becomes comparable to the stochastic one.

To show the relationship between the value of stochastic solution and the number of shipments in an instance, we calculate the average ratio of number of shipments per supplier and per customer as $(\frac{no.shipments}{|I|} + \frac{no.shipments}{|J|})/2$. Then, for all instances shown in Tables 2.3 and 2.4, we graph the ratio of the average number of shipments per supplier and customer versus the value of stochastic solution in Figure 2.3. The x-axis shows the ratio in an ascending order and its corresponding set number. We see in the figure that as the number of shipments per supplier and customer increases, the value of stochastic solution decreases. We also observe that the different experimental settings show very similar trends, with a slightly higher value of stochastic solution for settings B and D, with the lower average holding time.

Note from the tabular results that denser problem instances have much higher computational burden than sparser ones. This is seen in the SAA time reported in column 12 of Tables 2.3 and 2.4. Sets 7 and 8, for example, both have 10 suppliers and 30 customers. However, Set 8 has double the number of shipments of Set 7, i.e., 200 and 100 shipments, respectively. The average computational time of Set 8B for solving the 10 SAA runs is 12022 seconds, while that of Set 7B is only 524 seconds. Nonetheless, the average computational time of the upper bound calculation is slightly higher for Set 8B, but somewhat comparable, respectively 340 and 259 seconds for 8B and 7B.

Tables 2.3 and 2.4 also suggest that the difference in *expected outsourcing* percentage between solutions of SDPC and DDPC is more significant for sparser problems. This difference decreases for denser problem instances. In other words, for denser instances, the expected outsourcing percentage of DDPC is low (compared to sparser instances) and is closer in value to that of SDPC. Figure 2.4 plots the expected outsourcing for solutions of both SDPC and DDPC versus the ratio of number of shipments to suppliers and customers and highlights such an observation; the difference between the expected outsourcing percentage of SDPC and DDPC decreases for denser problem instances. For the expected utilization of reserved options, we observe that denser problem instances have slightly higher expected utilization than sparser instances in the solutions of both SDPC and DDPC.

The average optimality gap is at most 1.28% for instances of all data sets except Sets 8 and 10, with 200 shipments. For those two sets, the average optimality gap is at most 3.74%. We note, however, that in those sets, the maximum time limit of each SAA run (1200 seconds) is reached, which implies that some or all of the SAA runs may have not been solved to optimality, negatively affecting the quality of the candidate solutions and, in turn, the optimality gap estimate. Because of the low optimality gap across different instances under the current SAA setting outlined in Section 2.5.2, there is no motive to increase the number of scenarios N maintained in the SAA problem.

We notice that the results of Set B, with the lower holding time at the consolidation center, are comparable to those of Set A, implying that holding time does not have a major impact on the benefit of SDPC. Nonetheless, the value of stochastic solution is slightly higher for Set B compared to Set A, and the expected outsourcing of Set B is a bit lower than that of Set A. This indicates that the reduced wait time in Set B marginally

Table 2.3. Results of computational experiments Sets A and B with $r = 4$.

Set No.	Gap (%)	No. of Paths	Value of Stochastic Solution (%)	Expected Out-sourcing SDPC (%)	Expected Out-sourcing DDPC (%)	Expected Utiliza-tion SDPC (%)	Expected Utiliza-tion DDPC (%)	LB std. dev. (%)	UB std. dev. (%)	Total Time (sec)	SAA Time (sec)	UB Time (sec)	Det. Time (sec)
1A	0.30	606	23.58	0.59	8.85	80.03	82.78	0.09	0.00	37	7	29	0.31
2A	0.25	622	39.11	0.78	13.46	69.47	65.90	0.08	0.01	46	7	38	0.36
3A	0.96	1227	11.79	0.44	7.31	87.38	91.25	0.18	0.04	220	90	125	0.58
4A	0.76	1512	13.38	1.02	7.30	82.57	87.11	0.18	0.03	323	138	177	0.73
5A	0.39	1502	27.54	0.15	11.85	81.89	78.14	0.13	0.03	191	44	141	0.82
6A	1.10	3009	9.25	1.19	5.82	85.45	89.59	0.19	0.05	1558	1312	231	1.38
7A	0.94	2984	18.42	1.79	7.94	79.67	80.68	0.20	0.05	987	710	261	1.52
8A	1.71	6077	4.45	1.63	3.75	87.65	93.55	0.08	0.05	11572	11198	344	2.96
9A	0.64	2995	16.57	0.99	7.24	78.14	83.88	0.11	0.02	998	574	408	1.47
10A	2.66	6003	3.00	2.46	3.74	87.63	90.83	0.10	0.07	12502	12019	453	2.71
Avg	0.97	2654	16.71	1.10	7.73	81.99	84.37	0.13	0.04	2843	2610	221	1.28
1B	0.46	740	24.41	0.00	9.27	86.39	88.68	0.13	0.00	56	8	46	0.33
2B	0.31	740	42.79	0.00	13.46	76.01	72.55	0.09	0.00	40	8	31	0.37
3B	1.28	1480	13.27	1.01	7.00	88.50	91.43	0.29	0.10	217	90	121	0.58
4B	0.54	1850	16.50	0.00	7.80	85.78	90.68	0.12	0.00	245	80	158	0.74
5B	0.88	1850	29.49	0.23	12.07	86.46	83.23	0.20	0.04	250	76	167	0.83
6B	1.06	3700	10.91	0.61	5.74	87.92	92.22	0.19	0.06	2691	2439	237	1.40
7B	0.69	3700	20.22	0.26	7.41	83.49	87.35	0.13	0.04	799	524	259	1.51
8B	2.38	7400	5.78	0.97	3.72	91.56	95.72	0.09	0.05	12392	12022	340	2.92
9B	0.68	3700	17.93	0.12	7.36	86.56	90.34	0.11	0.02	983	582	385	1.48
10B	3.12	7400	3.95	1.29	3.46	89.34	94.88	0.06	0.07	12508	12022	456	2.97
Avg	1.14	3256	18.52	0.45	7.73	86.20	88.71	0.14	0.04	3018	2785	220	1.31

Table 2.4. Results of computational experiments Sets C and D with $r = 4$.

Set No.	Gap (%)	No. of Paths	Value of Stochastic Solution (%)	Expected Out-sourcing SDPC (%)	Expected Out-sourcing DDPC (%)	Expected Utiliza-tion SDPC (%)	Expected Utiliza-tion DDPC (%)	LB std. dev. (%)	UB std. dev. (%)	Total Time (sec)	SAA Time (sec)	UB Time (sec)	Det. Time (sec)
1C	0.41	1147	22.75	1.55	10.00	80.57	83.63	0.12	0.01	59	13	45	0.35
2C	0.15	1182	40.67	1.50	13.97	66.16	62.47	0.05	0.00	47	10	35	0.44
3C	1.06	2313	11.92	1.02	7.41	85.48	91.21	0.22	0.06	266	125	134	0.66
4C	0.94	2874	13.73	0.48	7.51	84.09	88.27	0.20	0.02	306	129	170	0.83
5C	1.01	2889	26.11	0.90	11.90	82.38	79.22	0.17	0.04	291	94	188	0.99
6C	0.94	5781	9.69	1.17	5.76	86.76	91.39	0.22	0.05	4428	4164	247	1.67
7C	0.65	5754	17.20	1.63	7.47	78.87	80.84	0.14	0.06	2647	2349	280	1.78
8C	2.29	11494	3.85	1.30	3.63	88.49	93.51	0.10	0.05	12441	12028	378	3.28
9C	0.81	5754	16.32	0.81	7.28	78.67	83.83	0.15	0.03	1779	1342	419	1.74
10C	3.52	11531	2.75	2.72	3.65	86.98	91.82	0.06	0.07	12559	12029	495	3.29
Avg	1.18	5072	16.50	1.31	7.86	81.85	84.62	0.14	0.04	3482	3228	239	1.50
1D	0.34	1320	27.19	0.00	9.54	85.24	88.14	0.11	0.00	49	13	34	0.36
2D	0.87	1350	40.59	0.00	12.91	77.34	72.77	0.26	0.00	58	12	44	0.42
3D	1.10	2771	12.39	0.64	6.76	87.20	91.30	0.26	0.07	301	149	145	0.68
4D	1.07	3394	14.64	0.07	7.79	85.70	90.28	0.21	0.01	376	169	198	0.87
5D	0.77	3447	28.57	0.22	11.79	85.55	83.14	0.17	0.03	241	63	170	1.00
6D	1.04	6812	10.27	0.88	5.19	87.91	92.00	0.18	0.07	3824	3565	241	1.68
7D	0.78	6706	19.20	1.04	6.89	84.89	86.55	0.19	0.07	1418	1139	262	1.90
8D	2.12	13498	5.89	1.16	3.50	90.78	95.18	0.07	0.06	12392	11974	382	3.50
9D	0.60	6854	18.30	0.15	7.38	85.81	89.98	0.09	0.02	786	377	391	1.77
10D	3.74	13590	3.07	1.87	3.50	89.17	94.73	0.08	0.07	12571	12031	504	3.41
Avg	1.24	5974	18.01	0.60	7.53	85.96	88.41	0.16	0.04	3202	2949	237	1.56

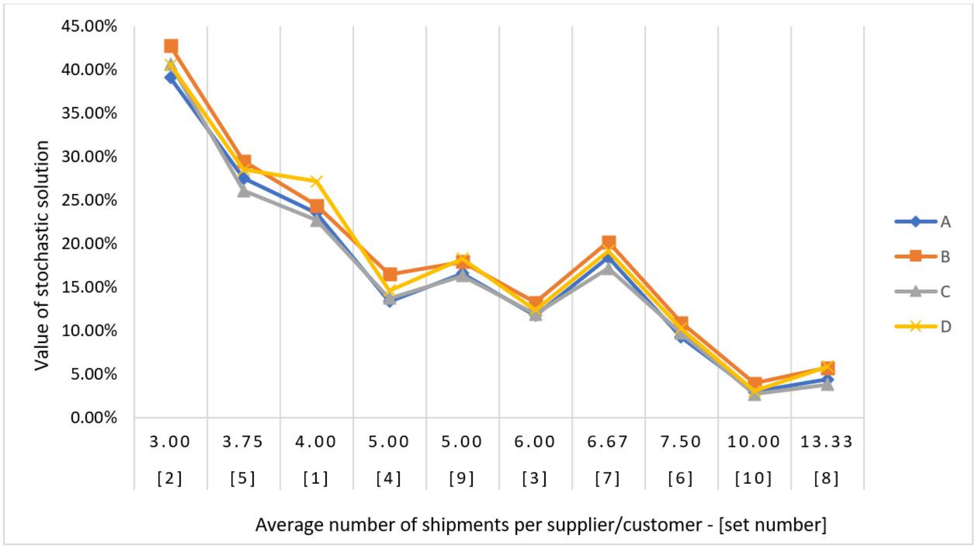


Figure 2.3. Relationship between value of stochastic solution and number of shipments per supplier and customer.

decreases the need for outsourcing done to reduce holding cost, thus improving the value of stochastic solution.

Sets C and D, with the greater number of options, show similar trends to Sets A and B, in terms of the value of stochastic solution, the percentage of outsourcing and utilization for both SDPC and DDPC. However, Sets C and D have higher average computational time of SAA runs, for most instances, compared to Sets A and B. This increase in computational time is at most double for most instances, with a few exceptions, e.g., the SAA time of 6C is about 3 times that of 6A.

Effect of the spot market disutility factor and the holding cost on the benefit of SDPC

We conduct analysis on experimental settings A and B, for all datasets, when the disutility factor changes from $r = 4$ to $r = 2$ and the holding cost decreases by 50%. Since experimental settings C and D show similar trends to A and B in the computational results in Tables 2.3 and 2.4 but have higher computational time, we focus on settings A and B only. We compare the optimality gap estimate, the value of stochastic solution, the expected outsourcing and expected utilization for different combinations of r and h. Results are shown in Table 2.5, where each row displays the average of 5 instances of the specified size and setting.

We observe that the change in optimality gap as the disutility factor and holding cost decrease is very slight for both experimental settings. We also note that for a given

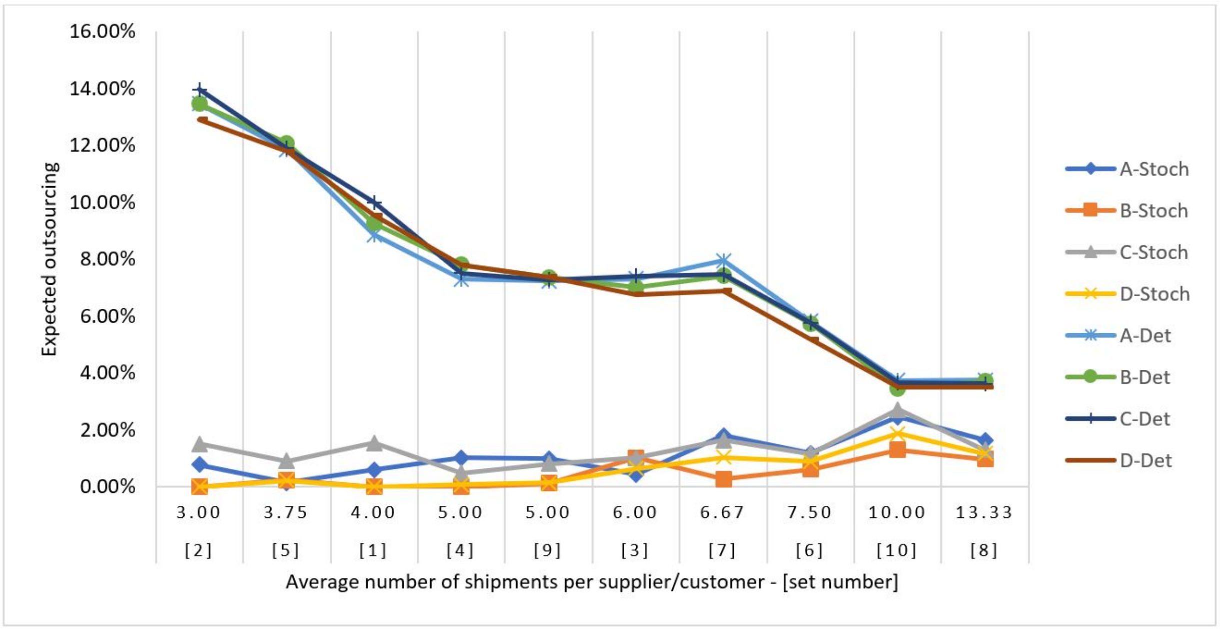


Figure 2.4. Expected outsourcing vs. number of shipments per supplier and customer for SPDC and DDPC.

disutility level, reducing the holding cost provides minor improvements in the value of stochastic solution. In other words, reduction in holding cost only provides a very small amount of additional cost savings for the solutions of SDPC compared to DDPC, even for instances with higher average holding time. The value of stochastic solutions improves an average of 0.67% and 0.29% for settings A and B, respectively, with the reduction in holding cost. This result is explained by the fact that the lower holding cost is reflected in both SDPC and DDPC, and the cost savings between the two problems are mainly achieved through reserving higher transportation option capacity to reduce the need to outsource to the spot market when demand is high. Nevertheless, we note that the reduction in holding cost does slightly reduce the amount of expected outsourcing, and this decrease is more notable for setting A instances as compared to B. We also observe that the expected utilization is only marginally affected by the changes in holding cost. Settings A and B have an average increase of utilization of 1.83% and 0.30% as holding cost decreases.

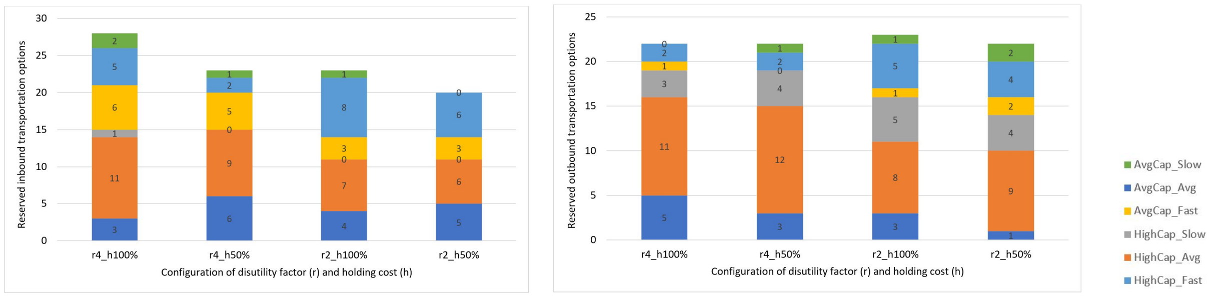
Results also suggest that the disutility factor has the most impact on both the value of stochastic solution and expected outsourcing percentage. This is anticipated, since the disutility cost is a variable per unit cost and the model assumes that shipping through a spot market carrier results in no holding cost. Lowering the disutility factor to $r = 2$ therefore reduces the cost difference between first stage options and spot market. Particularly, we notice that the benefit of incorporating randomness in the model is positively correlated to the value of the disutility factor. That is, as the spot market shipping gets closer to that of reserving a transportation option ahead of time, and the 3PL is indifferent to spot market shipping, considering customer demand stochasticity in the planning phase does not result in remarkable cost savings. Thus, any chosen first-stage transportation plan can easily be adjusted when actual demand is realized, at only a small cost.

Structural differences between solutions of the different configurations of disutility factor and holding cost

The previous section examines the impact of changes in the disutility factor and holding cost on the benefit of SDPC. Here we analyze how the structure of the distribution plans obtained from solution of SDPC differs as the configuration of disutility factor and holding cost changes. To do so, we focus on instance 9A3, discussed in Appendix A.3, and see how the transportation plan changes for distinct capacity and speed levels. Figure 2.5 shows a breakdown of inbound and outbound transportation options for each of the four combinations of disutility and holding cost considered in Section 2.5.3. The x-axis refers to the instance name by the specific values of r and h in the instance. For example, r4_h50% shows the results of the instance 9A3 when we solve it with $r = 4$ and 50% of the holding cost. Each column in Figures 2.5a and 2.5b shows the breakdown of all reserved inbound and outbound transportation options, respectively, based on their capacity and speed levels, under each parameter setting. The different parts of a given column show the number of options with a given capacity and speed level, where HighCap and AvgCap refer to options with $\gamma = 1.15$ and $\gamma = 1.00$, respectively, and Fast, Avg, Slow, refer to the three speed levels considered in the instance.

Figure 2.5 suggests that both the holding cost and the disutility factor impact the actual distribution plan of SDPC. A decrease of 50% in holding cost, when $r = 4$, reduces the number of inbound by 5, but keeps the number of outbound options unchanged. This implies that with a high holding cost, and when average holding time is high, reserving additional capacity may diminish the total network cost by cutting down on holding cost. We note, however, that the reduction in holding cost has a greater impact on the choice of speed levels of reserved options more than their capacity levels. For example, for a given value of r, when h decreases by 50%, approximately the same number of high capacity options is reserved as when h is kept at 100%. However, the change in the breakdown of the reserved options based on speed is more apparent. This is intuitive since as holding cost decreases, the trade-off between transportation and holding cost becomes less important. Therefore, the solution may choose plans that result in higher wait times in an effort to reduce transportation cost and in turn, minimize total cost.

As opposed to reduction in holding cost, we note from Figure 2.5 that the reduction in the disutility factor affects both the choice of the reserved transportation options' capacity and speed levels. For example, when h is at 100%, the number of high capacity inbound options decreases from 15 to 11, and the number of high capacity outbound options decreases from 19 to 16, when r changes from 4 to 2. We also notice an increase in the number of average-capacity average-speed options, both inbound and outbound. This reinforces the results of the analysis in Section 2.5.3; as the value of the disutility factor decreases and the spot market cost decreases, there is less need to develop robust distribution plans, since adjusting plans after demand is realized is not costly.



(a) Inbound reserved options

(b) Outbound reserved options

Figure 2.5. Breakdown of the transportation plan of the best SAA run of different configurations of r and h.

2.6 Conclusion

In this chapter, we studied the stochastic distribution planning with consolidation problem from the viewpoint of a 3PL managing a three-echelon supply chain network. We proposed a two-stage stochastic program with recourse to model the problem of selecting inbound and outbound transportation options for 3PL distribution planning, subject to stochastic customer demand. To date, the literature on distribution planning in transshipment networks does not consider uncertainty faced in practical applications. This study offers an extension of previous work by considering probabilistic demand in tactical decisions faced by a 3PL that is handling the distribution needs of its clients.

Because of the nonlinearity in the objective function of our proposed *stochastic distribution planning with consolidation - flow based formulation* (SDPC-FF) model, we suggested an alternative linear formulation, the *stochastic distribution planning with consolidation - path based formulation* (SDPC-PF). The latter generates all feasible paths for shipments in the network, and decides on the transportation options to reserve and the allocation of shipments to paths. We applied Sample Average Approximation (SAA) to solve the

Table 2.5. Analysis on the effect of changing r and h for experimental settings A and B.

Set No.	Gap				Value of Stochastic Solution				Expected Outsourcing (%)				Expected Utilization (%)			
	$r=4$		$r=2$		$r=4$		$r=2$		$r=4$		$r=2$		$r=4$		$r=2$	
	h=100%	50%	h=100%	50%	h=100%	50%	h=100%	50%	h=100%	50%	h=100%	50%	h=100%	50%	h=100%	50%
1A	0.30	0.58	1.43	1.31	23.58	24.38	9.29	11.18	0.59	0.00	9.15	7.36	80.03	82.51	82.71	83.33
2A	0.25	0.29	0.62	0.88	39.11	41.18	21.42	22.30	0.78	0.00	10.77	5.13	69.47	73.25	66.58	70.36
3A	0.96	0.75	0.96	0.85	11.79	12.28	3.39	3.40	0.44	0.36	6.23	5.92	87.38	87.23	88.51	88.66
4A	0.76	0.85	1.01	1.16	13.38	13.80	2.88	3.44	1.02	0.77	5.70	5.95	82.57	84.56	84.32	85.92
5A	0.39	0.51	0.80	0.89	27.54	28.61	12.84	13.68	0.15	0.15	5.25	4.47	81.89	83.25	83.14	84.10
6A	1.10	1.23	0.75	0.85	9.25	9.72	2.79	2.81	1.19	1.06	5.95	4.92	85.45	86.96	86.93	89.24
7A	0.94	0.93	0.59	0.60	18.42	19.24	7.19	8.13	1.79	1.35	8.02	6.41	79.67	80.74	78.42	79.98
8A	1.71	1.71	0.54	0.47	4.45	4.64	1.06	1.08	1.63	1.45	5.31	4.68	87.65	89.52	90.17	92.04
9A	0.64	0.79	0.84	0.88	16.57	17.45	4.79	6.11	0.99	0.97	10.24	8.98	78.14	80.41	76.84	81.09
10A	2.66	2.99	0.38	0.47	3.00	3.02	1.04	0.84	2.46	2.52	6.11	5.97	87.63	90.08	87.43	88.37
Avg	0.97	1.06	0.79	0.84	16.71	17.43	6.67	7.30	1.10	0.86	7.27	5.98	81.99	83.85	82.51	84.31
1B	0.46	0.76	1.40	1.78	24.41	25.00	8.93	9.32	0.00	0.00	3.53	1.76	86.39	86.39	86.92	86.95
2B	0.31	0.14	0.96	0.96	42.79	43.50	23.48	24.23	0.00	0.00	6.75	3.30	76.01	77.44	73.78	76.63
3B	1.28	1.10	1.10	1.05	13.27	13.48	4.11	4.20	1.01	0.83	2.35	2.26	88.50	88.15	90.00	89.87
4B	0.54	0.93	1.04	1.28	16.50	16.83	5.00	5.05	0.00	0.00	4.55	3.57	85.78	85.78	85.54	86.37
5B	0.88	0.65	0.86	0.87	29.49	30.05	14.02	14.42	0.23	0.23	1.92	1.76	86.46	86.46	86.96	87.55
6B	1.06	1.15	1.16	1.26	10.91	11.13	2.60	2.58	0.61	0.53	3.89	3.53	87.92	87.79	90.05	89.44
7B	0.69	0.82	0.72	0.77	20.22	20.62	7.96	8.26	0.26	0.38	2.65	2.45	83.49	83.77	85.12	84.93
8B	2.38	2.63	0.80	0.57	5.78	5.75	0.83	0.91	0.97	0.97	2.41	2.34	91.56	91.19	93.91	93.70
9B	0.68	0.81	1.28	1.27	17.93	18.37	5.65	5.84	0.12	0.38	5.29	3.99	86.56	86.83	85.38	86.60
10B	3.12	3.33	0.61	0.73	3.95	3.97	0.77	0.80	1.29	1.38	2.33	2.25	89.34	89.95	92.44	92.36
Avg	1.14	1.23	0.99	1.05	18.53	18.87	7.33	7.56	0.45	0.47	3.57	2.72	86.20	86.37	87.01	87.44

SDPC-PF, and tested it extensively to evaluate its benefits and limitations. We also compared the solutions obtained by the SDPC-PF to a deterministic heuristic for the stochastic problem, the *deterministic distribution planning with consolidation* (DDPC), with mean demand values, to assess the advantage of incorporating stochasticity in the modeling phase.

Our computational testing suggests that significant cost savings can be achieved when generating distribution plans using the SDPC rather than DDPC. The results also demonstrate that the stochastic model greatly reduces the amount of outsourcing needed in the second stage problem, compared to the deterministic case. We notice, however, that the benefit of SDPC is less notable for denser problem instances, where large numbers of shipments are consolidated in a single load. We also observe that changes in second-stage cost may affect the benefit of SDPC or the structure of the choice of first-stage transportation options, or both. For instance, reduction in holding cost does not affect the benefit of SDPC, but changes the choice of transportation options. On the other hand, the spot market cost plays a major role in how beneficial the SDPC problem is, and in the choice of transportation options. This finding suggests that if the cost of shipping through a spot market carrier is not much greater than the cost of reserving transportation options, and the 3PL's disutility of shipping through the spot market is low, there is less need to establish a robust distribution plan ahead of time. That follows since correcting the initial plan, once actual demand is realized, would not result in remarkable additional costs.

Future research could extend SDPC to also incorporate stochasticity in the arrival/dispatch times of transportation options and study how the solution would change compared to the current model as well as the deterministic case. Another possible extension is to consider the decision variables of transportation options as integer rather than binary, with the 3PL having to choose how many vehicles, of a particular level of capacity and a certain arrival/dispatch time, to reserve for inbound and outbound shipping for the duration of the planning horizon.

Studying multiple consolidation centers, but choosing to send each shipment through exactly one, is another interesting direction. A related suggestion is a model with two consolidation centers, one closer to suppliers and the other closer to customers. This is a more representative model of global distribution planning; what is the benefit of accounting for uncertainty under that setting? Other directions include considering different cost functions for the spot market, and the possibility of consolidating inbound and outbound spot market shipments, to achieve economies of scale.

Chapter 3

Crowdsourced Delivery: A Review of Platforms and Academic Literature[1]

3.1 Introduction

E-retailing and same-day delivery continue to experience remarkable growth year after year. The total revenue of online sales of physical goods in the US totaled up to \$504.6 billion US dollars, in the year 2018, and is expected to rise to \$735 billion US dollars in 2023 (Statista, 2019). With this growth comes increased competition, especially in providing shorter delivery time windows. In an effort to improve their logistics competitiveness, well-established e-retailers as well as local small to medium sized businesses have experimented with new innovative delivery systems to provide faster deliveries in cost-effective manners.

One such systems is "crowdsourced delivery", where ordinary people carry out last-mile deliveries with their own vehicles, from stores or warehouses to customers' destinations. This system falls under the broader emergent concept of the "sharing economy", which has created highly successful business models, such as *Uber* and *Airbnb*, in the last decade. The main distinguishing factor in those business models is that they rely on individuals *sharing* their under-utilized property for the mutual benefit of deriving value for themselves and for the business. Though the concept of the sharing economy is fairly old, also known as *collaborative consumption*, advances in mobile communication technologies and global positioning systems (GPS) enabled its widespread emergence in recent years. Habibi et al. (2017) argue that the boom of the sharing economy followed the financial collapse of 2008,

[1]This chapter is based substantially on a published article in Omega: The International Journal of Management Science. Alnaggar, A., Gzara, F., and Bookbinder, J.H., 2021. "Crowdsourced Delivery: A Review of Platforms and Academic Literature." Omega, 98, 102139.

which has created a greater need to reduce customer costs. According to PricewaterhouseCoopers (2015), the global revenue of the sharing economy was worth $15 billion in 2015, and is estimated to grow to $335 billion by 2025.

In 2015, Amazon implemented crowdsourced last-mile delivery by introducing *Amazon Flex*; an on-demand package delivery service that hires independent freelance drivers to transport same-day delivery packages to Amazon customers (Halzack, 2015). Some of the items delivered are ordered under *Prime Now*, a delivery scheme offered by Amazon with a guaranteed 2-hour delivery time window. Other items are ordered under *Amazon Fresh*, a company division that handles grocery orders (Amazon.com, 2019). In 2018, Walmart began piloting crowdsourced delivery in two US cities, under the name *Spark Delivery* (Bose, 2018). In addition to implementations of this system by large e-retailers, multiple start-ups have been launched in the last decade that offer last-mile crowdsourced deliveries (e.g. Deliv, DoorDash, Hitch, Postmates). Those companies aim to provide more affordable shipping services than old-school shipping methods through postal services or overpriced couriers. They do so by connecting shippers to a network of people who are willing to provide shared-mobility services, as a side activity that generates additional income. Those services may be used for personal use or by businesses that offer fast, same-day shipping.

The goal of this survey is to describe and analyze the state of the art of current crowdsourced delivery systems in the industry, and review the OR literature addressing this emerging topic. In doing so, we also identify and review related subproblems in this system that span other classical transportation problems heavily studied in the literature. We outline the key differences of crowdsourced delivery systems and the new challenges they bring about. To the best of our knowledge, there is no work in the literature that surveys and compares existing crowdsourced delivery systems, provides a classification or taxonomy of existing systems in practice, and proposes a typology of decisions within crowdsourced delivery systems

The rest of this chapter is arranged as follows. In Section 3.2 we describe the current crowdsourced delivery systems in the industry and compare their matching and scheduling mechanisms as well as their compensation schemes. Section 3.3 reviews the literature on crowdsourced delivery and compares the literature's features and assumptions to the real industry crowdsourced delivery systems. In Section 3.4, we provide a disaggregation of the managerial decisions within a crowdsourced delivery system, compare those subproblems to classical problems in the literature and identify new challenges that this emergent system creates. Section 3.5 is dedicated to discussing future research opportunities. Finally, we make some concluding remarks in Section 3.6.

3.2 Crowdsourced Delivery Platforms

3.2.1 Overview of Available Crowdsourced Delivery Platforms

In this section, we review the different crowdsourced delivery platforms, mostly available in North America. We compare and contrast how the different platforms work, what distinguishes them from their competitors, and what are the main items they deliver. We also observe how drivers are compensated, and the type of information collected from drivers. Note that we do not review companies that provide crowdsourced delivery in other regions, such as Europe and China, if their official websites are not in English. From our research, we found that the main companies in other countries that do not provide official documentation in English are Trunkrs and PickThisUp in the Netherlands, and Renren Kuaidi in China. Since these are only three companies, we believe that this exclusion does not significantly affect our analysis.

We group the delivery platforms into two main categories, e-retailers and couriers. For e-retailers, Amazon Flex (Amazon.com, 2019) and Walmart Spark Delivery (Walmart, 2019) are currently the only available crowdsourced delivery platforms in North America. In contrast, courier crowdsourced delivery companies are quickly growing in number. Those include companies like Postmates (2019), Deliv (2019), DoorDash (2019), Kanga (2019), UberEats (2019), Hitch (2019), Roadie (2019), PiggyBee (2019), Nimber (2019), DHL MyWays (DHL, 2013), UberFreight (2019), Truxx (2019), and BuddyTruk (2019). With the exception of Amazon Flex, which pays a driver a per-hour rate from the time he/she checks in at the distribution center, all the other delivery platforms pay drivers a compensation per completed delivery. According to the driver information on the official websites of those platforms, the value of such compensation is calculated by a formula, which may be city-specific, which considers factors like mileage, wait time, size of package and other factors. Drivers are notified of the earning amount before they accept a delivery task. We note that not all those companies are equally as established. Some of them are operating in multiple cities in the US and Canada, while others are just in a single city.

Companies like Postmates (2019), Deliv (2019), Roadie (2019) and Kanga (2019) allow for the delivery of all items, with the exception of a shortlist of prohibited items. Other companies like DoorDash (2019) and UberEats (2019) focus mainly on food delivery. UberEats (2019) partners with local restaurants to provide delivery for their meals. DoorDash (2019) provides both delivery from restaurants as well as grocery delivery. Instacart (2019) and Shipt (2019) provide grocery delivery, and require its independent contractors to not only deliver groceries to customers, but also to do the shopping themselves from a nearby grocery store. It's worth noting that aside from Walmart's piloted service, Spark Delivery, Walmart is currently also using Doordash and Postmates in fulfilling customers' online grocery orders (Bose, 2018).

All platforms, with the exception of Hitch (2019) and Roadie (2019), assume that drivers are making dedicated trips just for the sake of fulfilling the delivery request, and are not necessarily heading in that direction. Contrarily, Hitch (2019) and Roadie (2019) match delivery requests with a traveler's already pre-planned trip. In other words, they collect information on when a driver is planning to go somewhere (the time and destination), and offer to drivers delivery requests that are already on their way. While Hitch (2019) mainly supports local deliveries, Roadie (2019) allows for local and long-haul deliveries. All other platforms, with the exception of truck crowdsourced services, provide only local deliveries.

UberFreight (2019), Truxx (2019) and BuddyTruk (2019) are the main platforms that match bulky delivery requests to crowdsourced drivers with a truck. UberFreight (2019) is mainly for long-haul full-truckload requests by businesses or enterprises that can be fulfilled by owner-operators of trucks. Truxx (2019) and BuddyTruk (2019), on the other hand, aim to match personal delivery requests of bulky items (e.g. furniture) with drivers that own smaller-size trucks. Typically, requested trips for those platforms are less than an hour long.

Kanga (2019) and Hitch (2019)'s target market is individuals rather than enterprises, i.e., connecting people who want to send or receive a package or item with a driver willing to make the delivery. Kanga (2019) gives shippers and drivers even the ability to agree on a delivery price, while charging a service fee for connecting them.

Table 3.1 compares the different features of the various crowdsourced delivery platforms. We note that Amazon Flex is composed of four divisions: *Amazon Logistics, Amazon Prime Now, Amazon Fresh,* and *Amazon Restaurants*. The difference between the four divisions is indicated in Table 3.1, the primary difference being in the delivery time window and type of items delivered. We also observe that all platforms have a system for rating drivers, and incentives based on good performance. Drivers may earn higher compensation during busy seasons and periods of high demand.

3.2.2 Crowdsourced Delivery Scheduling and Matching Mechanisms

Available crowdsourced delivery platforms follow distinct approaches in scheduling crowdsourced drivers and matching them with delivery requests. We classify the scheduling and matching mechanisms into four main schemes.

1. **Pure self-scheduling.** All crowdsourced delivery services use flexibility of working hours as a major selling point that attracts crowdsourced delivery drivers. Yet different crowdsourced delivery systems use varying levels of such flexibility. Systems that follow pure self-scheduling refer to platforms that do not require drivers to indicate their hours of availability beforehand. Simply, when a driver is available to be matched with a delivery request, he/she logs into the mobile app and keeps the app running in the background. Once an order arrives, whose pickup is within a specified radius of where the driver is, the driver is notified through the app of the request, and may choose to accept or reject the offer. Those platforms are very similar to how ride-hailing services such as Uber and Lyft operate. The following crowdsourced delivery platforms follow this scheduling and matching mechanism: Postmates (2019), DoorDash (2019), UberEats (2019), as well as Truxx (2019) and BuddyTruk (2019) for truck delivery.

2. **Hybrid and centralized scheduling.** Some crowdsourced delivery systems use a more centralized scheduling approach to better balance supply and demand at various times of the day. Those systems either require drivers to indicate their availability on the mobile app, then receive delivery offers when they become available, or pick shifts that work for their schedule on a first-come first-serve basis. Shifts are usually posted well in advance, up to a week ahead, and additional on-demand shifts may be posted on the app throughout the day. Amazon Flex, Deliv (2019), Instacart (2019) and Shipt (2019) follow this type of scheduling and matching approach. Such scheduling and matching is closest to traditional delivery services with a company's own fleet, since supply and capacity are more predictable. Some systems, such as Amazon Flex and Deliv (2019), provide minimum pay guarantees for drivers, which means that drivers are promised to be paid a given minimum amount, even if they are not matched. Such programs further reduce uncertainty in supply and make the system closer to classical scheduling, matching and routing problems.

   As per the driver information on their official websites, Amazon Flex guarantees drivers a pay of $18 per hour, regardless of the number of delivery requests. Deliv, on the other hand, guarantees half the amount of the "Time on Task" rate, which is one factor in computing driver pay, in addition to a per mile rate. The time on task for Deliv is between $13-$18, depending on the city.

3. **En-route matching.** For this type of matching, drivers are matched with delivery requests that are on their way of a pre-planned trip. A traveler or commuter indicates on the system's mobile app the date, time, origin, and destination of an upcoming trip. Then the app matches the traveler with delivery requests on his/her way, such

Table 3.1. Comparison of features of crowdsourced delivery platforms.

Company	Delivery Time Window	Payment Scheme	Min pay guarantee	Min no. of hours/shift	Long haul	Delivery items	Pre-planned trip	Who determines shipping rate	Target Market
E-retailer									
Amazon Logistics	1-2 days	per hour	Yes	3-4 hours	No	Any	No	company	Businesses
Amazon Prime Now	1-2 hours	per hour	Yes	3-4 hours	No	Any	No	company	Businesses
Amazon Fresh	2-3 hours	per hour	Yes	3-4 hours	No	Any	No	company	Businesses
Amazon Restaurants	1 hour	per hour	Yes	3-4 hours	No	Any	No	company	Businesses
Walmart	0.5-4 hours	per delivery	No	No	No	Grocery	No	company	Businesses
Couriers									
Deliv	same day	per delivery	Yes	3-4 hour	No	Any	No	company	Businesses & Individuals
DHL MyWays	any	per delivery	No	No	No	Any	Maybe	company	Individuals
DoorDash	1 hour	per delivery	No	No	No	Restaurant & grocery	No	company	Businesses
Hitch	same-day	per delivery	No	No	No	Any	Yes	company	Individuals
Instacart	same-day	per order	No	2 hours	No	Grocery	No	company	Individuals
Kanga	any/same-day	per delivery	No	No	No	Any	No	shipper & driver	Individuals
Nimber	any	per delivery	No	No	Yes	Any	Yes	company	Businesses & Individuals
PiggyBee	any	per delivery	No	No	Yes	Any	Yes	shipper & traveler	Individuals
Postmates	1 hour	per delivery	No	No	No	Any	No	company	Businesses & Individuals
Roadie	any/same-day	per delivery	No	No	Yes	Any	Yes	company	Businesses & Individuals
Shipt	min 1 hour	per order	No	1 hour	No	Grocery	No	company	Individuals
UberEATS	1 hour	per delivery	No	No	No	Restaurant	No	company	Businesses
Couriers (bulky items)									
BuddyTruk	any	per delivery	No	No	Yes	Bulky	No	company	Individuals
Truxx	any	per delivery	No	No	Yes	Bulky	No	company	Individuals
Uber Freight	any	per delivery	No	No	Yes	Any	No	company	Businesses

that a maximum distance or time deviation of the original travel route is observed. Such matching closely resembles ride-sharing problems, which aim to match drivers with riders on their way, with a small possible deviation. We review the literature of ride-sharing in Section 3.3.3.

4. **Bulletin-board type matching.** This refers to systems that simply post delivery requests and a driver picks requests that match his/her schedule and preference. In such systems, no algorithm is used to automatically match drivers to delivery requests, and the matching is done mainly through the sharing of information. Walmart Spark Delivery follows such a system, where orders, with their associated destination and delivery time window, are posted on the app, and drivers pick orders that they can fulfill. Kanga (2019) and DHL MyWays employ that sort of approach, where requests are posted and drivers match themselves. Nimber (2019) and PiggyBee (2019) follow a similar system, but matching is also based on a driver's pre-planned trip. In PiggyBee, a traveler/commuter posts his/her travel plans and a customer who wants to send or receive something along that route contacts the traveler with information about the request, and they agree on a price. Nimber (2019), on the other hand, works the opposite way. A list of requests is maintained and a driver may look up a request that matches his/her travel plans. If none is available, a driver may input his/her travel plans and get notified once delivery requests become available along his/her route. UberFreight (2019) also uses this type of matching, where delivery requests are posted on the app, and a truck owner-operator chooses the loads that work for him/her.

Figure 3.1 provides an overview of the first three scheduling and matching schemes, and a sample of their resulting routes. Note that we exclude the fourth pattern as the matching is manually done, entirely through the sharing of information. The actual trip under pattern 4 may resemble either pattern 1 or 3. In other words, a driver who manually matches him/herself to a request may make a dedicated trip similar to pattern 1, though not necessarily in a short time window. He/she may also choose to match him/herself with an en-route order, similar to pattern 3. Note that the term *additional functions*, in the second pattern of Figure 3.1, refers to the possibility of a crowdsourced driver having to conduct other tasks, such as shopping for customers.

In Table 3.2, we analyze the target markets of different platforms categorized by the matching mechanisms described above. Note that we differentiate between platforms designed for the targeted use by individual consumers ordering from local stores/restaurants, and those designed as an alternate transportation solution for businesses. In the former, the crowdsourced delivery platform partners with local stores and restaurants that do not

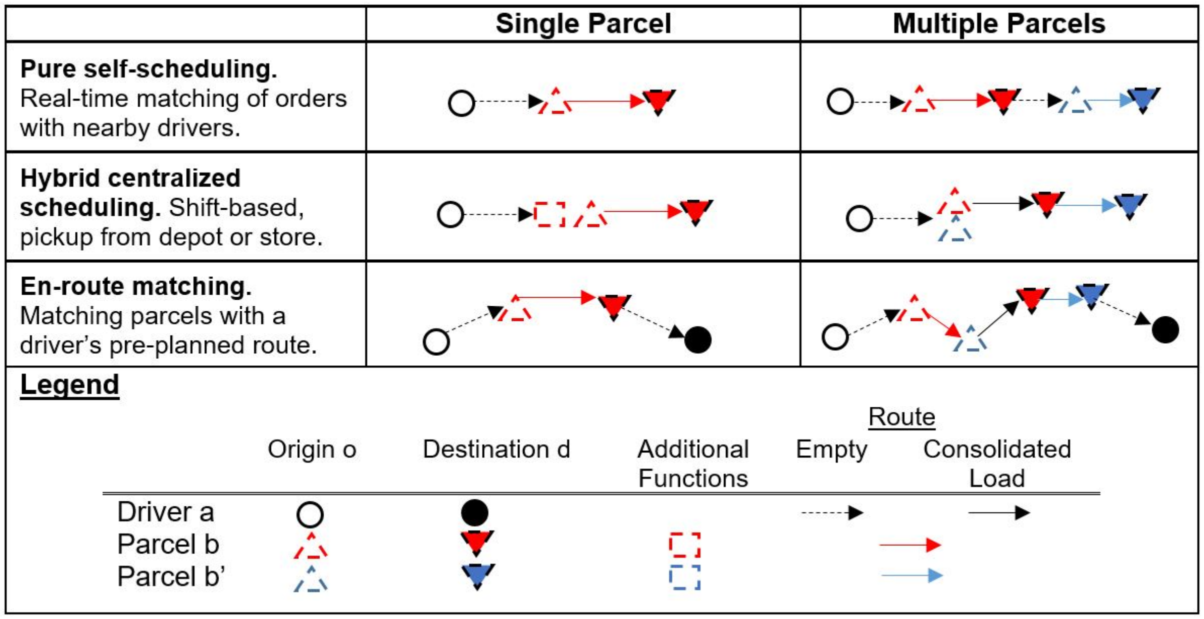

Single Parcel
Multiple Parcels
Pure self-scheduling. Real-time matching of orders with nearby drivers.
Hybrid centralized scheduling. Shift-based, pickup from depot or store.
En-route matching. Matching parcels with a driver's pre-planned route.
Legend
Route
Origin o
Destination d
Additional Functions
Empty
Consolidated Load
Driver a
Parcel b
Parcel b'

Figure 3.1. Matching schemes in crowdsourced delivery systems.

otherwise provide delivery services. The latter, however, provides alternative delivery solutions to enterprises or businesses that may utilize additional transportation channels for last mile delivery.

Observe that since shipments from local businesses typically have very short time windows (less than an hour), such delivery services are only offered through the pure self-scheduling mechanism. The fast-paced nature of the pure self scheduling mechanism makes it a good fit only for short-haul deliveries. On the contrary, long-haul deliveries may require additional lead time to find suitable driver matches. Therefore they are offered through en-route matching or bulletin-board matching mechanisms. Finally, centralized scheduling requires large amounts of demand for successfully combining delivery requests in cost-effective routes. Thus, this type of matching is more suitable as a last-mile delivery solution for well-established businesses with significant daily demands.

3.2.3 Compensation Schemes and Managing Supply of Drivers in Crowdsourced Delivery Systems

Crowdsourced delivery systems use varying compensation schemes to maintain a sufficient supply of drivers and to pay them. We classify those schemes into three main categories, and comment on challenges faced by the different approaches for compensation.

1. **Hourly compensation.** Amazon Flex is the only platform that uses just an hourly

Table 3.2. Target market of platforms under different matching mechanisms.

Matching Mechanisms	Target Market				
	Individuals			Business	
	Courier service		From Local Businesses	Short haul	Long haul
	Short haul	Long haul			
Pure self-scheduling	✓		✓	✓	
Centralized scheduling				✓	
En-route matching	✓	✓		✓	✓
Bulletin-board matching	✓	✓		✓	✓

rate to compensate crowdsourced drivers. A driver may earn more than the basic hourly rate of \$18/hour, if he/she has a larger capacity vehicle and is working under the *Amazon Logistics* division, which delivers regular Amazon packages with a 1 or 2-day delivery time window. For all other divisions, a driver may earn more through optional tips from customers. Amazon Flex manages the supply of drivers through its minimum pay guarantee program and centralized scheduling discussed earlier. The predictability in the compensation scheme is appealing to drivers, as drivers like to secure income during their scheduled hours. Combining its attractive compensation scheme with the centralized scheduling mechanism, ensures the balance of supply and demand at various times of the day.

A major challenge faced by a delivery system with an hourly compensation scheme is forecasting delivery needs and the number of drivers required to fulfill those deliveries, ahead of time. To offset such an obstacle, Amazon Flex offers real-time on-demand delivery blocks for drivers, in case scheduled drivers are not enough to meet current demand. However, those on-demand time blocks come with a higher level of risk, as the availability of drivers is not guaranteed. This may in turn reduce the service level through increasing the percentage of demands not met by the promised delivery time.

2. **Per-delivery compensation.** All other systems, with the exception of some systems in the category of manual bulletin-board-type matching, compensate drivers per completed delivery. As in the driver information on the official websites of the reviewed platforms, those platforms use formulas for computing the pay of drivers. Those formulas may depend on some or all of the following factors: mileage, wait time, size of package, traffic, and parking. The majority of systems under this payment scheme provide no payment guarantee for drivers, i.e. if a driver is available

but is not matched with a delivery task, then he/she does not receive any payment. Deliv, being an exception, guarantees a minimum pay for drivers even if they are not matched. DoorDash, which specializes in restaurant delivery, offers drivers occasional promotional guaranteed minimum pay during busy hours, around lunch and dinner time. As for systems with manual bulletin-board type matching, no payment guarantee is offered, but the pay is posted on the app with the rest of the delivery request information.

In general, for a system with a per-delivery compensation scheme and no payment guarantee, the availability of drivers and the maintenance of their loyalty are major challenges. In other words, drivers would prefer more predictability in their schedule and in their earning potential. That is, if they were to earn much less than the advertised earning potential, they may choose not to participate in this delivery system which may lead to a shortage of driver supply. This is especially true for platforms that often have multiple packages on a route delivered by the same driver. If he/she expects to deliver multiple packages, but gets only one or two, and is paid based on the number of packages, the resulting earnings may be significantly less than anticipated. So, to overcome the inconvenience of short-notice and unpredictability, drivers would choose these systems if, from their experience, they do get matched frequently during their available time, and are compensated well.

3. **Shipper and driver determine the compensation.** In some systems, a shipper and a buyer agree on a delivery price and the platforms receives a commission for matching them. This payment scheme is offered by some bulletin-board type matching systems where individuals are the target market. Such systems may transport many different types of items, either long-haul or short-haul. For instance PiggyBee (2019) supports international shipping, while Kanga (2019) focuses on local shipping, which may include bulky items. Platforms with such a compensation arrangement are typically community-based casual networks that do not deal with large numbers of daily deliveries. One major challenge for such a system is that matching is not guaranteed, which may deem the system unreliable.

3.3 OR Literature on Crowdsourced Delivery

In this section, we review the OR literature that explicitly addresses crowdsourced delivery. We compare the different logistics systems studied, the main decisions considered, the assumptions made, and the type of modeling used. We also point out how the studied logistics systems match with the delivery platforms in practice, as reviewed in Section 3.2.

To better comprehend how crowdsourced delivery fits within existing literature, we review representative papers in ride sharing and ride hailing, and highlight the similarities and differences between those problems and crowdsourced delivery.

3.3.1 Overview of Crowdsourced Delivery OR Literature

Archetti et al. (2016) were the first to model the problem of crowdsourced drivers in logistics networks by modeling the *vehicle routing problem with occasional drivers*. The paper considers an extension of the classical vehicle routing problem, by introducing the option of outsourcing part of demand fulfillment to what they called *occasional drivers*; in-store customers who are willing to make a delivery on their way home. The goal of the paper was to deliver some initial insight on the benefit of employing and further exploring crowdsourcing in logistics systems, so some assumptions may have oversimplified the problem. First, the proposed mixed integer programming model is static, and assumes that both demand and occasional driver availability is known before the planning period starts. Second, the model assumes that an occasional driver may be matched with a maximum of one delivery task, so as to avoid the need to consider routing.

Archetti et al. (2016) propose a multi-start heuristic to greedily assign customers to occasional drivers by solving a series of smaller scale integer programming problems that determine the subset of customers served by occasional drivers. Furthermore, the paper tests two compensation schemes, one based on only the delivery destination, and the other based on the deviation from a driver's original route or trip back home. Computational testing suggests that results are highly dependent on three factors: (a) the ratio of number of occasional drivers available to number of customers, (b) the flexibility of occasional drivers, in terms of the maximum deviation from their original trip that they are willing to drive, and (c) the compensation scheme employed. Macrina et al. (2017) later propose extensions to this problem by considering time windows of delivery and proposing two mixed integer programming models, one that allows for multiple deliveries per occasional driver, and another that allows for split deliveries. The authors argue that considering multiple deliveries per driver and split deliveries can provide significant cost savings, as compared to the vehicle routing problem with occasional drivers and additional time window constraints.

Arslan et al. (2018) study the crowdsourced delivery problem by proposing a variant of the dynamic pickup and delivery problem that considers *ad hoc drivers*. Similar to Archetti et al. (2016), the paper assumes that those ad hoc drivers are in-store customers who are willing to make a few stops on their way home. The paper assumes that a driver may complete more than one delivery task that may have different origins, not necessarily the store at which the driver was shopping. Three logistics networks, or geographies, are investigated: one-to-many, few-to-many, and many-to-many. One-to-many assumes that pickups are all in one location: the store at which the driver was shopping. Few-to-many assumes that the driver has to pick up packages from a small number of locations and not necessarily from the store at which he/she is shopping. Finally, many-to-many assumes that each delivery task has its own unique pickup and delivery locations. Ad hoc drivers and online orders arrive according to a uniform distribution, so the availability of drivers is uncertain and there is no scheduling.

Arslan et al. (2018) assume that a fleet of dedicated vehicles is always available, to guarantee the feasibility of the problem. An ad hoc driver indicates his/her stop-willingness, which is the number of stops he/she is willing to make on the way home. An exact recursive algorithm is proposed to determine all feasible driver-job matches, where a "job" is composed of one delivery task or up to four delivery tasks that are feasible to group together with respect to time constraints. Then, for jobs consisting of more than one delivery task, all feasible routes are enumerated, and some reduction techniques based on theoretical observations are used to reduce the number of possible routes. A heuristic is also proposed to eliminate non-promising jobs and routes, thus reducing the number considered. Once feasible matches and routes are determined, a matching problem is solved to find the best allocation of orders to ad hoc drivers or dedicated drivers. The authors note that the one-to-many geography turns out to be the most promising in terms of potential cost savings.

Similar to Arslan et al. (2018), Dayarian and Savelsbergh (2020) model the crowdsourced delivery problem as a dynamic problem. The paper allows matching at most one delivery task to an occasional driver, who is also assumed to be an in-store customer. Two dynamic models are proposed, a *myopic* one that does not consider any information on future arrivals of orders and drivers, and *sample scenario planning*, where arrival rates of drivers and online orders are used in addition to the system state to make decisions. A maximum weighted matching problem is solved to determine optimal matches of orders to drivers, where higher weight is given to urgent orders and orders that are far away from the store. Orders that are not matched to occasional drivers are fulfilled by the store's dedicated vehicles. A multi-trip vehicle routing problem with release and due times is solved to determine the optimal routes of dedicated vehicles. Because of the dynamic nature of the problem, Tabu search is used to quickly generate good vehicle routes in near real-time. Decisions are made at various decision epochs, where a decision epoch is (1) at fixed times throughout the planning horizon, and (2) upon every dedicated vehicle's return to the store. It is worth noting that the authors assumed the cost of delivering orders through occasional drivers is zero, and justified that by arguing that compensation can be through store credit.

Gdowska et al. (2018) model the crowdsourced delivery problem as a bi-level stochastic problem. The paper refers to in-store customers willing to make deliveries on their way home as occasional carriers (OCs). Unlike other reviewed papers, this paper assumes that OCs may choose to accept or reject possible assignments. At the first level of the proposed algorithm, a stochastic model is solved to determine the subset of customer orders to propose to OCs. Then, based on the rejection probability of the first stage, the remaining orders are fulfilled by the company's private fleet, by solving a capacitated traveling salesman problem. The proposed algorithm does not keep track of the allocation of orders to OCs, but rather determines the subset of customer requests to be offered to OCs. The authors develop a heuristic that calculates the cost of fulfilling all orders through the company's private fleet, then iteratively increases the subset of customers to be served by OCs until no more cost savings can be achieved. The paper suggests a basic pricing scheme that compensates drivers based on delivery location and parcel size, independent of the driver's final destination. The authors also acknowledge the necessity of investigating dynamic (or surge) pricing of OC's compensation. They offer an initial dynamic pricing scheme that correlates the compensation fee with the willingness of OCs to accept a task, based on historical data.

Soto Setzke et al. (2017) design an algorithm that matches drivers and transportation requests, based on drivers' already planned routes or daily commutes. The authors model the problem as a max-flow min-cost problem on a bipartite graph, where an arc exists if a request is time feasible relative to a driver's trip. The cost of each edge represents the additional time that a driver would need to carry out the delivery request. The goal is to establish an algorithm that gives drivers good matches of possible delivery requests, where a driver may pick up a delivery from any location and not necessarily a store or distribution center.

Qi et al. (2018) study the problem from an economics perspective and compare it to traditional shipping. The paper is the first attempt to design and analyze the prospective sharing logistics system based on analytical models and empirical parameter estimates. The authors study a network that contains multiple last-mile delivery terminals, that act as transshipment nodes. Inbound deliveries are shipped by the logistics service provider's trucks, while outbound shipments are completely carried by shared-mobility drivers. The authors assume the drivers are always available, and argue that one of the main features of shared mobility is its one-way one-shot nature; a car starts an outbound trip by approaching its first demand destination and the service ends once it drops off the last package.

Qi et al. (2018) develop a *continuous approximation* model to study the open vehicle routing problem faced by drivers. The authors also propose a wage response model that assumes a driver is willing to participate in a delivery service only if the payment is at least the amount that he/she can otherwise expect to earn by providing shared rides to passengers. The goal of this model is to characterize the cost of crowdsourcing shared mobility. Finally, the authors develop an equilibrium model for synergy and competition with the ride-share market, which aims to determine the supply of shared-mobility drivers. The authors conclude that shared mobility is not as economically scalable as the conventional truck-only system in terms of operating cost, unless the pool size of shared mobility keeps pace with the increase in demand density. Furthermore, the value of shared mobility is not in immediate operating cost savings, but rather the ability to reduce truck fleet size and the additional operational flexibilities it provides that may enable the cost-efficiency of the whole distribution system.

Literature Feature Comparison and Applicability to Real Crowdsourced Delivery Platforms

Let us compare the above reviewed papers in terms of the main features considered, the assumptions made to characterize the crowdsourced delivery system, and the applicability of those assumptions to the real platforms discussed in Section 3.2. Table 3.3 presents a summary of comparisons.

First, we note that all papers, with the exception of Qi et al. (2018) and Soto Setzke et al. (2017), assume that drivers are in-store customers who are willing to make stops on their way home. This setting is not currently applied in practice, as none of the crowdsourced delivery platforms reviewed in Section 3.2 employ drivers that way. Qi et al. (2018) consider the more applicable case of when drivers are hired for the sole purpose of making last-mile deliveries, but with flexible hours similar to ride hailing. This setting is close to hybrid centralized matching (Pattern 2), as discussed in Section 3.2.2, where many orders are consolidated at one pickup location. Soto Setzke et al. (2017) investigate the problem of matching regular commuters' trips to delivery tasks, which is similar to the *en-route matching*, pattern 3.

We also note that four of the seven reviewed papers assume a maximum of one delivery task per driver. This is more applicable for pure self-scheduling and en-route matching patterns. However, for the hybrid centralized matching pattern, where orders share the same pickup location, consolidating deliveries is preferred to achieve economies of scale.

In terms of decisions, all papers except Qi et al. (2018) and Gdowska et al. (2018) considered the decision of allocating orders to drivers. Different assumptions were made regarding the arrivals of customer orders and drivers. Gdowska et al. (2018) consider the problem of determining the subset of customer orders to offer to crowdsourced drivers, while Qi et al. (2018) study the economic impact of such a delivery system, rather than providing a decision-making tool for operational use.

Compensation of drivers was considered a decision variable by Qi et al. (2018) only in their wage response model. While a few papers (Dayarian and Savelsbergh, 2020 and Soto Setzke et al., 2017) did not discuss compensation at all, other papers assumed that it was based on the location of customers (Archetti et al., 2016 and Gdowska et al., 2018), deviation of the driver's pre-planned trip (Archetti et al., 2016, Macrina et al., 2017 and Arslan et al., 2018), or parcel size (Gdowska et al., 2018). Furthermore, Gdowska et al. (2018) investigated linking drivers' pay to their probability of accepting an order fulfillment task based on historical data, as an initial step in introducing dynamic pricing in crowdsourced delivery.

Let us now consider the development of crowdsourced delivery over time in both the industry and academic literature. We generate a timeline in Figure 3.2 that illustrates when the companies, discussed in Section 3.2, *were launched*, as well as the evolution of the academic literature that explicitly study crowdsourced delivery. We observe that many of the companies were founded years before the first academic work studied this system. Though the first company was launched as early as 2011, we speculate that it took the system a few years to become popular and commonly used, and that is when it gained the interest of researchers.

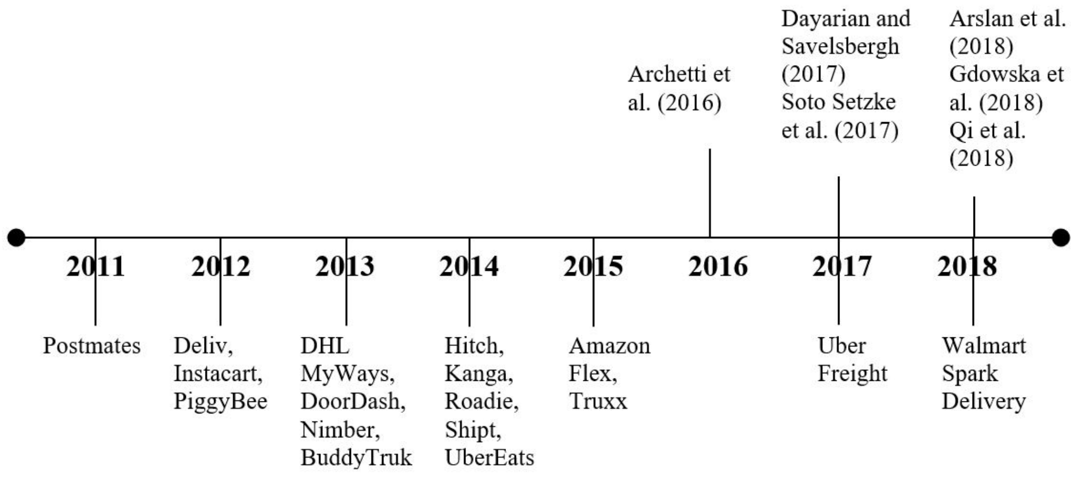


Figure 3.2. Timeline of industry practices and academic work.

Table 3.3. Comparison of different problem settings in crowdsourced delivery OR literature.

Paper	Problem Features			
	Occasional drivers	**Stop willingness**	**Compensation**	**Model**
Archetti et al. (2016)	In-store customers	1	Customer location, Trip deviation	Static MIP
Dayarian and Savelsbergh (2020)	In-store customers	1	Zero (store credit)	Dynamic model
Macrina et al. (2017)	In-store customers	Multiple	Trip deviation	Static MIP
Soto Setzke et al. (2017)	Regular commuters	1	Not discussed	Max flow min cost
Arslan et al. (2018)	In-store customers	2-4	Trip deviation	Dynamic model
Gdowska et al. (2018)	In-store customers	1	Customer location, Parcel Size	Stochastic model
Qi et al. (2018)	Hired for shared-mobility	Multiple	Wage response model	Continuous approximation, Stochastic model, Equilibrium model

3.3.2 Other Crowdsourced Delivery Studies and Hybrid Transportation Systems

The literature also discusses other types of crowdshipping systems. For instance, Kafle et al. (2017) study a hybrid network, where pedestrians and/or cyclists complete the last leg of a delivery task (or the first leg of a pickup task), while a truck carries on the rest of the delivery. The authors propose a mixed integer nonlinear program that selects crowdsourced individuals from a set of bids, and determines the relay points and truck routes and schedules.

A few studies have examined the concept of crowdsourced delivery from a qualitative perspective. One such study is by Rougès and Montreuil (2014), which extensively reviewed public documents of 18 crowdsourced delivery businesses and presented a typology of business models in the crowdsourced delivery industry. In contrast to our work, their proposed typology focuses on the business characteristics of the various companies, such as their business philosophies, revenue model, characters, etc. We also note that the list of crowdsourced delivery companies has notably changed since that paper was published, i.e., some of the reviewed companies are no longer in business, while new companies have launched in the last 5 years. Another qualitative study by Carbone et al. (2015) identifies different types of logistics in the sharing economy, by conducting an exploratory analysis of 32 cases.

Devari et al. (2017) consider the benefit of exploiting a social network in last mile delivery. The authors examine the results of a survey that aims to determine the willingness of people to make a delivery to a friend, on their way home. They then build a logistics regression model to determine the probability that a person would agree to make a delivery to a member of his/her social network. The study concludes that such a delivery scheme may greatly reduce emissions as well as last-mile delivery costs.

Hybrid Transportation Systems: Integrating Passenger and Freight Transportation

Researchers have also examined systems that combine transporting passengers and parcels. Li et al. (2014) study such a network, where people and parcels are transported via the same taxi network. The authors refer to the problem as *the share-a-ride problem* (SARP) and model it as a mixed integer linear program. Because of the computational demand of SARP, the authors propose a reduced problem, *the freight insertion problem* which starts with a given route for passengers, and inserts parcel requests into the route. The static and dynamic cases of both models are studied and analyzed. Later, Li et al. (2016) develop an adaptive large neighborhood search (ALNS) heuristic to find good quality solutions to the SARP in a short time. Another study, by Ghilas et al. (2013), examines the potential use of a public transportation network in freight transportation. Those authors propose an extension of the Pickup and Delivery problem (PDP), which they refer to as *the pickup and delivery problem with fixed scheduled lines* (PDP - FSL), that synchronizes delivery vehicles with scheduled city buses. Similarly, Fatnassi et al. (2015) and Masson et al. (2017) also study variations of the problem of integrating passenger and freight transportation. In their recent survey paper, Mourad et al. (2019) review the literature of passenger shared mobility, as well as shared mobility systems that combine people and goods.

3.3.3 Crowdsourced Delivery vs. Ride-sharing

Ride-sharing has received much attention in the OR literature for years. It refers to individuals sharing the excess capacity of their personal vehicles to transport passengers on their way to a pre-planned trip, such that a maximum distance or time deviation from their original route is maintained. This is not to be confused with *ride-hailing*, or activities of *Transportation Network Companies* like Uber and Lyft, where drivers use their personal vehicles to offer taxi-like services. Drivers in a ride-sharing system are performing an activity of self-interest, and are matched with riders on their way, with the ultimate goal of sharing the transportation cost and reducing their carbon footprint. Furuhata et al. (2013) present a classification of existing ride-sharing systems and challenges that prevent the widespread use of those systems.

Dynamic ride-sharing, in particular, has received considerable attention in the last decade. This problem deals with the dynamic matching of drivers and riders, such that the total system-wide traveled mileage is minimized. Agatz et al. (2011) propose an optimization-based approach to model the dynamic ride-sharing problem and suggest some implications of improved overall system performance. Agatz et al. (2012) later review the literature on OR in dynamic ride-sharing, and outline optimization challenges that arise in implementing ride-sharing technologies.

Later, Lee and Savelsbergh (2015), Stiglic et al. (2015) and Masoud and Jayakrishnan (2017) study extensions of dynamic ride-sharing problems that aim to improve the percentage of matches in the system in an effort to increase its reliability. Lee and Savelsbergh (2015) investigate the benefits of employing a small number of dedicated drivers in a ride-sharing network to serve riders that cannot be matched with ride-sharing drivers. On the other hand, Stiglic et al. (2015) assess the benefits of meeting points in a ride-sharing system, where riders may be picked up/dropped off at a location close to their origin/destination, in an effort to exploit any flexibility from the riders' side and increase the number of feasible matches in the system. Masoud and Jayakrishnan (2017) propose a fully flexible ride-sharing system that aims to maximize the number of served riders by allowing for some flexibility features such as multi-hop itineraries to riders. The authors argue that their suggested approach significantly increases the number of riders served.

We observe that the dynamic ride-sharing problem is closely related to the crowdsourced delivery problem when matching is done *en-route*, as explained in Section 3.2.2. That is, instead of drivers being matched with other passengers, drivers are matched with packages to deliver on their way. The main difference between the two systems is that a driver's utility may change in the two systems; a driver may prefer delivering parcels over driving passengers. Furuhata et al. (2013) notes that the building of trust in a network of unknown travelers is a challenge facing the widespread implementation of ride- sharing systems. We observe that this issue is less prominent in parcel delivery.

3.3.4 Crowdsourced Delivery Services vs. Ride-hailing

Ride hailing or services offered by Transportation Network Companies (TNC), like Uber and Lyft, have some similarities with crowdsourced delivery systems. One major similarity is that drivers are independent self-scheduling contractors in both systems. We observe that crowdsourced delivery platforms that fall under Pattern 1, pure self-scheduling and real-time matching, follow a similar matching mechanism as in ride hailing. Drivers indicate their availability by logging into the app; no prior planning of their hours is required.

Balancing supply and demand in ride-hailing systems is managed through the use of surge pricing. Now we will review some of the literature examining surge pricing in ride hailing services, then discuss the differences between ride hailing and crowdsourced delivery that pose challenges in implementing surge pricing in crowdsourced delivery platforms.

Banerjee et al. (2015) model pricing in ride hailing as a queuing network, and assume that drivers care more about their long term than their short term earnings. The authors show that a static policy is optimal if system parameters are deterministic, but a dynamic pricing policy is more robust to changes in system parameters. Bimpikis et al. (2019) analyze pricing for steady-state conditions in a network where drivers behave in equilibrium and decide whether and when to provide service and where to reposition to. They propose a “balance” property based on the demand patterns of the network and examine its implications for prices, profits and consumer surplus. Cachon et al. (2017) investigate various compensation schemes in a service platform with self-scheduling capacity. They conclude that though surge pricing is not the optimal compensation scheme, it is often close to optimal. They show that the optimal contract is one where both wages and prices are allowed to vary, but unlike surge pricing, there is no fixed ratio between the two. In addition, they argue that all stakeholders can benefit from surge pricing, in periods of high demand, as providers are better utilized, and consumers benefit from expanded access to the service. Finally, Castillo et al. (2017) point out that surge pricing can help eliminate inefficiencies in traditional transportation systems, namely the “wild goose chase” in which drivers’ earning are low due to long pick up times.

We observe that in ride hailing, surge pricing may increase the cost of a ride, and consequently the pay of a driver, in areas with high demand and low supply. This potentially balances supply and demand by attracting drivers to high paying regions and forcing customers who are not willing to pay a higher price to leave the system. To the contrary, in some crowdsourced delivery systems, such dynamic pricing might be more challenging to implement as all deliveries have to be made. No orders can leave the system in periods of high demand, and shipping costs should stay relatively stable to maintain a high level of customer service. In other systems, especially those that exclusively offer restaurant delivery such as UberEats, such surge pricing still exists. A customer ordering from a restaurant at a very busy time gets charged an extra variable fee, in an effort to balance supply and demand. This ensures system reliability, i.e. customers receive their order at the right time.

A question that arises in such a setting is how to stimulate drivers to increase their working hour flexibility, so as to ensure an adequate level of supply and create a reliable delivery system. One way to attract drivers is to offer a sufficiently high rate of compensation that maximizes their personal earning potential, while minimizing the transportation cost of the system.

3.4 Breakdown of Decision Problems within Crowdsourced Delivery Systems

In this section, we analyze the various managerial aspects within crowdsourced delivery systems and relate them to classical problems already studied in the literature. More explicitly, we define the decisions encountered in a crowdsourced delivery system and examine how these decision problems relate to other problems already investigated in the literature, and what additional challenges they bring about.

We break down such problems based on the matching and scheduling mechanisms indicated in Section 3.2.2. Note that we do not include bulletin-board type matching in our analysis since the mechanism operates completely manually. Such a system depends fully on the sharing of information and does not rely on any algorithm for determining optimal decisions. For each of the remaining three types, we identify subproblems within the system and review representative papers in the literature that address those subproblems. The objective is to highlight factors that distinguish those management decisions, in the context of crowdsourced delivery, from other contexts studied in the literature. This then helps identify promising research directions.

- **Pure self-scheduling:**
  Features: systems that follow pure self-scheduling are for rush deliveries with short time windows. Because of such small time windows, typically a driver is assigned one order at a time. Some platforms, such as Postmates, may match drivers with more than one order, if the driver agrees and if all delivery time windows can be met. However, during a given interval, the short time windows significantly restrict the number of possible matches, which eliminates the need for considering routing decisions. Therefore the main problems become:

  - Matching drivers and packages. This can be modeled as a *maximum weighted matching problem* as in the dynamic models proposed by Dayarian and Savelsbergh (2020) and Arslan et al. (2018). We note that the weight of a feasible driver-package match may represent different factors, such as the proximity of the driver, the rating of the driver, and additional issues that may give a driver priority. The weights may also consider other issues, such as not leaving a driver unmatched for a long time, so as to ensure that drivers will return to the app in the future.
  - Supply of drivers. Similar to ride-hailing, certain pricing mechanisms may be used to guarantee the supply of drivers at various times of the day. We note,

however, that the cost structure of the crowdsourced delivery service plays a major role in the possible level of variability in driver compensation. In other words, for platforms where customers pay a variable shipping fee, similar to surge pricing, driver wage may simply be a portion of that shipping fee which increases in periods of high demand. On the other hand, for platforms where the shipping fee is fixed (e.g. through a monthly or yearly subscription service), there is less flexibility in increasing driver pay, as the higher driver pay would be an extra cost for the platform, which may in turn affect the profitability and sustainability of the system.

- **Hybrid and centralized scheduling:**
  Features: when schedules of drivers are communicated well before the delivery due date, the problem becomes close to traditional delivery systems with dedicated drivers. Such a system allows for consolidation of delivery orders into routes to achieve transportation cost savings. The main problems under this setting are:

  - Matching and routing of drivers and packages. This can be represented by variants of the *vehicle routing problem* (VRP) if all packages are picked up from the same location (see Golden et al., 2008 and Toth and Vigo, 2014), or the *pickup and delivery problem* (PDP) if packages have different pickup locations (see Berbeglia et al., 2007). Variants of the VRP that may be a good fit for crowdsourced delivery systems with centralized scheduling include the *vehicle routing problem with time windows* to represent the delivery deadlines of packages, and the *heterogeneous fleet vehicle routing problem* to model the different levels of capacity of drivers' personal vehicles.
  - Scheduling of time blocks under uncertain demand. Because of the uncertainty in customer demand when driver time blocks are scheduled, advance determination of the number and duration of time blocks is a challenge for this system. Since scheduling is centralized, the problem resembles employee scheduling with demand uncertainty. Though the literature of employee scheduling in the service sector is rich (see Van den Bergh et al., 2013 for a comprehensive review), only a few papers study such a problem with demand uncertainty. Those papers also address particular application areas. For example, Bard et al. (2003) model a staff scheduling problem faced by the United States postal service, where the shifts of full-time workers and the number of part-time workers is determined before the exact demand is known. The paper assumes that demand is known with certainty a week ahead. Once demand is known, schedules are established by assigning overtime to full-time employees, assigning shifts to part time em-

ployees, and hiring casual workers as needed. Restrepo et al. (2017) study an employee scheduling problem when employees have identical skills. Their proposed model determines for each employee, their work days, days off, start and end of shifts, before demand is known. After demand is known, employees are assigned to jobs and breaks within their shifts. Bürgy et al. (2018) model the employee scheduling problem arising in retail stores, where assigning overtime work by extending employee shifts is possible, to cope with periods of high demand.

- **En-route matching:**
  Features: this system aims to maximize the number of matches, such that a driver's deviation from his/her original trip does not exceed a distance or time maximum value. As indicated earlier, this problem resembles dynamic ride-sharing problems. The main decision of this system is:

  – Matching of drivers and packages. This can be modeled as a *maximum weighted matching problem*, as proposed by Agatz et al. (2011) and Lee and Savelsbergh (2015), in solving dynamic ride-sharing problems. Such a model would guarantee the maximum profit (or least cost) that is feasible with regards to distance/time constraints. Another possible approach is formulating the problem as a *minimum-cost maximum-flow* problem, where cost represents distance or time, as suggested by Soto Setzke et al. (2017) in studying a crowdsourced delivery setting with en-route matching. The advantage of such approach is that the maximum matchings that guarantee the least distance/time deviation are found. Thus, this model is more suitable when there is no preference on feasible driver-package matches, other than minimizing distance or time traveled, so as to reduce the inconvenience faced by a driver in deviating from his/her original trip.

The former breakdown of decisions suggests that the main decisions faced in a crowdsourced delivery system can be categorized into four main classes: *matching, routing, driver scheduling* and *compensation*. In the following section, we identify promising research directions under each of those decision classes, that narrow the gap in the existing literature by considering specificities of crowdsourced delivery systems.

3.5 Future Research

We present below some promising future research directions in crowdsourced delivery, under each of the four decision classes suggested in the previous section.

1. **Matching Decisions.** Matching drivers and order requests is an important decision in pure self-scheduling and en-routing patterns. For crowdsourced delivery systems that follow those two patterns, it is interesting to consider different objective functions that reflect the multi-objective nature of matching faced in realistic applications. Such objectives include minimizing total fulfillment cost, giving higher priority to higher rated drivers, minimizing the time interval a driver is unmatched, and many others. Those objectives may be combined through a function, and may be given different priority levels. Using the *maximum weighted matching problem* with multiple objectives in a dynamic framework is an interesting direction to assess the implication of different objectives on the quality of the obtained matchings, in terms of profitability, driver utility, and other possible metrics.

   For the pure self-scheduling pattern, where drivers are matched to orders within a given radius, studying the effect of radius size as a hard constraint vs. a soft constraint is an interesting avenue. Hard constraints assume that it is not feasible to match a driver out of his/her radius, while soft constraints assume it is still feasible, but the match is penalized by the extra driving cost that the driver incurs. Assessing the effect of such constraints on the number of matches in the system, as a measure of the reliability of the system, is one possible goal of such a study.

2. **Routing Decisions.** As mentioned earlier, hybrid centralized scheduling is the only pattern that requires rigorous routing decision making, as it combines multiple packages in routes and plans drivers' availability ahead of time. Though drivers' schedules are planned in advance, systems that follow such a pattern in practice (e.g. Amazon Flex) only require a minimum vehicle capacity size, and do not plan for each individual driver's vehicle size. As such, it is interesting to investigate routing decisions when vehicle capacities are robust (i.e. within a range of values), and customer demand in the network is uncertain. This will enable the creation of routes that are more robust with respect to the different uncertainty parameters, rather than planning only for the minimum capacity level.

3. **Scheduling Decisions.** Systems with hybrid centralized scheduling are faced by the decision of determining the duration of drivers' work shifts, as well as the number of drivers needed for each shift, before demand is known. The literature on employee

scheduling with uncertain demand is application-specific, as explained in Section 3.4, and assumes that employees are regular rather than crowdsourced independent contractors. Thus, proposing scheduling models, with uncertainty considerations, that apply to crowdsourced delivery, and possibly extend to other crowdsourcing services, is a promising research direction.

4. **Compensation Decisions.** Since drivers are independent contractors, determining the appropriate level of compensation that guarantees the availability of those drivers, while maintaining the system's profitability, is an important decision in all crowdsourced delivery systems. Thus, studies that aim to find optimal compensation decisions, under the different crowdsourced delivery systems, are promising and essential. In fact, the main distinguishing factor between crowdsourced delivery systems and traditional systems is that drivers are independent contractors. Therefore, studying optimal compensation of drivers that induce more time and distance flexibility from the drivers' side, so as to ensure system reliability without compromising its profitability is critical. This also includes deciding on the optimal price from the customer side, which affects the supply-demand balance of the service network. Also, other possible tools for balancing the supply and demand within the service network is an important and interesting direction, explored in Chapter 4.

   Related to compensation, is the issue of driver welfare in crowdsourced delivery. Since drivers are independent contractors, they are not entitled to employee regulatory protection under labor law, such as minimum wage. Thus, an important direction, is how do we maintain the flexibility of the sharing economy while improving the welfare of drivers. This question is investigated in Chapter 5.

3.6 Conclusion

This chapter analyzes the current industry trends in crowdsourced delivery, and provides a taxonomy of available systems based on their scheduling and matching mechanisms, their target markets, and compensation schemes. The taxonomy introduced can help clarify how the various crowdsourced delivery systems differ, and ultimately aid researchers in validating the realism of their assumptions when examining this emergent transportation system. A review of the academic literature on this topic is also presented and a comparison between crowdsourced delivery, ride sharing and ride hailing is discussed. Because of the limited literature that explicitly examines this problem, we suggested a typology of decisions within a crowdsourced delivery system, and compared those decisions to classical problems in the literature. We highlighted the new challenges that crowdsourced delivery brings about, and proposed various directions for future research to narrow down gaps in the literature.

We observed that the pure self-scheduling matching mechanism is employed when delivery time windows are very short, typically an hour or less. This matching mechanism works similar to ride-hailing services, but delivers packages instead of transporting passengers. We explained how the use of surge pricing in the context of crowdsourced delivery poses more challenges as compared to ride-hailing, and highlighted future research on compensation schemes as a promising avenue. We also noticed that hybrid and centralized matching is utilized by e-retailers with large amounts of daily demand, and is close to traditional employee scheduling as driver availability is planned well in advance. En-route matching was found to be closest to ride-sharing, since drivers are completing an activity of self-interest, and are willing to make only a small time and distance deviation from their original trip.

With regard to the academic literature, we noticed that most of crowdsourced delivery studies consider assignment and matching decisions, and overlook the other decisions faced by the system. In comparing the literature's assumptions to the platforms in practice, we noticed that some of the assumptions are far from reality, specifically relating to how those drivers are employed. Most literature assume that crowdsourced drivers are in-store customers, but this idea was never implemented in practice.

Based on the taxonomy suggested, we highlighted multiple promising areas for future research that fall under four main decision categories: *matching, routing, scheduling,* and *compensation.* We described multiple directions that have the potential to improve the accuracy of decision support tools for this delivery system, by considering more of the realistic constraints in the system. The proposed future research focuses on *operational decisions* (matching and routing), as well as *tactical decisions* (scheduling and compensation). Yet we hope that the comprehensive analysis in this chapter will inspire researchers to think of strategic, innovative, and practical ways to enhance crowdsourced delivery in future years.

Chapter 4

Heatmap Design for Crowdsourced Delivery

4.1 Introduction

Crowdsourced delivery is a new trend in last-mile delivery that uses freelance drivers to transport goods and parcels. This emerging transportation solution uses the concept of shared mobility, similar to ride-hailing and ride-sharing, with the fundamental difference of delivering packages rather than giving rides to passengers. Unlike ride-sharing, where drivers are completing a trip of self-interest and are matched with riders on their way, crowdsourced delivery tasks, in most cases, are dedicated trips done for the sole purpose of delivery. Thus, the system shares more commonality with ride-hailing, which includes services like Uber and Lyft. Since drivers are paid per *gig* (task), the platform does not directly control their movement. Surge pricing with fixed commission contracts have been widely used in ride-hailing platforms to balance supply and demands, where the wage of a driver is a fraction of the price paid by the customer. However, crucial differences between ride-hailing and crowdsourced delivery makes this type of dynamic pricing not always a feasible option for crowdsourced delivery platforms. For instance, many crowdsourced delivery platforms rely on membership fees or have a fixed delivery charge throughout the day (e.g., DoorDash, 2021). Customers' price sensitivity is higher in such a setting than in ride hailing, and they are not willing to pay a significantly higher delivery charges when driver supply is limited (Titone and Goch, 2018). A question that naturally arises from the emergence of this system is: *What tools can a platform use to balance supply and demand in a crowdsourced delivery system, and how effective are they?*

In this chapter, we focus on one main tool: heatmaps, which are used in practice through the platform's app to communicate driver shortage across the service region. A heatmap comprises a heat level at each zone within a service region, the higher the heat level in a zone, the higher the likelihood that a driver, located in that zone, would be matched with an order. Thus, it informs drivers of areas with driver shortage where they are more likely to receive a revenue-generating match, which triggers probabilistic driver movement, ultimately better balancing supply and demand. Heatmaps may be associated with a financial incentive, such as a higher revenue per matching. In this research we propose a general framework that is independent of the particular compensation scheme a platform offers drivers. Instead, we model heatmaps as a decision that a platform controls, which results in the probabilistic relocation of drivers as a response.

We study a setting in which a platform, aiming to maximize service level throughout a planning horizon, has two main control levers for balancing supply and demand within the service region: (a) matching drivers and demand and (b) selecting a heatmap that influences relocation decisions of unmatched drivers. The goal of this research is to propose a novel formulation that models heatmaps as a control lever, then study the effectiveness of heatmaps in balancing supply and demand of a crowdsourced delivery system. We aim to identify the main factors that enhance service level in this dynamic setting, as well as to quantify how well heatmaps can influence drivers to move to regions where they are most needed, relative to a benchmark model where drivers can be directly managed. More particularly, we study a crowdsourced delivery platform that provides same day delivery solutions to local stores following the pure self-scheduling matching pattern as discussed in Alnaggar et al. (2019). Under this matching pattern, driver availability as well as customer demand is unknown in advance, but may be characterized probabilistically. Orders originate from multiple pickup points (e.g. local stores) and not a single warehouse. Delivery time windows are typically very short, about an hour long, and thus there is limited opportunity for consolidation. Examples of platforms in practice operating under this setting include Postmates, DoorDash, UberEats.

This research develops a Markov Decision Process (MDP) model that captures the dynamic and uncertain nature of crowdsourced delivery operations. At a particular point in time, the model retains information on the set of active drivers and their locations, as well as the set of all orders, their origins and destinations. Then for a given decision epoch, the model decides on the best allocation between drivers and orders as well as the best heatmap. Decision epochs represent discrete points in time when the platform makes those decisions. As a response to a heatmap, unmatched drivers within the network may choose to relocate to other regions, stay in the same region, or exit the system. Inactive drivers may choose to join the system and be considered for matching in areas with supply shortage, determined through the heatmap that they see in the platform app. The response of drivers to a given heatmap is characterized as a transition matrix with known probabilities.

In this chapter, we make several key contributions. (1) We introduce a new problem that describes a new modeling appraoch to a realistic setting faced by crowdsourced delivery platforms that aim to maximize service level by balancing supply and demand across service regions. We present a Markov Decision Process (MDP) model to formulate the problem, accounting for specific features such as delivery time windows, varying travel time between origin-destination nodes, and service time for preparing orders. To the best of our knowledge, this work is the first attempt to formalize the use of heatmaps in a decision model. (2) We then present analytical results that reduce the state and action spaces of the MDP model when orders are homogeneous in terms of order fulfillment priority, enabling reduction of the solution space. (3) We propose and formulate a stochastic look-ahead policy that utilizes properties of the problem to efficiently solve the problem without the need to solve the full MDP model. We also propose a simple heuristic policy for prescribing a heatmap, given the state of the system. (4) Finally, we conduct computational experiments generated from a real-world dataset to assess the proposed solution approaches against two benchmark models, and evaluate the benefit of using heatmaps. We find that the effectiveness of heatmaps in improving service level is most notable when demand patterns within a service region are imbalanced, and when the number of drivers in the network exceeds average demand.

This chapter is organized as follows. Section 4.2 reviews relevant literature. Section 4.3 describes the problem and introduces the proposed MDP model. Section 4.4 develops the proposed solution methodology and formulates the stochastic look-ahead policy. Section 4.5 explains the computational experiments and presents computational results. Finally, the chapter is concluded in Section 4.6.

4.2 Literature Review

This work falls under the growing body of literature addressing *same day delivery*, a class of the broad *last-mile delivery* problem with the additional challenge of shorter delivery time windows. Particularly a delivery task needs to be completed the same day it is announced. This problem has been studied under different settings and assumptions, especially concerning the mode of delivery. Most commonly, same-day delivery is assumed to be completed by a privately owned fleet of trucks, where the underlying problem is a variation of the vehicle routing problem with time windows or its dynamic counterpart (e.g., Voccia et al. (2019), Ulmer et al. (2019)). Researchers have also investigated the viability of integrating goods and passengers transportation to efficiently complete last-mile deliveries. Studies have examined the benefit of utilizing passenger city transit for last mile deliveries (see Fatnassi et al. (2015), Ghilas et al. (2016) and Masson et al. (2017)), as well as idle capacity in taxis (see Li et al., 2014), and have concluded that such settings can achieve cost and energy savings compared to traditional ones.

Crowdsourced delivery is one trend that has gained the attention of researchers in recent years. In contrast to the traditional use of trucks, crowdsourced delivery enables more efficient short-time-window delivery completion through the use of freelance drivers. Though it provides economic advantages over traditional delivery systems, particularly for being an *asset-light* transportation model (Qi et al., 2018), it introduces the additional challenge of uncertainty in driver supply. That is because a platform cannot *directly* bring more drivers to work or increase their supply when there is shortage. Alnaggar et al. (2019) provide a recent comprehensive review of the operations research literature addressing crowdsourced delivery, as well as a review of the main trends of this system in practice.

Many papers that study crowdsourced delivery assume a supply of both crowdsourced drivers and full time employees, and model the problem as a variant of the *vehicle routing problem* (e.g., Archetti et al. (2016), Macrina et al. (2017) and Dayarian and Savelsbergh (2020)) or the *dynamic pickup and delivery problem* (e.g., Arslan et al. (2018)). Those studies also assume that crowdsourced drivers are in-store customers, a setting that has not been implemented in practice, although considered by some companies (e.g., Walmart), as the papers suggest.

Ulmer and Savelsbergh (2020) study a different angle of the crowdsourced delivery problem, in particular workforce scheduling, in a setting where deliveries may be completed by both crowdsourced drivers and company drivers. Given the uncertainty in supply of crowdsourced drivers, and the time those drivers are available to complete deliveries, the paper aims to find a minimum cost schedule of company drivers that guarantees a desired service level. Unlike our problem, the paper assumes an environment where all orders originate from a warehouse.

Mousavi et al. (2020) consider a last-mile delivery system that uses mobile depots and crowdsourced drivers. The paper assumes that crowdsourced drivers are commuters, whose availability is stochastic, and are willing to deliver packages on their way to a trip of self-interest. The goal is to find the optimal location of mobile depots and the optimal allocation of orders to crowdsourced drivers. Qi et al. (2018) study a similar transshipment network structure that assumes multiple terminals where crowdsourced drivers can pick up consolidated orders. The paper assumes that drivers, hired for the sole purpose of completing deliveries, are not necessarily commuters. The aim is to assess the large-scale integration of shared mobility in logistics networks. Lei et al. (2020) investigate the problem of dynamically acquiring crowdsourced delivery workers by sending invitation requests such that supply level matches demand as closely as possible.

In contrast to the literature, this work assumes a *pure self-scheduling* pattern, which resembles the setting of ride-hailing problems, where orders may originate from any point in a pre-specified service region, such as a local business or restaurant. As such, our problem description adopts slightly different assumptions, the most important of which is that the delivery time window is very short. That is, platforms that employ a pure self-scheduling pattern are often third-party delivery services that provide same-day delivery to a variety of local businesses. Examples of such platforms include Doordash and Postmates. Those platforms are characterized by their very short delivery time windows, typically an hour long, and the varying pickup locations, making order consolidation especially challenging. Thus, a crowdsourced driver typically completes a single order delivery at a time.

This research is also relevant to the stream of literature on surge pricing. Bimpikis et al. (2019) study the impact of spatial pricing in ride hailing networks and derive conditions under which spatial price discrimination maximizes a platform's revenue. The authors study the system in steady state and do not account for temporal variability. Besbes et al. (2021) study the short term spatial pricing problem, in particular how to optimally set prices within a continuous service region so as to maximize the platform's revenue. Hu et al. (2021) investigate the temporal aspect of surge pricing in ride-hailing, particularly that drivers and riders respond to surge pricing on different timescales, and identify two types of equilibrium pricing strategies. Lu et al. (2018) empirically analyze whether short-run variations in surge prices attract drivers. Using a difference-in-differences approach, the authors show that the ability to see the surge heatmap has a statistically significant impact on drivers' decisions to reposition and drivers' revenue. In contrast to this research, the authors study the drivers' problem and model it as a multinomial logit discrete choice model. This work investigates the problem from the platform's perspective and captures the behavior of drivers as a stochastic function of the chosen heatmaps.

Surge pricing as an incentive mechanism assumes that this additional revenue enabling price increase comes from the customer. This gives the platform more flexibility to vary prices, and encourages price-sensitive customers to exit. In a crowdsourced delivery setting additional revenue is typically not collected from customers and delivery is expected to be somewhat stable, and thus customers do not exit the system when driver availability is limited.

Even if we associate a price to each heat level, in contrast to surge pricing we consider a set of discrete heat levels at each node, then decide on the complete heatmap. By doing so, we capture the interdependence between nodes when assigning them heat levels, i.e., a heat level at a given node not only attracts/repels drivers to/from that node, but also affects flow to neighboring nodes. This contrasts to the literature on spatial surge pricing (over a network) which assumes an infinite supply of drivers (Bimpikis et al., 2019), i.e., drivers continue to join the system as long as their expected earning satisfies some equilibrium constraints. In a way we can think of the heatmap design problem as a generalized framework that allows us to focus on the effect (probabilistic movement) of drivers rather than assume a particular payment structure that triggers their movement (e.g. fixed commission contract). Our proposed model also considers a more general setting where demand may be time-varying, thus capturing both spatial and temporal effects of demand and supply variations.

4.3 Markov Decision Process (MDP) Model

We consider a last-mile delivery network operated by a crowdsourced delivery platform, where orders and drivers arrive to the system randomly throughout the day. Orders have known origins and destinations and are featured by a very short delivery time window. At discrete points in time throughout the day, i.e., *decision epochs*, the platform decides on the allocation of orders to active drivers, as well as the choice of heatmap. The latter triggers a probabilistic movement of active unmatched drivers and inactive drivers to regions with driver supply shortage.

Let N denote the set of nodes, representing *centers* of geographic regions and A is the set of arcs connecting adjacent nodes. At decision epoch t, the platform keeps track of the set of drivers, $\mathcal{M}_t$, with information on their location and status (active, en-route, inactive). The platform also keeps track of the set of active orders $\mathcal{D}_t$, which specifies the origin and destination of each order, as well as the delivery deadline.

For a given decision epoch, the platform matches orders to drivers so as to maximize the number of fulfilled orders in immediate and future epochs; this is a proxy for maximizing the service level of the platform. A driver at node i who is assigned to an order with destination j at epoch t will be at node j at epoch $t + \tau_{ij}$, where τ_{ij} is the travel time between nodes i and j. Thus, matching decisions influence driver availability in the future. Since drivers are freelancers who may enter/exit the system anytime they wish, the platform communicates to them about regions with driver shortage, i.e., those where drivers have a higher chance of being matched, by using a *heatmap*. This encourages drivers to reposition to areas where they are needed most.

To account for drivers joining or exiting the system throughout the planning horizon, we add an auxiliary node (denoted as node 0) that captures the set of drivers that enter or leave the platform. That is, a driver may exit the system and decline any proposed deliveries by repositioning to node 0. Similarly, a driver who is initially outside the system, may choose to join by repositioning from node 0 to any other node.

Before proceeding with the model formulation, we first provide a formal definition of a heatmap.

Definition 1. A heatmap h is a vector of size $|N|$ that takes integer values in the range $[1, \gamma]$. Each value is referred to as a *heat level*. A given choice of heatmap h triggers the probabilistic movement of available drivers in the network, characterized by transition matrix $P(h)$.

Note that we define all notation used in the model formulation and solution approach within text, and also provide a comprehensive notation summary in Appendix B.5.

4.3.1 State and Action Spaces

The state variable, or the state of the system S_t at decision epoch t, is the minimum amount of information that is necessary and sufficient to make a decision. This is characterized, in this problem, by the set of drivers in the network, $\mathcal{M}_t$, and the set of active orders, $\mathcal{D}_t$. An active order is an order that has been received, but not yet assigned to a driver, and it still may be delivered within its delivery time window. We assume that if an order is not delivered by the delivery deadline, it is lost or outsourced to a third party. This reflects the short time window of the crowdsourced delivery system. The state variable is then defined as $S_t = (\mathcal{M}_t, \mathcal{D}_t)$.

At a given decision epoch t, the platform makes two main decisions: (a) the matching of orders and active drivers and (b) the choice of heatmap. A particular heatmap $h \in H$, where H is the set of all possible heatmaps, induces the movement of unmatched drivers to neighboring regions with known probabilities. We assume that drivers only reposition to neighboring regions, in response to a heatmap, that are reachable in one decision epoch. We capture the repositioning probabilities for a heatmap h through a transition matrix $P(h)$, where p_{ij} represents the probability that a driver at node i repositions to node j. The number of unmatched drivers at $i \in N$, denoted as m^u_{ti}, that may choose to reposition (empty) to other regions is known at epoch t. Thus, the number of drivers at i that move to j is a multinomially distributed random variable with m^u_{ti} trials and p_{ij} probability of success.

We define the decision function $X^\pi(S_t) = (\boldsymbol{x_t}, \boldsymbol{y_t})$ as a rule for mapping state S_t to action vector $(\boldsymbol{x_t}, \boldsymbol{y_t})$, where π is a policy that indicates the type of the decision function, $\boldsymbol{x_t}$ and $\boldsymbol{y_t}$ are vectors specifying the matching and heatmap decisions, respectively.

We note that the probability of driver movement as a response to a heatmap is assumed to be known. The details and implementation of an approximation algorithm tailored to the problem at hand is out of the scope of this chapter and is left for future work. One way of estimating and updating the transition probabilities, used in the proposed optimization framework, is through Bayesian inference. For a given heatmap, we start with a prior belief, obtained from domain knowledge. We assume that each observation of driver repositioning is an independent trial. For a given node $i \in N$, we have $|N_i| + 2$ discrete choices of repositioning locations, each with an unknown probability. Those choices represent transitioning to adjacent nodes, staying in the same node, or exiting the system, where N_i is the set of nodes adjacent to i. Thus, the probability of transitioning is a multinomial distribution.

In Bayesian statistics, the parameter vector associated with a multinomial distribution is drawn from *Dirichlet Distribution*, which forms the prior distribution (Congdon, 2007). The Dirichlet distribution is characterized by the number of outcomes k and a concentration parameter α; a vector of positive real numbers. Increasing the magnitude of the concentration parameter α increases the level of confidence in the prior belief. The expected value of the posterior distribution of the probability of repositioning can then be used as the heatmap transition probability in the proposed optimization framework. Interested readers are referred to Congdon (2007) for more details on Bayesian modeling and inference.

Before proceeding with the rest of the MDP model, we provide a toy example that illustrates the interaction between matching and heatmap decisions.

Toy Example

Example. Consider a two-node, two-period problem, and assume that the number of drivers is fixed (case 1). At $t = 1$, the number of drivers at each node is $m_1 = \begin{bmatrix} 9 \\ 3 \end{bmatrix}$. The active orders for each o-d pair is $d_1 = \begin{bmatrix} 0 & 5 \\ 10 & 0 \end{bmatrix}$.

To maximize demand fulfillment, the platform chooses the following matching decisions $x_1 = \begin{bmatrix} 0 & 5 \\ 3 & 0 \end{bmatrix}$. This results in the unmatched drivers $m_1^u = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$. Assuming a travel time of 1 epoch, as a result of matching, the initial distribution of drivers at $t = 2$ is $m_2^{in} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$.

When no additional orders are received, the remaining orders at $t = 2$ are $d_2 = \begin{bmatrix} 0 & 0 \\ 7 & 0 \end{bmatrix}$.

Since the initial supply of drivers at node 2 as a result of matching is not enough to cover the remaining orders, the platform chooses a heatmap that triggers the movement of unmatched drivers from node 1 to 2. For instance heatmap, $h = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ results in transition matrix $P(h) = \begin{bmatrix} 0.1 & 0.9 \\ 0 & 1 \end{bmatrix}$. Thus, the unmatched drivers in node 1 move to node 2 following a binomial distribution with probability $p_{12} = 0.9$. A random sample of such distribution indicates the following repositioning $u_1 = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}$. Thus, at $t = 2$ the number of drivers is $m_2 = \begin{bmatrix} 3 + 1 = 4 \\ 5 + 3 = 8 \end{bmatrix}$, which covers the remaining orders.

4.3.2 Post-decision State, Exogenous Information, Contribution

The post-decision state refers to the state of the system immediately after a decision has been made, but before arrival of any new information. In the considered problem, this refers to the destination of *matched* drivers and the number of active unfulfilled orders.

Exogenous information, on the other hand, is the information that becomes available to the platform between two consecutive decision epochs. This constitutes the realization of the random variables of driver repositioning as a response to a chosen heatmap and the arrival of new orders. The arrival of orders for an origin-destination (o-d) pair (i, j) is assumed to follow a known distribution. Let W_t denote new driver and order information that first becomes known between t and $t + 1$. Given the post-decision state and the realization of exogenous information (W_t), the system transitions to the next state S_{t+1}, denoted as the *pre-decision state*. Let $C(S_t, (\boldsymbol{x_t}, \boldsymbol{y_t}))$ denote the contribution (reward) of making decision vector $(\boldsymbol{x_t}, \boldsymbol{y_t})$ when in state S_t. Let x_{tij} denote the number of orders, matched at t, with o-d pair (i, j). The contribution is defined as

$$C(S_t, (\boldsymbol{x_t}, \boldsymbol{y_t})) = \sum_{i \in N} \sum_{j \in N} x_{tij}.$$

We use the *expected total reward* criterion for computing the optimal policy. Let $v^{\pi}(S_t)$ denote the expected reward from epoch t to the end of the planning horizon T, and Π denote the set of all possible policies. We set a terminal reward at decision epoch T, $C(S_T) = 0$, for all states S_T, which represents the end of a day when no orders are accepted.

$$v^\pi(S_0) = \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} C(S_t, X^\pi(S_t)) | S_0 \right\} \tag{4.1}$$

The objective is to find an optimal policy π^* such that the expected total number of fulfilled orders is at least as good as any other policy; $v^{\pi^*}(S_0) \geq v^\pi(S_0)$ for all $\pi \in \Pi$.

4.3.3 Reduction of the Dimension of State and Action Spaces

Customer demand in crowdsourced delivery systems is typically homogeneous in terms of order fulfillment priority. While the objective is to fulfill as many orders as possible, no order is prioritized over another, except based on their delivery deadline. This characteristic enables us to prove a proposition that simplifies the state and action spaces at each epoch t.

Recall that x_{tij} denotes the number of (i, j) orders matched at epoch t. Define $\mathcal{D}_{tij} \subseteq \mathcal{D}_t$ as the set of orders with o-d pair (i, j) that are active at epoch t, and which is ordered in increasing order of delivery deadline. Let $\mathcal{X}_{tij}$ be the set of fulfilled orders at epoch t chosen as the first x_{tij} elements of the ordered set $\mathcal{D}_{tij}$, and let $\mathcal{X}'_{tij}$ be an arbitrarily chosen subset of $\mathcal{D}_{tij}$. Recall that $v^\pi(s_t)$ denotes the reward from decision epoch t up to the end of the planning horizon, following policy π, and W_t is a realization of the random exogenous information between $t-1$ and t.

Proposition 1. For a given choice of heatmap h at t, and a given realization of exogenous information W_{t+1}, $v^\pi(S_{t+1}|S_t, x_t = \mathcal{X}_{tij}, y_t = h) \geq v^\pi(S_{t+1}|S_t, x_t = \mathcal{X}'{}_{tij}, y_t = h)$.

In words, choosing set $\mathcal{X}_{tij}$ of matched orders, over any other arbitrary set $\mathcal{X}'{}_{tij}$, guarantees that for a given policy, from decision epoch $t+1$ onward, the accumulated reward $v^\pi(\cdot)$ is at least as good as that obtained from any other subset. We present the proof in Appendix B.1.1.

Proposition 1 enables us to simplify the decision variable at each decision epoch under the assumption of the homogeneity in demand fulfillment priority and drivers. That is, any driver at node i matched to any order with an o-d pair (i, j), would result in the same reward. It suffices for the platform to decide on the number of (i, j) orders to be matched at each decision epoch, rather than solving for the explicit matching between these two sets, then fulfill orders in an increasing order of delivery deadline. However, presenting the more general form lays the foundation for future work on variations of the problem that may consider the explicit assignment of drivers to orders and other features of the problem, such as the bundling and routing of orders under short time windows.

We note that this assumption also imposes the implicit assumption that the exit probability of the aggregate number of drivers in a node is independent on the time each specific driver joins the system. We argue that since we consider the probabilistic movement of the aggregate number of drivers in a node, the time aspect may be approximately factored in the computation of the transition probabilities. For instance, from historical data, the platform may know that a certain time of the day is unfavorable to drivers and thus adjusts the transition probability to account for the higher exit probability.

4.4 Solution Methodology

Even with the simplification of decision variables resulting from proposition 1, the model suffers from the *curse of dimensionality* in state, action and outcome spaces. The underlying uncertainty in both driver availability and demand creates a very large state space S_t at each epoch. This, when combined with the extensive range of feasible actions for each state and epoch, makes the model computationally intractable to solve to optimality by Bellman's equation, for reasonable size problems. Thus, we turn to a stochastic look-ahead (SLA) policy as an alternative solution methodology for sequentially finding the optimal heatmap as new information is revealed. SLA is an approximate dynamic programming policy that is used to obtain good approximate solutions for MDPs suffering from the curse of dimensionality (Powell, 2011).

A stochastic look-ahead policy is a rolling horizon framework that selects decisions at epoch t, while considering possible realizations of random information within a forecast window. The platform's problem is formulated as a two-stage stochastic program, where information at epoch t is known with certainty, while future information within a forecast window $\{t+1, \dots, t+\Gamma\}, t+\Gamma \leq T$ is only known probabilistically when decisions are made, where Γ is the duration of the forecast window. Under an SLA policy, the platform's primary focus is to decide on actions for a particular decision epoch t; decisions for future periods are only incorporated to account for the effects of decisions made at t on subsequent time periods. Recall that the dependence of decision epochs is contributed mainly to the fact that the destinations of matched drivers at t determine (in part) drivers' availability in consecutive time epochs, and that it takes one decision epoch for the effect of the heatmap (movement of drivers) to materialize. Our proposed solution methodology formulates a two-stage stochastic program to determine the best matching and heatmap decisions. To improve computational efficiency, we decompose matching and heatmap decisions into two separate optimization problems. Figure 4.1 is a visual overview of the proposed solution approach, which we detail next.

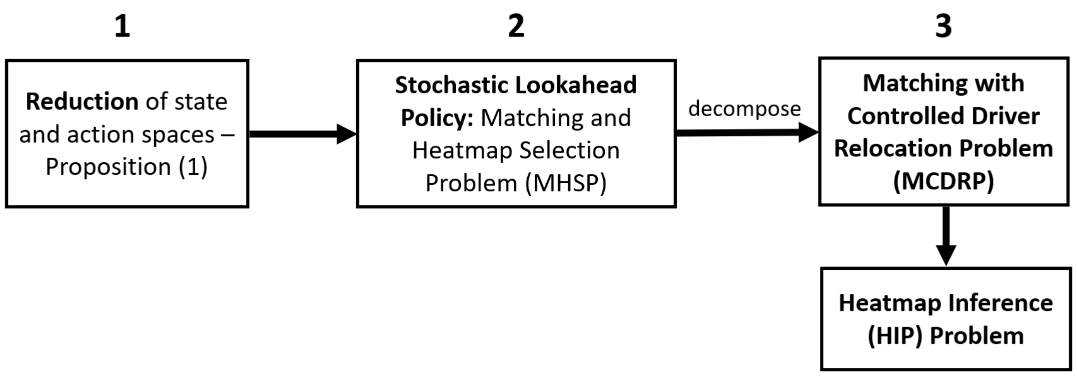

1
2
3
Reduction of state and action spaces – Proposition (1)
Stochastic Lookahead Policy: Matching and Heatmap Selection Problem (MHSP)
decompose
Matching with Controlled Driver Relocation Problem (MCDRP)
Heatmap Inference (HIP) Problem

Figure 4.1. Overview of the proposed solution approach steps.

4.4.1 Matching with Controlled Driver Relocation - an Upper Bound

At a given decision epoch t, the decision concerning the choice of heatmap and number of orders to match can be expressed as an optimization problem that maximizes the number of matches at decision epoch t, plus the expected number of matches in decision epochs within the forecast horizon $\{t+1,\ldots,t+\Gamma\}$. We first formulate the problem faced by the platform assuming that it has full control over the movement of drivers, and is able to directly request drivers to move to locations where they are needed. That is, instead of relying on the probabilistic responses to heatmaps, the platform directly controls the relocation of drivers. The purpose of this model is twofold. First, it enables us to evaluate the effectiveness of heatmaps in managing the flow of drivers by creating a *benchmark* of the best case scenario. Since orders have a short matching time window, fulfilling all orders at each decision epoch may not be possible. Thus, the matching with controlled driver relocation problem (MCDRP) computes the best possible matching for particular problem instances, allowing us to accurately assess the potential of heatmaps in balancing supply and demand. Second, this problem enables us to partially quantify the benefit, in terms of the improvement in service level, of treating crowdsourced drivers as employees rather than freelancers in terms of the improvement in service level. That is, how much improvement in total matching will the platform gain, if drivers are treated as employees who can be directly managed and moved in the network, rather than as independent contractors who respond probabilistically to heatmaps.

We first define the following notation. l is the index specifying the decision epoch, i.e., $l \in \{t, t+1, \ldots, t+\Gamma\}$. $\mathbb{S}$ is the set of samples of random orders considered in the SLA policy. x_{lij}, x^s_{lij} are decision variables denoting the number of matched known and forecasted orders, respectively, for an o-d pair (i,j). m_{li}, m^s_{li} are the decision variables for the number of drivers at each node i, in the first and second stages, respectively. The number of drivers is known at epoch t, but unknown for subsequent epochs. u_{lij}, u^s_{lij} denote the number of drivers that transition from node i to j between decision epochs l and $l+1$, in the first and second stages. $d_{t\omega ij}$ is a parameter representing the number of known active orders with (i,j) o-d pairs at decision epoch t with deadline at epoch $t+\omega$ where ω is the duration of the delivery time window. Similarly, $d^s_{l\omega ij}$ is a parameter for the number of forecasted orders with (i,j) o-d pairs in sample $s \in \mathbb{S}$, at decision epoch l with deadline at epoch $l+\omega$. d_{lkij}, d^s_{lkij} are decision variables for carried forward orders at epoch l that expires at epoch $l+k$, $k<\omega$, in the first and second stages, respectively. We also define service time Δ as the time duration, in decision epochs, it takes for an order to be ready for pickup (which may be set to 0). Finally, α is a scalar $\in (0,1)$.

We note that for ease of exposition, we use a fixed delivery time window ω for all orders. However, orders may have variable delivery time windows that are dependent on the store from which they originate or their o-d pair. This can easily be handled by updating the number of orders with a particular o-d pair and a given deadline, d_{tkij}, at the beginning of each decision epoch.

The problem is formulated as follows.

[MCDRP]

$$\max \sum_{l=t}^{t+\Delta} \sum_{i \in N} \sum_{j \in N} x_{lij} + \frac{1}{|\mathbb{S}|} \sum_{s \in \mathbb{S}} Q^s(m, d, x^s, m^s, u^s, d^s) \tag{4.2a}$$

$$\text{s.t.} \sum_{j \in N} x_{lij} \le m_{li} \qquad \forall i \in N, l \in \{t, \dots, t+\Delta\} \tag{4.2b}$$

$$x_{lij} \le \sum_{k=l+\tau_{ij}}^{l+\omega} d_{lkij} \qquad \forall i,j \in N, l \in \{t, \dots, t+\Delta\} \tag{4.2c}$$

$$d_{lkij} = d_{(l-1)kij} - \max\{(x_{(l-1)ij} - \sum_{k'=l+\tau_{ij}-1}^{l+k-1} d_{(l-1)k'ij}), 0\} \qquad \forall i,j \in N, l \in \{t+1, .., t+\Delta+1\}, \; k \in \{l+\tau_{ij}, .., l+\omega-1\} \tag{4.2d}$$

$$\sum_{j:(i,j) \in A} u_{lij} = m_{li} - \sum_{j \in N} x_{lij} \qquad \forall i \in N, l \in \{t, \dots, t+\Delta\} \tag{4.2e}$$

$$m_{li} = \sum_{j \in N} x_{(l-\tau_{ji})ji} + \sum_{j:(j,i) \in A} u_{(l-1)ji} \qquad \forall i \in N, l \in \{t+1, \dots, t+\Delta+1\} \tag{4.2f}$$

$$\begin{aligned} & u_{lij} \in \mathbb{Z}_{\ge 0} && \forall (i,j) \in A, l \in \{t, \dots, t+\Delta\} \\ & x_{lij} \in \mathbb{Z}_{\ge 0} && \forall i,j \in N, l \in \{t, \dots, t+\Delta\} \\ & m_{li} \in \mathbb{Z}_{\ge 0}, d_{lkij} \ge 0 && \forall i,j \in N, l \in \{t+1, \dots, t+\Delta+1\}, \\ & && k \in \{l+\tau_{ij}, .., l+\omega-1\} \end{aligned} \tag{4.2g}$$

where

$$Q^s(m,d,x^s,m^s,u^s,d^s) = \sum_{l=t+\Delta+1}^{\Gamma} \sum_{i\in N}\sum_{j\in N} \alpha^{(l-t)} x^s_{lij} \tag{4.3a}$$

$$\text{s.t.} \sum_{j\in N} x^s_{lij} \le m_{li} \qquad \forall i \in N, l \in \{t+\Delta+1\} \tag{4.3b}$$

$$\sum_{j\in N} x^s_{lij} \le m^s_{li} \qquad \forall i \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\} \tag{4.3c}$$

$$x^s_{lij} \le d^s{}_{l\omega ij} + \sum_{k=l+\tau_{ij}}^{l+\omega-1} d_{lkij} \qquad \forall i,j \in N, l \in \{t+\Delta+1\} \tag{4.3d}$$

$$x^s_{lij} \le d^s{}_{l\omega ij} + \sum_{k=l+\tau_{ij}}^{l+\omega-1} d^s_{lkij} \qquad \forall i,j \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\} \tag{4.3e}$$

$$d^s_{lkij} = d_{(l-1)kij} - \max\{(x^s_{(l-1)ij} - \sum_{k'=l+\tau_{ij}-1}^{l+k-1} d_{(l-1)k'ij}), 0\} \qquad \forall i,j \in N, l \in \{t+\Delta+2\}, \; k \in \{l+\tau_{ij},..,l+\omega-1\} \tag{4.3f}$$

$$d^s_{lkij} = d^s_{(l-1)kij} - \max\{(x^s_{(l-1)ij} - \sum_{k'=l+\tau_{ij}-1}^{l+k-1} d^s_{(l-1)k'ij}), 0\} \qquad \forall i,j \in N, l \in \{t+\Delta+3,..,t+\Gamma\}, \; k \in \{l+\tau_{ij},..,l+\omega-1\} \tag{4.3g}$$

$$\sum_{j:(i,j)\in A} u^s_{lij} = m_{li} - \sum_{j\in N} x^s_{lij} \qquad \forall i \in N, l \in \{t+\Delta+1\} \tag{4.3h}$$

$$\sum_{j:(i,j)\in A} u^s_{lij} = m^s_{li} - \sum_{j\in N} x^s_{lij} \qquad \forall i \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\} \tag{4.3i}$$

$$m^s_{li} = \sum_{j\in N} x^s_{(l-\tau_{ji})ji} + \sum_{j:(j,i)\in A} u^s_{(l-1)ji} \qquad \forall i \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\} \tag{4.3j}$$

$$u^s_{lij} \in \mathbb{Z}_{\ge 0} \qquad \forall (i,j) \in A, l \in \{t+\Delta+1,\dots,t+\Gamma\}$$

$$x^s_{lij} \in \mathbb{Z}_{\ge 0} \qquad \forall i,j \in N, l \in \{t+\Delta+1,\dots,t+\Gamma\}$$

$$m^s_{li} \in \mathbb{Z}_{\ge 0}, d^s_{lkij} \ge 0 \qquad \forall i \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\}, \; k \in \{l+\tau_{ij},..,l+\omega-1\} \tag{4.3k}$$

Objective function (4.2a) maximizes the number of matches of known active orders, plus the expectation of matched forecasted orders for decision epochs within the forecast horizon. The second-stage objective function (4.3a) maximizes a discounted value of forecasted orders matches, for decision epochs $\{t+\Delta+1,\dots,t+\Gamma\}$. The discount parameter $\alpha^{(l-t)}$ is inversely proportional to the length of time interval between the current epoch and the future period, i.e., the later the period the lower its weight. This discount factor recognizes that the farther ahead the decision epoch is, the higher the cumulative noise in order information and driver availability. It is important to note, however, that this objective does not contradict the proposed *expected total reward criterion* in the underlying MDP.

We note here that the linking variables between the first and the second stages are the number of drivers and the carried forward orders, m_{li} and d_{lkij}, $l = t + \Delta + 1$. Those are reflected in constraint sets (4.3b, 4.3d, 4.3f, and 4.3h). Constraints (4.2b) and (4.3b-4.3c) ensure that the number of matched orders with an origin i, do not exceed the number of drivers at that node, for both known and forecasted orders. Constraints (4.2c) and (4.3d, 4.3e) make sure that the number of matched orders is at most equal to the number of orders (new and carried forward) with an (i, j) o-d pair. Constraints (4.2d) and (4.3f, 4.3g) compute the orders that are carried forward between two consecutive decision epochs.

Constraints (4.2e) and (4.3h, 4.3i) guarantee the flow balance of drivers; total number of drivers that reposition to any node j, equals the number of unmatched drivers at the origin node i. Constraints (4.2f) and (4.3j) compute the number of drivers at subsequent decision epochs $l+1$ for each node i. Constraints (4.2g) and (4.3k) assign integer values to all variables with the exception of carried forward orders d_{lkij}, d^s_{lkij}, which are continuous.

To eliminate the nonlinearity in constraint (4.2d), we replace it with the following set of constraints:

$$d_{lkij} = d_{(l-1)kij} - \theta_{lkij} \tag{4.4a}$$

$$\theta_{lkij} \geq x_{(l-1)ij} - \sum_{k'=l+\tau_{ij}-1}^{l+k-1} d_{(l-1)k'ij} \tag{4.4b}$$

$$\theta_{lkij} \geq 0 \tag{4.4c}$$

Similar linearization is applied to constraints (4.3f, 4.3g).

4.4.2 Matching and Heatmap Selection Problem

We now describe the problem of concurrently optimizing heatmap selection and matching decisions so as to maximize the expected matching within the forecast horizon. We refer to this problem as the *matching and heatmap selection problem* (MHSP). The model is shown in Appendix B.2.1. The goal of MHSP is to match drivers and orders and to sequentially prescribe heatmaps that direct probabilistic driver movement to nodes where they are most needed, so as to improve total driver-order matching. In MHSP, both driver relocation and future orders are uncertain. The model aims to maximize the expected matching, considering mean values of driver relocation, and random future orders.

Capturing heatmap selection and probabilistic driver movement in MHSP results in a high number of integer and binary variables. We observe that MCDRP is a valid upper bound on the matching and heatmap selection problem. Thus, to improve computational efficiency, matching and heatmap selection decisions are decomposed into two separate optimization problems. An optimal solution of MCDRP determines the optimal matching of orders to drivers, as well as the optimal repositioning of drivers. This solution is then used to infer the optimal heatmap, such that the platform selects a heatmap that results in a matching at the next decision epoch $t+1$, that is as close as possible to that if drivers were directly managed. We refer to this problem as the *heatmap selection* problem (HSP).

Lemma 1 proves the relationship between MCDRP and MHSP.

Lemma 1. MCDRP is a relaxation of MHSP.

The proof is given in Appendix B.1.2.

Heatmap Selection Problem

A solution to MCDRP specifies the optimal number of unmatched drivers to move from i to j ($\overline{u}_{(t-1)ij}$) for a given decision epoch $t-1$. It also determines the number of unmatched drivers at each node i ($\overline{m}_i^u = m_{(t-1)i} - \sum_{j:(i,j)\in A} \overline{x}_{(t-1)ij}$), where $\overline{x}_{(t-1)ij}$ are the optimal values of the decision variables of the number of fulfilled orders. Define variable y_h as a binary variable that equals 1 if heatmap $h \in H$ is selected. The rest of the notation is similar to that introduced in Section 4.4.1. We use *bar* over a variable to indicate that the value of a variable is known, i.e., taken from the solution of MCDRP.

The goal of the heatmap selection problem is to get an expected matching at the subsequent decision epoch, t, as a result of drivers responding to the heatmap, that is as close as possible to one obtained from MCDRP. The problem of selecting an optimal heatmap, is formulated as follows. We refer to solving MCDRP then inferring the best heatmap using HSP as the *stochastic lookahead policy* (SLA).

$$[\text{HSP}] \quad \min \sum_{i\in N}\sum_{j\in N} |\overline{x}_{tij} - x_{tij}| \tag{4.5a}$$

$$\sum_{h\in H} y_h = 1 \tag{4.5b}$$

$$u_{(t-1)ij} = \sum_{h\in H} \overline{m}_i^u P_{ij}(h) y_h \qquad \forall (i,j)\in A \tag{4.5c}$$

$$\sum_{j\in N} x_{tij} \le \sum_{j\in N} \overline{x}_{(t-\tau_{ji})ji} + \sum_{j:(j,i)\in A} u_{(t-1)ji} \qquad \forall i \in N \tag{4.5d}$$

$$x_{tij} \le \sum_{k=t+\tau_{ij}}^{t+\omega} \overline{d}_{tkij} \qquad \forall i,j \in N \tag{4.5e}$$

$$x_{tij} \ge 0, y_h \in \{0,1\} \qquad \forall h \in H, i,j \in N$$

$$u_{(t-1)ij} \ge 0 \qquad \forall (i,j) \in A \tag{4.5f}$$

Objective function (4.5a) minimizes the absolute deviation between the number of matches in the solution of MCDRP and the number of matches as a result of a heatmap selection y_h at decision epoch t. Constraint (4.5b) ensures that a single heatmap is selected. Constraints (4.5c) compute the expected number of repositioning drivers as a result of selecting a particular heatmap. Constraints (4.5d) ensure that the total number of matches at a given node i does not exceed the expected number of drivers in that node, which is the sum of drivers arriving by matching plus those that reposition. Finally, Constraints (4.5e) make sure that the number of matches for a given o-d pair do not exceed the number of active orders.

A linearization of objective function (4.5a) is achieved by rewriting it as follows, with some additional constraints.

$$\begin{aligned}
\min \sum_{i \in N} \sum_{j \in N} \phi_{ij} & \\
\phi_{ij} \geq \overline{x}_{tij} - x_{tij} & \qquad \forall i, j \in N \\
\phi_{ij} \geq x_{tij} - \overline{x}_{tij} & \qquad \forall i, j \in N \\
\phi_{ij} \geq 0 & \qquad \forall i, j \in N
\end{aligned}$$

We note that in the heatmap selection problem, we only infer a single heatmap for a given decision epoch, $t-1$, and not for subsequent decision epochs within the forecast window, as in MHSP. When the objective function value of (4.5a) is zero, then there exists a heatmap with expected driver movement similar to the movement suggested by MCDRP, and thus the difference in matching at t is zero. However, since future decisions in the forecast window, Γ, are approximated in MCDRP assuming that driver movement is directly controlled, the optimal heatmap of HSP need not be optimal with respect to MHSP.

4.4.3 Simple Policy for Heatmap Selection

As the number of nodes in the network $|N|$ increases, the size of the heatmap selection problem (HSP) increases exponentially. That is because the response of drivers to a heatmap depends not only on the heat level of a particular node $i \in N$, but rather depends on the complete heatmap of all nodes in the network. In other words, the movement triggered by a heat level at a particular node is dependent on the heat levels of adjacent nodes. The number of possible heatmaps is $\gamma^{|N|}$, where γ is the number of heat levels, and thus the size of the heatmap selection problem grows exponentially with slight increases in the size of N.

To overcome this difficulty, we propose a simple easy-to-implement policy for selecting a heatmap, as an alternative to the SLA policy. Given the state of the system and a particular matching, this policy iteratively updates the heat level of regions with high driver supply shortage, computing the expected movement of drivers at each iteration. We assume that matching decisions are done independently; the policy focuses on selecting a heatmap in a way that considers, when assigning a heat level to a region, the implication on the movement of drivers throughout the network. Thus, the suggested policy selects a heatmap given a particular matching, and driver locations at $t+1$ as a result of this matching.

To maximize the reward obtained from matching alone, we propose a matching problem that modifies MCDRP such that no driver movement is assumed. That is, unmatched drivers stay in their regions of origin, and a driver at i moves to another region j only through being matched with an order of destination at j. We formulate this model in Appendix B.2.2 and show that it is a valid lower bound on MHSP, under one condition. We refer to this problem as the *matching with no driver relocation problem* (MNDRP) and use it as a benchmark to compare the effectiveness of heatmaps against. We use this model to maximize the reward earned from matching alone, then use the proposed simple policy to select a heatmap that further improves the accumulated reward.

Simple Policy for Heatmap Selection

For a decision epoch $t-1$, a solution of MNDRP results in an initial distribution of drivers at the subsequent decision epoch, $\bar{m}_{ti}$, based on the destinations of orders matched to drivers at $t-1$. For remaining drivers, i.e., unmatched drivers and inactive drivers, a heatmap is selected such that nodes with highest shortage receive additional supply first, by reflecting that on the heat level. We compute $\delta_{ti} = \bar{m}_{ti} - \sum_{j \in N} \sum_{k=t+\tau_{ij}}^{t+\omega} \bar{d}_{tkij}$, which is the shortage of drivers at node i and epoch t relative to orders. The policy is composed of two main steps: a modification step, and an update step. Those two steps are repeated iteratively until one of the following stopping criteria is reached: (i) there are no unmatched drivers available to move, (ii) no driver shortage at any node, (iii) the maximum heat level is reached and their is still shortage, or (iv) the maximum number of iterations is reached. The policy steps are as follows:

1. Solve MNDRP. Get an initial distribution of drivers as a result of matching, $\bar{m}_{ti}$.
2. Iteratively complete the following steps until one of the stopping criteria is reached:
   (a) Modification: find a node with the maximum driver shortage ($\min \delta_{ti}$). Increase its heat level by 1.
   (b) Update: calculate expected number of drivers in each region and expected shortage based on driver repositioning probabilities.

4.5 Computational Experiments

We test the proposed solution approach and benchmark policies on multiple test instances generated from the Chicago ridehailing dataset as we explain in the following section. We design the computational experiments to test the effect of four main factors on the benefit of heatmaps: (1) driver supply, (2) demand scale, (3) demand imbalance over the service network, and (4) repositioning probability estimate. Particularly, we vary initial driver supply to five different supply levels, computed as a percentage of average demand. We create three different demand arrival scenarios that vary the demand scale and network balance. We also create two different repositioning probability estimates to test the sensitivity of the proposed algorithm with respect to the approximated probability. For each of those settings, we solve ten instances of two cases of the problem; case 1: where the supply of drivers is constant throughout the planning horizon, and case 2: where drivers may enter or exit the service network throughout the day by transitioning to/from the artificial node. This creates a total of 400 test instances, each solved using 4 different solution approaches:

- *Matching with Controlled Driver Repositioning Problem* (MCDRP), where the movement of drivers is directly managed by the platform,
- *Stochastic Lookahead Policy* (SLA), the proposed solution approach that utilizes that MC-DRP solution to infer the best heatmap using HSP,
- *Matching with Controlled Driver Repositioning Problem* (MNDRP), where the network is balanced through matching,
- *Simple Policy* (simple) where we first optimize matching decisions using MNDRP, then iteratively assign heat levels to nodes based on their supply shortage as detailed in Section 4.4.3.

The computational time of the SLA policy is a few seconds per decision epoch. All other policies require less computational time.

4.5.1 Chicago Dataset Instances

We use the Chicago ridehailing dataset (TNP-Chicago-Trips, 2019) to generate the network and demand information of test instances. We use trip data from September 1, 2019 to February 29, 2020 during evening hours between 4pm and 7pm. We then aggregate the community areas of Chicago based on the city of Chicago data portal (Pump, 2012), which results in nine sides: far north, northwest, north, west, central, south, southwest, far southwest, far southeast. The trip pickup and delivery time in the dataset is reported in increments of 15 minutes, and thus, we use 15 minutes as the length of decision epochs.

Aggregating community areas into sides helps obtain representative estimates of the distribution of demand, since aggregating data results in more data points for each pair of nodes in the network. Drivers typically transition to neighboring nodes when considering repositioning, and thus partitioning the service region into granular nodes may reduce the capability of drivers understanding the heatmap collectively, and thus make it difficult to approximate driver response to a selected heatmap. The solution approach relies on the enumeration of the set of possible heatmaps, which grows exponentially with the size of the network. We leave the investigation of dominance relationship within the set of heatmaps and the extension to larger networks to future research. The current testing captures the essence of the trade-offs we test for in assessing the potential of heatmaps. Depending on the specific operations of platforms, the movement of drivers may be decomposed spatially, motivating the focus on the interaction within smaller networks.

For each date in the filtered dataset, we count the number of trips between each origin-destination pairs based on the aggregate nodes for a 15-min interval. There are twelve 15-min intervals between 4 pm and 7 pm, thus each day results in 12 data points of trip counts, for each o-d pair, corresponding to each of the 15-min intervals. Because of the very high number of rides in the Chicago ride hailing dataset (e.g., for some o-d pairs, there are over 300 rides in just a single 15-min interval), we scale down the count data by a factor of 10. We then fit the trip count data, for each o-d pair, into a negative binomial distribution. The negative binomial distribution is a generalization of the Poisson distribution where the equi-dispersion assumption does not necessarily hold. Travel time between nodes is computed as the average travel time between community areas of each pair of nodes.

For all test instances, we assume that the heatmap consists of 3 heat levels, $\gamma = 3$. Though the proposed methodology can handle a higher number of levels, heat levels should be chosen in a way that enables drivers to understand their implication, and thus a high number of heat levels is likely to cause confusion to drivers and create more randomness in their response. We set $T = 100$ decision epochs, and a forecast window, Γ, of 10 decision epochs. We also use a sample set of 10 scenarios ($|\mathbb{S}| = 10$) for the two-stage stochastic models. Drivers are randomly located throughout the service region at the first decision epoch. In case 2, where drivers may enter/exit the system, at the first epoch we set the number of inactive drivers in the auxiliary node to be 25% of total expected demand.

Heatmap Transition Probabilities

The probabilities of repositioning between nodes, as a response to a heatmap are first set for the special case (case 1), where active drivers do not exit the system and no new drivers join. This is then extended to the more general case (case 2). We note that we do not estimate those probabilities from the Chicago dataset as the set only contains trip information, and does not provide any data on the movement of *unmatched* drivers in the service network. We briefly discuss initial steps for estimating those probabilities in Section 4.3.1. For the computational testing, we randomly generate those probabilities following a systematic process. First, we define a pairwise repositioning probability for each pair of nodes (i, j) with heat levels ϵ_i, ϵ_j as $\rho_{\epsilon_i \epsilon_j}$. This denotes the probability of transitioning to node j with heat level ϵ_j from node i with heat level ϵ_i. Thus, the complement $1 - \rho_{\epsilon_i \epsilon_j}$ denotes the probability of staying at node i.

We define the repositioning probability between nodes $(i, j) \in A$ as shown in Equations (4.6). The number of drivers that reposition from any node $i \in N$ is a multinomial random variable, with m^u_{ti} trials and probabilities of success p_{ij}. We assume that drivers stay at their node of origin with probability equal to the compliment of the maximum pairwise transition probability, and thus we take the maximum with respect to adjacent pairwise probabilities. Then the probabilities of moving is normalized such that the sum of all probabilities adds to 1.

$$
\begin{aligned}
p_{ij} &= \left(\frac{\rho_{\epsilon_i \epsilon_j}}{\sum_{\{j:(i,j)\in A\}} \rho_{\epsilon_i \epsilon_j}}\right)(1 - p_{ii}) \\
p_{ii} &= 1 - \max_{\{j:(i,j)\in A\}} (\rho_{\epsilon_i \epsilon_j})
\end{aligned}
\tag{4.6}
$$

The extension of the repositioning probabilities to case 2, as well as the exact values used in the testing are specified in Appendix B.3.

4.5.2 Detailed Analysis of a Decision Epoch

We present detailed results of one decision epoch to demonstrate the use of a heatmap and the movement of drivers. Figure 4.2 gives a visual summary of the heatmap solution on the aggregate Chicago network, for one decision epoch of an instance with initial supply level equal to 120% of mean demand, solved using the SLA policy. For ease of exposition, we focus on the heatmap decisions, however, the complete solution specifies both heatmap and matching decisions. Recall that drivers may relocate to adjacent nodes, that can be reached within one decision epoch (15-min). The figure shows the optimal heatmap in that decision epoch, as well as the number of unmatched drivers in each node. It also shows the outcome of the heatmap, i.e., the movement of drivers as a response to the heatmap (no. drivers). Note that the reported number of drivers in each node represents only the response of drivers to the heatmap, and does not account for the number of drivers as a result of matching decisions.

Unmatched drivers in each node, as well as active drivers in the auxiliary node that may enter the service region, respond to the heatmap by relocating probabilistically. For example, given this heatmap, the 14 unmatched drivers at the *central* region will stay in the same region with probability 70%, relocate to the *west* region with probability 10%, and exit the system with probability 20%. To transition between decision epochs, we generate 50 samples of the multinomial distributions of driver movement, given the number of unmatched/inactive drivers and the probabilities of repositioning for the selected heatmap. We then choose the movement with the highest frequency, so as to simulate the movement with the highest probability. If multiple realizations have the same highest frequency, we break the tie arbitrarily.

4.5.3 Assessing the Effectiveness of Heatmaps

To examine the effect of heatmaps in changing the locations of drivers in a service network, we compare total demand fulfillment relative to two benchmarks, MCDRP, where movement of drivers is directly managed, and MNDRP, where unmatched drivers stay in the same region

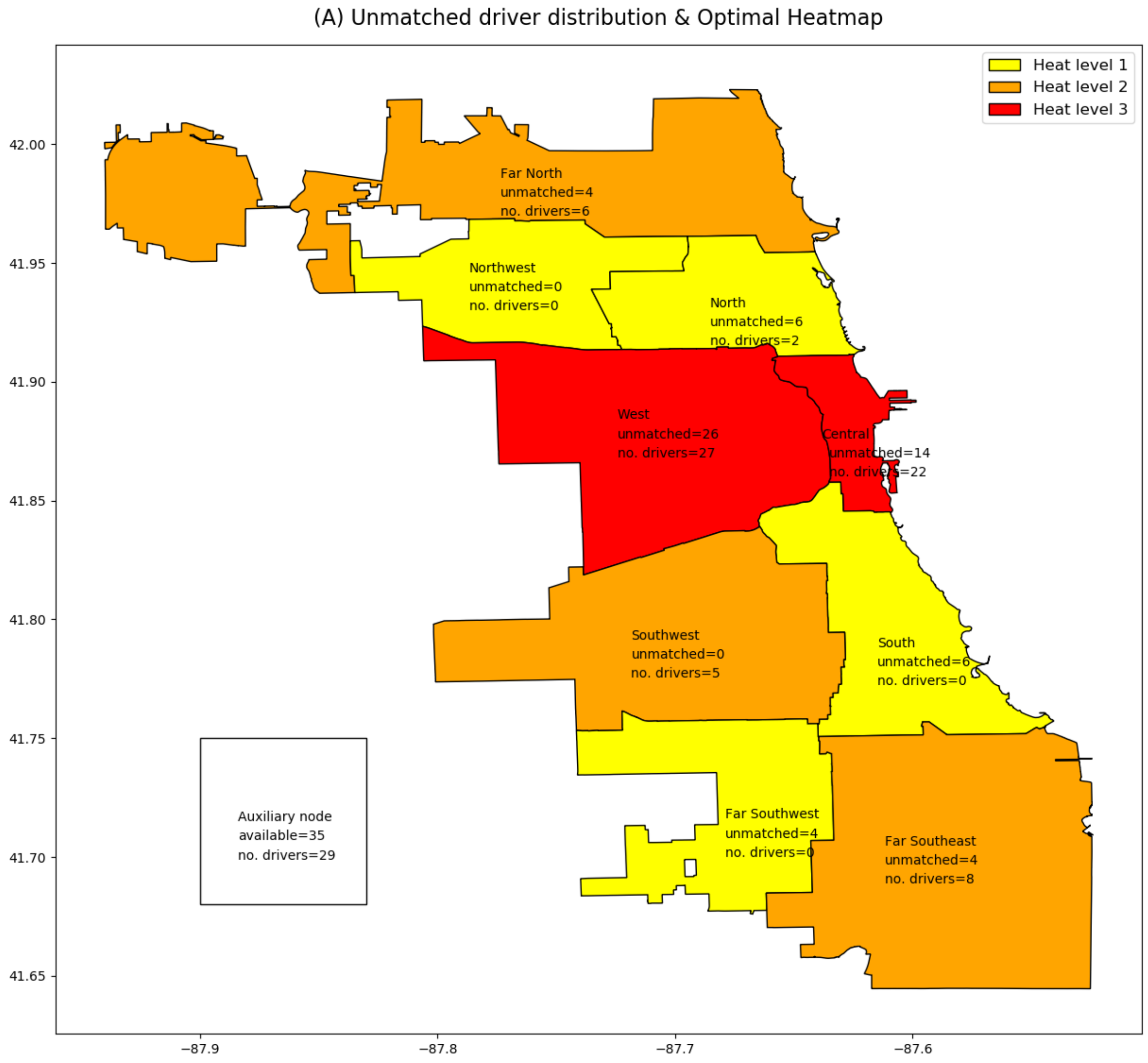


Figure 4.2. Heatmap solution for one decision epoch. The figure shows the optimal heatmap, the distribution of unmatched drivers, and the number of drivers in each node as a response to the heatmap.

and do not relocate to neighboring nodes. Recall that MCDRP is an upper bound on the SLA policy, and represents the maximum possible demand fulfillment, while MNDRP maximizes order fulfillment utilizing matching as the sole control lever. Figure 4.3 plots the percentage of orders fulfillment for 10 test instances, three levels of initial supply, under each of the two cases (with or without the option of additional drivers entering/exiting the network). The three levels of supply are relative to total expected demand in the network. For instance, 80% supply is when the number of drivers in the network is 80% of the total arrival rate of orders between all pairs of nodes, which represents a scenario where supply is less than demand. Similarly, 100% and 150% supply represent scenarios where supply is equal to or exceeds expected demand, respectively.

We note here that when the number of drivers in the network is constant (case 1) and driver supply does not exceed demand (80% - 100% of expected demand), balancing the network through matching is sufficient to maximize order fulfillment. Driver repositioning in this case only slightly improves order fulfillment. This can be noted from the small gap in the percentage of demand fulfilled for the MNDRP relative to MCDRP. With a supply of 80%, average order fulfillment is 77.13% and 78.65% for MNDRP and MCDRP, respectively, while a supply of 100% results in 84.73% and 86.32% for MNDRP and MCDRP, respectively. This indicates that even if driver movement is managed directly (MCDRP), which is the best possible outcome, little improvement in service level can be achieved relative to balancing the system only through matching (MNDRP). However, when supply exceeds demand (150% of expected demand), MCDRP and heatmap policies (SLA and simple policy) achieve higher demand fulfillment, while MNDRP only improves marginally. Average order fulfillment of MNDRP is 93.03%, while that of the SLA policy is 99.02%, an improvement of 8.14% relative to the supply level of 100%. This shows that heatmaps are more effective when the supply of drivers in the network exceeds demand, so that they improve the network balance beyond the level achieved by matching alone.

When drivers enter and exit the service region throughout the planning horizon (case 2), we see a slightly higher gap between the SLA policy and the simple policy. For instance, with the supply level of 100%, the SLA policy achieves an average order matching of 98.28%, while that of the simple policy is 94.55%. This indicates that the policy by which we select the heatmap plays a role in its effectiveness. This is especially true for case 2, since drivers leave the network (probabilistically) as a result of the heatmap choice, and thus a suboptimal choice may result in more drivers exiting the system. For detailed statistics refer to Tables B.1 and B.2 in Appendix B.4.

Improvement in service level by driver supply

Figure 4.4 plots the average demand fulfillment for cases 1 and 2 under varying levels of supply. We consider five supply levels (80%, 90%, 100%, 120%, and 150% of expected demand) and plot the average order fulfillment over 10 problem instances. We notice in case 1 that the difference in demand fulfillment between the policies is more significant as supply passes the 100% mark.

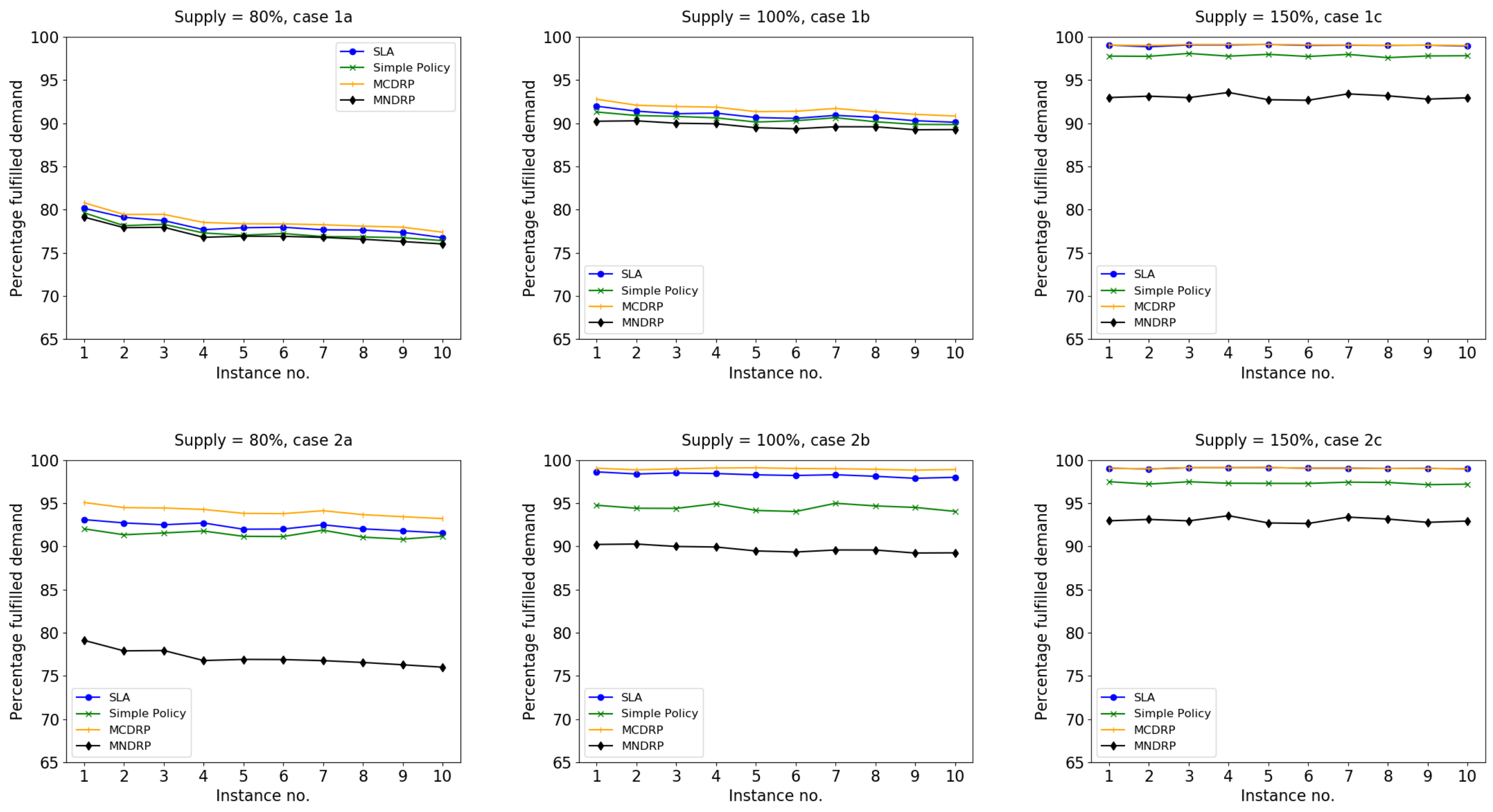


Figure 4.3. Percentage demand fulfillment varying policies and driver supply: cases 1 (no enter/exit) and 2 (enter/exit).

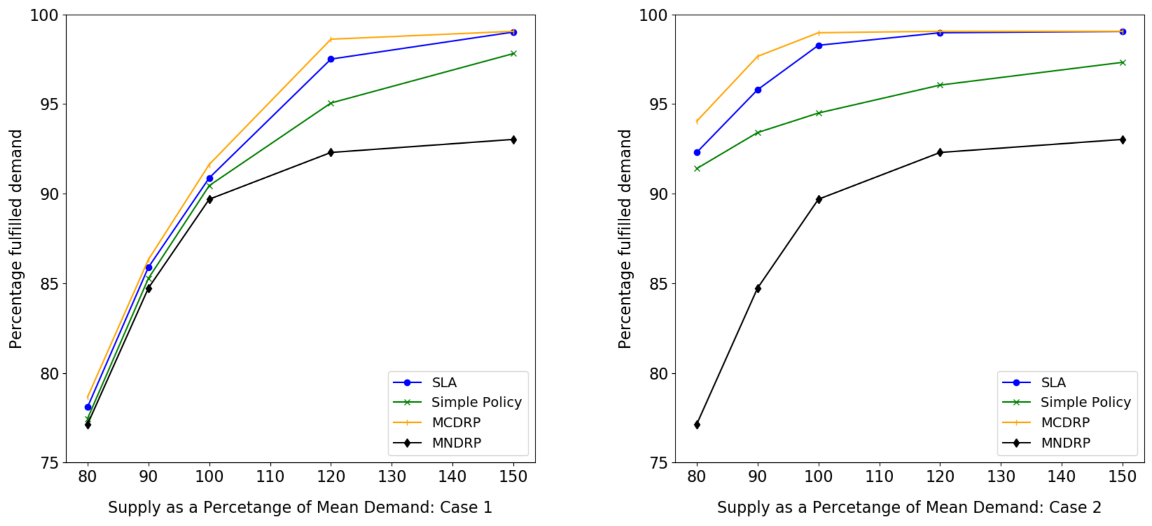


Figure 4.4. Improvement of demand fulfillment by driver supply: case 1 and case 2.

The SLA policy achieves 1.54% improvement in demand fulfillment relative to the MNDRP, for a supply level of 100%. However, when supply increases to 150%, this performance gap increases to 5.99%. The difference is more apparent in case 2: the SLA policy almost converges to the performance of the MCDRP at the supply level of 120% and 150%, with a gap of 1.11% and 0.04%, respectively. For 150% supply level, a larger gap exists between MCDRP and MNDRP (6.03%), as well as MCDRP and the simple policy (1.24%).

Service level fairness by region

Since the platform's goal is to maximize total demand fulfillment, this may come at the expense of low service level in regions where demand outflow is lower than inflow, especially for low driver supply. We plot the average demand fulfillment by region in Figures 4.5. For the supply level of 80%, we observe a gap of 29.5% and 16.6% between the best and worst service level by region under the SLA policy, for cases 1 and 2 respectively. The service level improves sharply with higher driver supply and the variation between different regions diminishes. At the supply level of 150%, the maximum service level gap between regions is 0.5% for both cases 1 and 2.

The opposite effect is observed in Figures 4.6 where we plot the percentage of drivers matched over regions. The percentage of matched drivers is high for all nodes, when the supply of drivers is low. For example, for case 2 and a supply level of 80%, the percentage of matched drivers by zone is between 89.7% and 99.8%. This indicates that the platform fulfills orders to destination nodes where outflow demand is expected. When supply increases to 150%, the percentage of matched drivers ranges between 28.6% and 90.1% for different zones, but the high order fulfillment (ranging between 98.9% and 99.4%) indicates that the majority of demand is fulfilled, even if there are no orders to match drivers with at destination nodes.

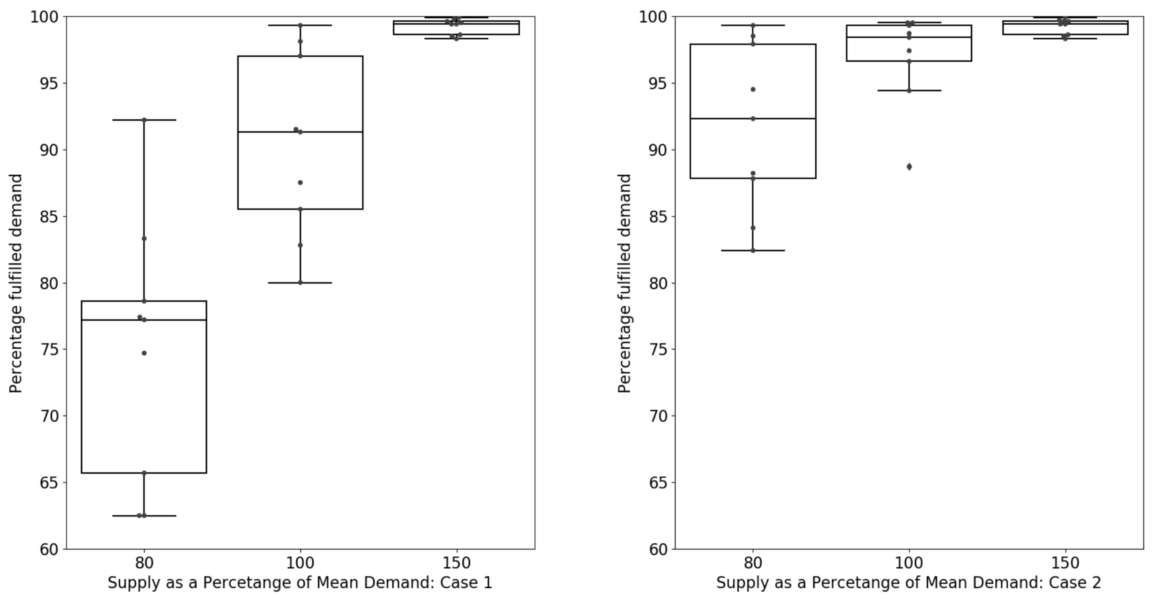


Figure 4.5. Average demand fulfillment by zone with varying driver supply - SLA policy.

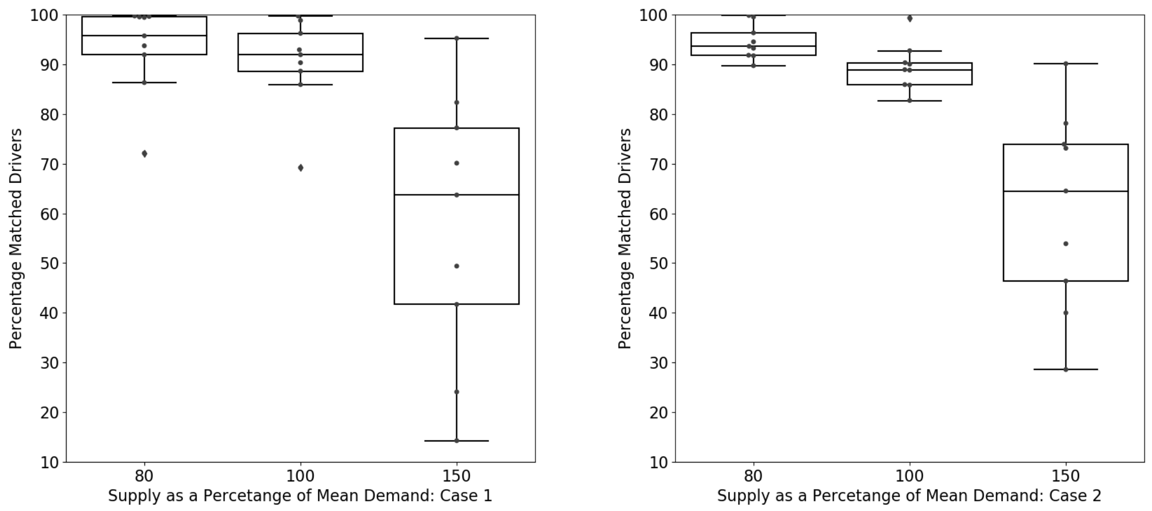


Figure 4.6. Average driver matching by zone with varying driver supply - SLA policy.

4.5.4 The Effect of Changes in Demand Patterns on the Benefit of Heatmaps

To test the effect of changes in demand on the benefit of heatmaps, two testing scenarios are generated. (1) Demand data, discussed in Section 4.5.1, is scaled down by a factor of 4. (2) An imbalanced network is created by reducing the demand inflow of two nodes in the network by a factor of 10, while keeping the rest of the demand unchanged. This represents a scenario where many orders originate from certain regions, but much fewer orders are delivered to that region. We notice that scaling down demand does not significantly affect the results. Thus, we focus on scenario (2) and report the results of scenario (1) in Appendix B.4.

Figure 4.7 plots the percentage of fulfilled demand, for imbalanced network testing scenario, with various levels of driver supply. We observe from Figure 4.7 that for case 1 with a supply level of 80%, heatmaps prescribed by the SLA policy improves total order matching by 8.63% relative to balancing through matching only (MNDRP). However, this gap quickly increases with additional driver supply; with a supply of 100% the gap between the two policies increases to 16.36%, while for a supply of 150%, this gap further widens to 34.34%. In an imbalanced network, using matching alone (MNDRP) as a control lever quickly fails; even with the increase in driver supply, the service level is leveled slightly below the 50% mark. With a supply of 80% an order fulfillment rate of 48.16% is achieved. When driver supply increases to 150%, order fulfillment only slightly increases to 49.1%.

We also note that the performance gap between the SLA policy and the simple policy is further highlighted when the network is imbalanced. The order fulfillment gap between those two policies increases from 5.46% to 14.7% for supply levels of 80% and 150%, respectively, under case 2. This indicates that the way by which heatmaps are prescribed can play a significant role in their effectiveness. The performance of the SLA policy also does not converge to that of the UB policy with higher supply levels, unlike the results of the original dataset shown in Figure 4.4. At a supply level of 150% in case 2, 98.07% of orders are fulfilled when driver repositioning decisions are controlled directly (MCDRP), while 84.58% of orders are fulfilled under the SLA policy.

Service level fairness by region

We examine the service level fairness by region when demand is imbalanced in the network and plot the demand fulfillment by region for the SLA policy in Figure 4.8. We notice that the two regions with demand outflow much higher than demand inflow (nodes 3 and 5) have the lowest service level. For case 1 and a supply level of 80%, those regions have an order fulfillment rate of 25.2% and 51.3%, while the next lowest order fulfillment rate by region is 89.7%. For case 2, a similar low order fulfillment rate is observed (35.4% and 62%). However, significant improvement

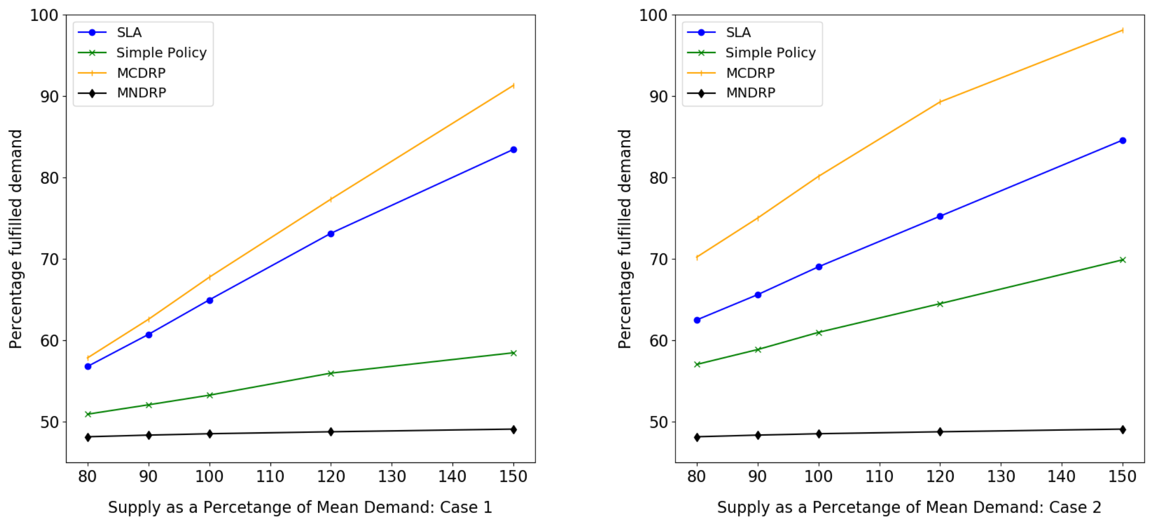


Figure 4.7. Improvement of demand fulfillment by driver supply - imbalanced network: case 1 and case 2.

in order fulfillment is achieved as supply increases. For those two regions, fulfilled orders increases to 67.2% and 90.7% for case 1, and 74.6% and 87.7% for case 2, given a supply level of 150%.

The opposite effect is observed when examining the percentage of matched drivers by region in Figures 4.9. Nodes 3 and 5 always have a matching of close to 100% regardless of driver supply. Other nodes, however, have driver matching percentages that vary by driver supply. For instance, for case 2 and a supply level of 80%, the percentage of matched drivers in other nodes varies between 66.6% and 92.8%. This variability increases further with additional supply; for a supply level of 150%, driver matching in other nodes varies between 29.4% and 91.0%.

4.5.5 Sensitivity to Transition Probabilities

We test the sensitivity of the performance of heatmaps when we vary the repositioning probability estimates. Particularly, we look at two scenarios. (1) For case 1, we reduce the probability of moving to any node by 50%. That is, before we compute the full transition probability matrix, we reduce the pairwise probability of movement by 50% and increase the probability of staying in the same node by that amount, so that the sum of probabilities adds up to 1. (2) For case 2, we reduce the probability of exiting the system by 50%, then increase the probability of staying in the same node by that amount. We test the algorithms on the imbalanced network discussed in Section 4.5.4.

Figure 4.10 plots the improvement in demand fulfillment by driver supply when we vary the transition probabilities as discussed. We see that for case 1, when the probability of moving is reduced by 50%, the gap between the performance of the SLA policy further deviates from that

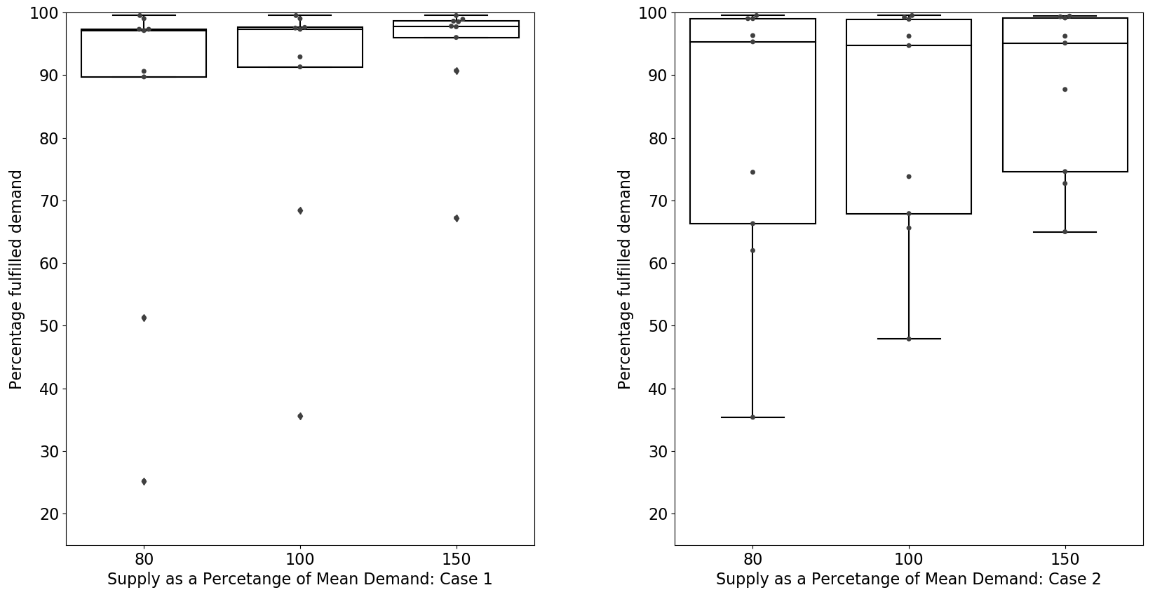


Figure 4.8. Average demand fulfillment by zone varying driver supply - imbalanced network, SLA policy.

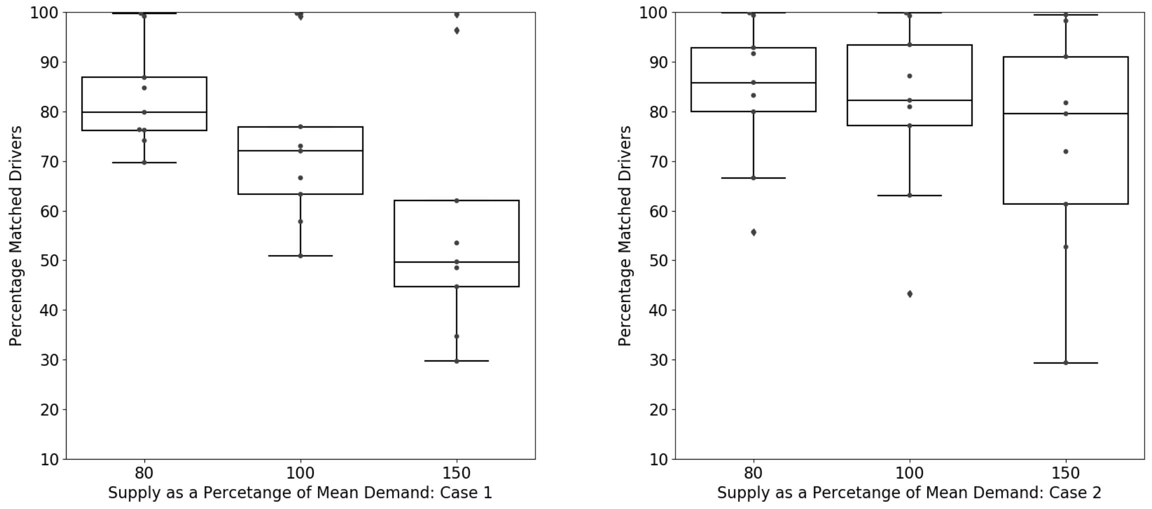
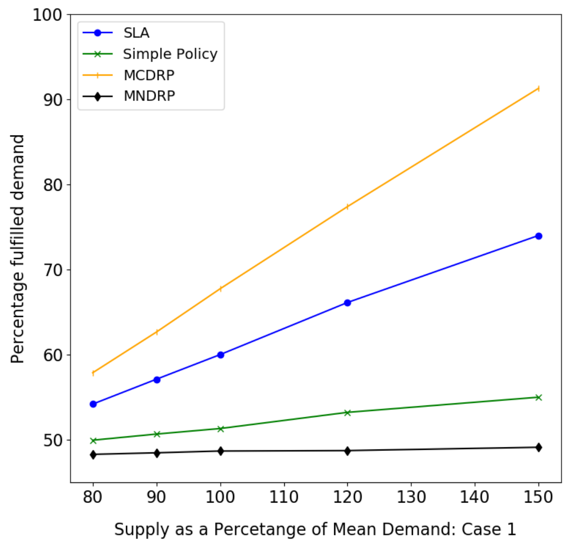


Figure 4.9. Average driver matching by zone varying driver supply - imbalanced network, SLA policy.



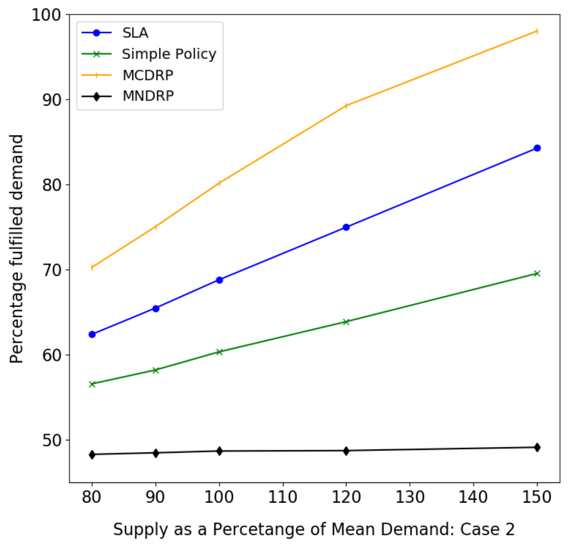


Figure 4.10. Improvement of demand fulfillment by driver supply - imbalanced network - P-sensitivity: case 1 and case 2.

of MCDRP. For instance, for a supply level of 120%, the SLA policy achieves 66.13% order fulfillment, while MCDRP achieves 77.39%. This contrasts with the results of the original dataset of the imbalanced network, where the SLA policy and MCDRP fulfill 73.14% and 77.33% of orders, respectively, for the same supply level. However, the SLA policy still significantly improves order fulfillment relative to MNDRP (48.74%), where the platform does not interfere with driver movement. For case 2, decreasing the exit probability seems to have little effect on the performance, and shows a consistent trend with the results of the imbalanced network in Figure 4.7.

4.5.6 Main Takeaways

The computational experiments highlight the benefit of utilizing heatmaps as a control mechanism to balance driver supply in a service region. The results show that heatmaps are most valuable when:

1. **The supply of drivers exceeds demand.** When the number of drivers is lower than demand and the level of network imbalance is low (i.e., demand inflow for all nodes is only slightly different than demand outflow), heatmaps may not provide additional order fulfillment relative to balancing the network through matching alone. This is observed in Figure 4.3. However, as supply increases, additional demand fulfillment is attainable with the use of heatmaps.

2. **The demand in the network is highly imbalanced.** In networks where the difference between demand inflow and outflow of some regions is high, heatmaps achieve considerably

higher demand fulfillment compared to balancing through matching alone. In particular, they achieve an improvement of up to 35.48% compared to the matching only benchmark (MNDRP), when supply of drivers is high (150% of expected demand). This is observed in Figure 4.7.

3. **Matching and relocation decisions are considered together.** The policy by which heatmaps are prescribed plays a significant role in their effectiveness. In particular, the SLA policy outperforms the simple policy, especially as both demand imbalance and the number of drivers increase. This highlights the importance of considering matching and relocation decisions concurrently, rather than sequentially. It also shows the benefit of the heatmap selection problem, over using a simple heuristic for selecting a heatmap.

We also observe that for highly imbalanced networks, the performance of the heatmap does not converge to that of the UB problem. This shows that further incentives are needed to reduce the stochasticity in driver repositioning as a response to heatmaps so as to approach the MCDRP solution, where driver repositioning decisions are directly managed.

4.6 Conclusion

This chapter studies the use of heatmaps as a tool to balance driver supply and demand for delivery in a crowdsourced delivery system, where orders originate from any region within a service network and delivery time windows are very short. A heatmap communicates to crowdsourced drivers, through the mobile app, shortage within the service region, which triggers probabilistic movement of unmatched and inactive drivers. This research assesses the effectiveness of heatmaps in achieving a high service level by directing drivers to regions with supply shortage, compared to a benchmark model where driver movement is directly managed. Given the dynamic and uncertain nature of the problem, an MDP model is proposed for making allocation and heatmap selection decisions. A reduction of the solution space is derived, under the assumption that orders are homogeneous, i.e., have the same order fulfillment priority.

The model is then solved using a stochastic look-ahead policy, an approximate dynamic programming approach that iteratively makes decisions given the current status of the system and probabilistic information of the future. We propose an upper-bound problem that assumes drivers can be directly managed. We observe that the SLA policy can be more efficiently solved by disaggregating the matching decision, obtained by solving the upper-bound problem, then inferring the optimal heatmap from the solution of the upper-bound problem. We also present a simple policy as an alternative to the SLA policy, to efficiently solve large scale problems. This policy sequentially assigns heat levels to regions with the highest supply shortage. We assess the proposed policies against the upper-bound problem, and a lower-bound benchmark that assumes the platform can only influence driver movement through matching; drivers do not relocate to other regions if not matched.

We conduct computational experiments derived from the Chicago ridehailing dataset to assess the performance of the proposed policies and analyze when heatmaps can help the platform achieve a high service level. Computational results show that the number of drivers in the system has a great impact on the effectiveness of heatmaps in balancing supply and demand. That is, with a higher number of drivers, a larger gap exists between the service level obtained when using heatmaps as compared to the LB policy, where the platform does not interfere with driver movement. We also find that imbalanced networks, where the demand inflow of some nodes significantly differs from the outflow, especially benefit from the use of heatmaps. Accounting for uncertainty in future demand and the interdependence between heat levels of different regions can also significantly improve supply-demand balance in the network. This is observed from the performance gap between the SLA policy and the simple policy.

The performance of the SLA policy converges to that of the UB policy when the supply of drivers increases, for the original dataset with a moderately balanced network. However, for highly imbalanced networks, though the use of heatmaps improves service level significantly relative to the LB policy, the performance does not converge to that of the UB policy. This indicates that additional incentives are needed to reduce the stochasticity in driver response to heatmaps so as to further approach the level achieved when drivers are directly managed.

An interesting future direction that complements our current study would be designing a data-driven estimation algorithm, from historical data, to compute the probability of driver movement as a response to a heatmap. Further work could propose enhancements to the simple policy to improve its performance and efficiently select close-to-optimal heatmaps, without needing to assess the whole set of heatmaps.

Another possible extension is to study various modifications to the proposed problem setting, including heterogeneous drivers, explicit matching between orders and drivers, routing of orders under short time windows, among others. This builds up on the original MDP model presented, but with the need for a different solution methodologies that account for more detailed information of drivers and orders.

Investigating and modeling uncertainty in the impact of heatmaps by assuming that repositioning probabilities are uncertain is another interesting extension of this work. Furthermore, investigating dominance relationships in the set of heatmaps so as to handle larger service networks would further strengthen the proposed model and solution approach. Lastly, from a practical perspective, it is interesting to find ways for drivers to disclose their repositioning decisions to the platform, so that the movement and shortage of drivers is updated in real time. This helps in managing the competition between drivers, which may negatively affect the impact of heatmaps.

Chapter 5

Dynamic Matching with Driver Welfare Considerations in Crowdsourced Delivery

5.1 Introduction

The rise of the sharing economy has enabled the creation of many on-demand service platforms, that connect customers looking for quick service with freelance workers who are ready to provide them. In most platforms, workers are considered independent contractors, and thus are not entitled to employee regulatory protection (e.g., minimum wage, sick leave). The lack of such regulations has attracted criticism from workers, labor advocates and regulators, who call for more equitable welfare measures. This has been increasingly noticeable in recent years, with the rapid growth of such platforms, which has led to the increase in the number of workers associated with sharing economy platforms.

In December 2018, New York city passed the first minimum wage pay rate for ride-hailing drivers in the United States, which forces companies to pay drivers a minimum of $17.22 per hour, after deducting trip related expenses (O'Brien, 2019). Later, in September 2019, California passed bill AB-5, what since became known as the "gig economy law". This law, which came into effect on January 1st 2020, aims to protect the welfare of gig economy workers and other independent contractors by enforcing that they be treated as employees, who receive at least the minimum wage and are entitled to sick and vacation leave. Most recently, Uber drivers in the UK are now entitled for minimum wage, holiday pay and pension (Uber-UK, 2021). The welfare of gig economy workers has been a well-noted drawback of the sharing economy. Yet, recognizing those freelance workers as employees has the unintended negative effect of many of those individuals losing work (Staggs, 2020), especially in smaller markets.

One of the main features that promoted the success of sharing economy platforms is the flexibility it offers its workers by allowing them to schedule their own working hours. Such flexibility has attracted workers that do not depend on the gig economy for full-time employment, but rather use it as a means of supplementing their income. Enforcing that platforms consider all workers as employees will eliminate the inherent flexibility of the sharing economy, and will drag the business models of those platforms back to traditional ones. In this research, we explore one possible middle ground solution that continues to offers flexibility to gig economy workers, but provides compensation guarantees for active workers. The goal is to design a system with some compensation guarantees that ensure that gig economy workers meet a target utilization rate, or a minimum wage, without the rigid requirement of considering them employees.

In particular, we consider three types of guarantee policies: (i) a maximum utilization guarantee that aims to achieve a utilization target for an activity window of a specified number of consecutive hours, (ii) an hourly minimum wage guarantee for an activity window, and (iii) an hourly minimum wage only while drivers are actively delivering orders, but not between consecutive order matches. Since drivers are typically paid when they are actively fulfilling a task, the fraction of time they expect to be busy (i.e., their utilization) plays a significant role in the earnings they receive (Benjaafar and Hu, 2020).

We develop a dynamic matching framework that continues to treat workers as independent contractors, but establishes guarantees to improve the earning potential of workers, ultimately contributing to enhancing their welfare. This enables the continuation of service and maintaining an abundant supply of workers, since the proposed system does not affect the core business model of gig economy companies, and the flexibility of workers is not compromised. Our purpose in this research is to answer the following fundamental questions: *What are the implications of the different types of earning guarantees on the operations of the platform? Are drivers better off under such guarantees? Is the platform worse off?*

We focus our modeling and analysis on sharing economy platforms providing transportation services, in particular crowdsourced last-mile delivery. We propose a Markov decision process (MDP) model to capture the highly dynamic and stochastic nature of such platforms. The state of the system constitutes detailed driver and order information. When a driver logs in the app, the driver is available for matching, but inactive. Once a driver receives their first order, their status changes to *active* and the platform guarantees a utilization level or minimum wage for a specified *activity time window* of τ consecutive hours. The goal of the platform is to maximize profit earned from fulfilling orders, while providing utilization guarantee for active drivers. Since the setting is dynamic, a matching at time t, not only incurs the immediate costs/revenue, but also affects the feasibility and optimality of future decisions. Thus, our choice of the modeling framework (MDP) is motivated by the importance of capturing the downstream cost of decisions, when such guarantees are in place.

In this chapter we make several key contributions. (1) We introduce and model a timely new problem that studies a middle-ground solution in the battle of improving the welfare and regulatory protection of gig economy workers, by guaranteeing a particular utilization/wage, thus improving workers earning potential without changing the flexible matching system employed by such platforms. We present a Markov Decision Process (MDP) to model the problem. We also propose a base-case model that reflects the case of no utilization guarantees. (2) To handle large scale problems, and overcome the curse of dimensionality, we present a value function approximation solution approach, based on approximate dynamic programming, which enables the efficient computation of good quality solutions. (3) Finally, we conduct thorough computational experiments to test the performance of the proposed model and solution approach relative to the myopic policy, and the base-case model. We also compare the different types of guarantee policies and quantify the expected earning for drivers and the expected profit for the platform under each policy, and relative to the base policy.

The remainder of this chapter is organized as follows. Section 5.2 reviews relevant literature. Section 5.3 describes the problem and introduces the proposed MDP model. Section 5.4 explains the different guarantee policies, and the explicit formulation of their cost functions. Section 5.5 details the proposed value function approximation algorithm and derives some theoretical properties of the proposed solution approach. Section 5.6 presents computational experiments and main results. Finally, the chapter is concluded in Section 5.7.

5.2 Literature Review

There is a growing body of literature addressing decision problems arising in the context of on-demand service platforms that were enabled by the rise of the sharing economy. Benjaafar and Hu (2020) review the main types of sharing economy platforms, discuss major streams of research, and open research questions. The authors note that on-demand service platforms have come under scrutiny for the lack of regulatory protections of workers. Industry groups have argued that these platforms compete unfairly against established businesses; since workers are independent freelancers, the platform does not abide by labor regulations.

Initial literature has examined welfare considerations in sharing economy platforms. Yu et al. (2017) show that moderate regulatory policies improve total social welfare, by striking a balance between key competing objectives, such as job creation, reducing traffic congestion, maximizing consumer and worker welfare. Benjaafar et al. (2019) examine the question of whether labor pool expansion reduces workers welfare. They find that growth in labor pool size initially stimulates demand, improving the utilization of workers, and increasing their wages. Welfare of workers decreases only when the labor pool is sufficiently high. Hu and Zhou (2019) study the effect of minimum wage on the platform, drivers and riders. The authors derive worst-case performance of fixed commission contracts to quantify a guaranteed lower bound on performance.

Existing research addressing wage and welfare questions in sharing economy platforms has mainly focused on strategic questions concerning the effect of such minimum wages on the platform, drivers, and consumers, or lack thereof. In this research, we examine a different angle of the problem: the impact of putting such guarantees into action from the platform's perspective. We model the problem of dynamically matching drivers with orders when such a minimum wage or utilization guarantee is in effect, and design a methodology for providing real-time matching that maximizes the platform's profit while respecting driver wage guarantees.

Most literature investigating problems arising in the sharing economy examine questions on how best to price services and compensate workers, especially when workers are self-scheduling and prices and wages are dynamic. Cachon et al. (2017) examine various compensation schemes in a service platform with self-scheduling capacity. The authors conclude that there is notable benefit to the platform in adopting a pricing policy that dynamically adjusts prices and wages. They show that all stakeholders can benefit from surge pricing, in periods of high demand, as providers are better utilized, and consumers benefit from expanded access to the service. Bimpikis et al. (2019) investigate the impact of spatial pricing in ride hailing networks and derive conditions under which spatial price discrimination maximizes a platform's revenue. Besbes et al. (2021) examine the short term spatial pricing problem faced by ridehailing platforms, in particular how to optimally set prices within a continuous service region so as to maximize the platform's revenue.

Sharing economy platforms for last mile delivery, i.e., crowdsourced delivery, partially or fully utilize freelance workers to complete last mile delivery tasks. Alnaggar et al. (2019) review the operations research literature on crowdsourced delivery along with the main trends of this system in practice. The authors classify decisions faced in this system into four main categories: *matching, routing, driver scheduling,* and *driver compensation.* This work loosely falls under the *driver compensation* category. Compensation of drivers in crowdsourced delivery was only considered a decision variable by Qi et al. (2018) in their wage response model. Many papers in the crowdsourced delivery literature assume that driver compensation is a function of the location of customers (e.g., Archetti et al., 2016 and Gdowska et al., 2018), deviation of the driver's pre-planned trip, when drivers are not dedicated workers (e.g., Archetti et al., 2016, Macrina et al., 2017 and Arslan et al., 2018), or parcel size (Gdowska et al., 2018). In contrast to the literature, in this research we aim to achieve a compensation guarantee over a period of time, rather than decide on a particular compensation structure for workers. The proposed model and solution approach are general, and can handle different types of compensation schemes.

5.3 Markov Decision Process (MDP) Model

We consider a last-mile delivery network operated by a crowdsourced delivery platform, where orders and drivers arrive to the system randomly throughout the planning horizon (e.g., a given day). Order arrival follows a Poisson process with a known arrival rate λ per epoch. When an order arrives, its origin and destination information are known, as well as the time window of order fulfillment. New drivers arrive to the platform with a known arrival rate μ per decision epoch.

The pool of drivers is divided into two sets: (a) active drivers: drivers who were matched with a delivery request and who may be en-route delivering an order and (b) inactive drivers: drivers who are available to provide service, but have not yet received any order requests. Inactive drivers, set (b), receive no matching guarantees. However, once a driver receives their first delivery request, they join the pool of active drivers (a), and the platform offers them a compensation guarantee W, for τ consecutive hours. At discrete time points throughout the day, i.e., *decision epochs*, the platform decides on the matching between orders and drivers that maximizes the number of matched requests while offering active drivers guarantee W, expressed in terms of utilization or wage, for τ consecutive hours, which is the duration of the activity time window during which they receive the guarantee.

Note that we define all notation used in the model formulation and solution approach within text, and also provide a comprehensive notation summary in Appendix C.1.

5.3.1 State Space

The state variable S_t, at decision epoch t, is the minimal amount of information necessary and sufficient to make a decision. The two main components that constitute the state variable in this problem are driver information and order information. Driver information includes the set of drivers in the system, each with known attributes detailing their activity status, availability for matching, location, progress towards guarantee W (i.e., utilization or wage earned), active time spent in the system. Order information describes the the set of active orders, their origin-destination (o-d) locations and time window (announcement time and delivery deadline).

At decision epoch t, each driver has an attribute vector a, such that:

$$a = (s_a, m_a, o_a, h_a, y_a)$$

where s_a is a binary indicator that equals 1 if the driver is active (i.e., has received an order request and is within their guaranteed time interval τ), and 0 otherwise. m_a is a binary indicator that is set to 1 if the driver is available for matching, and 0 if the driver has been matched at an earlier epoch and is currently delivering an order, and thus not available for matching. o_a denotes the current location of the driver, h_a is the active time spent up to time t. For active drivers ($s_a = 1$), active time h_a is less than or equal to the activity time window τ. After being active for τ consecutive hours, the driver status changes to inactive, thus the driver joins the pool of inactive drivers and no longer receives the compensation guarantee. y_a denotes the progress towards the desired guarantee policy, e.g., the wage earned by the driver up to time t or the portion of active time that the driver is utilized up to time t. We denote the set of driver attributes at time t, for drivers who are available for matching ($m_a = 1$), and drivers who are en-route ($m_a = 0$), as $\mathcal{A}_t$ and $\mathcal{A}'_t$, respectively.

Each order request is associated with an attribute vector b that retains necessary information about the order. That is,

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

where o_b and d_b are the origin and destination of the order, and $[t_b^{min}, t_b^{max}]$ is the time window of order fulfillment. $\mathcal{B}_t$ is the set of all demand attributes b at time t.

We denote the number of drivers with attribute a at time t as R_{ta}, and the vector of available drivers at t as $R_t = (R_{ta})_{a \in \mathcal{A}_t}$. Similarly, the number of orders with attribute b at t is denoted as D_{tb}, and the vector of orders at t is $D_t = (D_{tb})_{b \in \mathcal{B}_t}$. Thus the state of the system at epoch t is compactly written as $S_t = (R_t, D_t)$. Note here that the driver and order vectors R_t and D_t are typically very sparse because of their detailed attributes.

5.3.2 Action Space

We define variables x_{tab} and x_{ta0} as the number of drivers with attribute a assigned to orders with attribute b in decision epoch t, and the number of drivers with attribute a that are not assigned to any order in t, respectively. Let $\mathcal{B}^+$ denote the set of all possible decisions, i.e. $\mathcal{B}^+ = \mathcal{B}_t \cup \{0\}$. The decision vector at time t is expressed as $\boldsymbol{x}_t = (x_{tab})_{a \in \mathcal{A}_t, b \in \mathcal{B}^+}$.

5.3.3 Myopic Objective Function

The objective of the platform is to maximize profit obtained from demand fulfillment, while providing drivers with the specified guarantee. That is, once a driver is *active*, they are offered a guarantee W, while they are within their activity time window τ. We detail the different types of guarantee policies in the next section. The reward of assigning a driver with attributes a to an order with attributes b is expressed in general form as follows:

$$\pi_{ab} = \begin{cases} R(b) - C(a,b), & \text{if } b \in \mathcal{B}_t, (h_a < \tau) \text{ or } (h_a \geq \tau \text{ and } y_a \geq W) \\ R(b) - C(a,b) - F(a), & \text{if } b \in \mathcal{B}_t, h_a \geq \tau, y_a < W \\ -F(a), & \text{if } b = 0,\ s_a = 1, h_a \geq \tau \text{ and } y_a < W \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

The first case specifies the reward of matching a driver with an order, which is the revenue minus the cost of the match, if after fulfilling the match, the driver will still be within their activity window τ, or the driver would complete τ and has met the target guarantee W. The revenue term $R(b)$ is a function of the attributes of order b, whereas the cost term $C(a, b)$ depends on both the driver attributes (a), and order attributes (b). The exact forms of the revenue and cost functions are discussed in Section 5.4. The second case is similar to the first, but with an additional penalty term $F(a)$, a function of driver attributes (a), when a driver does not meet the desired guarantee. The third case assigns the penalty term for unmatched drivers who do not meet their policy guarantee at the end of their activity time window τ. Finally, the last case assigns a cost of 0 for all other possible cases.

5.3.4 Myopic Policy

At decision epoch t, given a particular state S_t, the myopic policy finds the best matching given the current state, without any consideration of future uncertainty or the impact of decisions made now on the feasibility and optimality of future decisions. Modeling the myopic policy serves as an essential building block in modeling the approximate dynamic programming algorithm that we detail in Section 5.3.5. We also use the myopic policy as a benchmark against which to compare the performance of the proposed algorithm.

We first discuss the time window constraint of fulfilling orders. At decision epoch t, the time it takes a driver a to fulfill an order b is denoted as ι_{ab}.

$$\iota_{ab} = \eta\big(||o_a - o_b|| + ||o_b - d_b||\big) \tag{5.2}$$

where $||\cdot||$ is the norm that measures the travel distance between two nodes (e.g. L^2 norm), and η is a scalar that converts the distance measure to a time estimate. We also let ι_b denote the travel time between the origin and destination of demand b, i.e., $\iota_b = \eta(||o_b - d_b||)$. Let $\mathcal{D}_{ab}$ denote the set of time-feasible assignments between drivers and orders, i.e., $\mathcal{D}_{ab} = \{(a,b) \in \mathcal{A}_t \times \mathcal{B}_t^+ | \iota_{ab} \leq t_b^{max}\}$

We formulate the myopic policy as an integer programming model expressed as follows:

$$\max \sum_{a,b \in \mathcal{D}_{ab}} \pi_{ab} x_{tab} \tag{5.3}$$

$$\sum_{b:(a,b)\in\mathcal{D}_{ab}} x_{tab} = R_{ta}, \qquad \forall a \in \mathcal{A}_t \tag{5.4}$$

$$\sum_{a\in\mathcal{A}_t} x_{tab} \leq D_{tb}, \qquad \forall b \in \mathcal{B}_t \tag{5.5}$$

$$x_{tab} \in \mathbb{Z}^+, \qquad \forall (a,b) \in \mathcal{D}_{ab} \tag{5.6}$$

Objective function (5.3) maximizes the profit of fulfilling orders, as detailed in (5.1). Constraint set (5.4) is the flow conservation of drivers, ensuring that each driver is assigned to one feasible decision, either to an order or is unmatched in the current epoch. Constraint set (5.5) ensures that the number of drivers assigned to orders with attribute b does not exceed the number of orders with that attribute. Constraint set (5.6) limits the domain of the decision variables to positive integer values.

Note that the constraint matrix expressed by (5.4) and (5.5) is totally unimodular (Wolsey and Nemhauser, 1999), and thus the linear relaxation of the above integer program results in integral solutions at all extreme points. We can therefore relax the integrality requirement (5.6) and replace it with the non-negativity constraint,

$$x_{tab} \geq 0, \forall (a,b) \in \mathcal{D}_{ab} \tag{5.7}$$

5.3.5 MDP Contribution Function

The linear program (5.3-5.5, 5.7) finds the optimal matching between drivers and orders, given known information, but does not account for the impact of decisions made on the future dynamics of the system. Let C_t denote the myopic objective function (5.3), and $\mathcal{X}_t$ denote the feasible region of the LP, characterized by Constraints (5.4, 5.5 and 5.7). The objective of the proposed MDP model is

$$V_t(S_t) = \max_{x_t \in \mathcal{X}_t} \left(C_t(S_t, x_t) + \gamma \sum_{s \in \mathcal{S}} p(s|S_t, x_t) V_{t+1}(s) \right) \tag{5.8}$$

Equation (5.8) is the Bellman recursion function. The first term minimizes the immediate myopic cost of choosing decision vector $x_t \in \mathcal{X}_t$, when in state S_t. The second term denotes the cost-to-go function, which maximizes a (discounted) expectation of the value of future states. Computing the exact value of $V_t(S_t)$ for all possible states S_t is intractable for reasonable size problems. Thus, rather than solving Equation (5.8) exactly, we employ approximate dynamic programming techniques to propose tractable policies that approximate the cost-to-go function.

We define W_t as the exogenous information that is continuously arriving between epochs $t-1$ and t. $W_t = (\tilde{R}_{ta}, \tilde{D}_{tb})_{a \in \mathcal{A}_t, b \in \mathcal{B}_t}$, where $\tilde{R}_{ta}$ is the number of drivers joining or leaving the system between $t-1$ and t, and $\tilde{D}_{tb}$ is the number of new order arrival between epochs $t-1$ and t.

Due to its computational advantages, we rewrite Bellman's equation utilizing a post-decision state representation (S_t^x) when estimating the cost-to-go, rather than the predecision state S_t. A post-decision state determines the state of the system immediately after a vector of decisions is selected, but before the arrival of any new stochastic information (e.g., new order arrival). Rewriting Bellman's equation as a function of the post-decision state, as shown in Equations (5.9), substantially reduces the dimensionality of the outcome space. For every realization of the random vector W_t, we solve one deterministic optimization problem, then find the mean of the optimal values of all optimization problems. This is much less computationally demanding than the original representation of Bellman's equation (5.8), where we need to compute the maximum over the entire distribution of random outcomes W_t (Ruszczyński, 2010; Powell, 2011).

$$\begin{aligned} V_t(S_t) &= \max_{x_t \in \mathcal{X}_t} \left\{ C_t(S_t, \mathbf{x}_t) + \gamma V_t(S_t^x) \right\} \\ V_t(S_t^x) &= \mathbb{E}[V_{t+1}(S_{t+1}) | S_t^x] \end{aligned} \tag{5.9}$$

The system thus evolves as follows: $(S_0, \mathbf{x}_0, S_0^x, W_1, S_1, \mathbf{x}_1, S_1^x, W_2, S_2, \ldots, S_t, \mathbf{x}_t, S_t^x, \ldots, S_T)$.

5.3.6 Transition Function and Post-decision State

The post-decision state captures the changes in the state vector immediately after a decision $\mathbf{x}_b$ is made. Thus, $S_t^x = (R_t^x, D_t^x)$ captures changes to both the driver and order vectors. To derive the post-decision state, we let $a' = a^M(a, \mathbf{x}_t)$ denote the changes of driver attribute vector a after decision $\mathbf{x}_t$ is selected, where a^M is a transition function. At decision epoch t, consider all positive assignments between drivers and orders $(a, b) \in \mathcal{A}_t \times \mathcal{B}_t$ such that $x_{tab} > 0$. To describe the changes to the attribute vector as a result of matching, we first introduce an indicator function

$$\delta_{a'}(a, \mathbf{x}_t) = \begin{cases} 1, & \text{if } a^M(a, \mathbf{x}_t) = a' \\ 0, & \text{otherwise.} \end{cases}$$

Then, the post-decision resource state $R_t^x = (R_{ta}^x)_{a \in \mathcal{A}_{t,x}}$ is updated as

$$R_{ta'}^x = \sum_{a \in \mathcal{A}_t} \sum_{b \in \mathcal{B}_t^+} \delta_{a'}(a, \mathbf{x}_t) x_{tab}, \qquad \forall a' \in \mathcal{A}_{t,x} \tag{5.10}$$

where $\mathcal{A}_{t,x}$ is the set of possible attributes of drivers after decision $\mathbf{x}_t$ is made.

To update the post-decision order vector, we note that any order with an expired time window, such that $t + 1 > t_b^{max} + \iota_b$ is considered lost, and is thus not moved forward to $t + 1$. We denote the set of lost orders as $\bar{\mathcal{B}}_t$. For the remaining orders, we update the order vector as follows:

$$D_{tb}^x = D_{tb} - \sum_{a \in \mathcal{A}_t} x_{tab}, \qquad \forall b \in \mathcal{B}_t \setminus \bar{\mathcal{B}}_t \tag{5.11}$$

Note that an active driver's attribute changes even if the driver is not matched. Particularly, the number of hours active in the system increases by the duration of the time interval of the decision epoch, i.e., $h'_a = h_a + \Delta$, where Δ is the time interval between two consecutive decision epochs.

5.3.7 Pre-decision State

The arrival of exogenous information between epoch t and $t + 1$, $W_{t+1} = (W_{t+1}^r, W_{t+1}^d)$, transitions the system from post-decision state S_t^x to the next pre-decision state S_{t+1}, where $S_{t+1} = (R_{t+1}, D_{t+1})$. We define a transition function $a^{M,W}$ that captures the progression of matched drivers' trips. That is, $a' = a^{M,W}(a), \forall a \in \mathcal{A}_t$, where $a' = (s'_a, m'_a, o'_a, h'_a, y'_a) \in \mathcal{A}_{t+1}$. For instance, if a driver arrives at the pickup location of the matched order o_b, the new origin of the driver is now $o'_a = o_b$. But since the order delivery is not complete, $m'_a = 0$, $a \notin \mathcal{A}_{t+1}$, indicating that the driver is not considered for matching at $t + 1$.

We again utilize an indicator function representation as follows:

$$\delta_{a'}(a) = \begin{cases} 1, & \text{if } a^{M,W}(a) = a' \\ 0, & \text{otherwise.} \end{cases}$$

Then the pre-decision resource vector $R_{t+1} = (R_{t+1,a})_{a \in \mathcal{A}_{t+1}}$ is computed as

$$R_{t+1,a'} = \sum_{a \in \mathcal{A}_{t,x}} \delta_{a'}(a) R^x_{ta} + W^r_{t+1,a'}, \qquad \forall a' \in \mathcal{A}_{t+1} \tag{5.12}$$

where $W_{t+1,a'}$ denotes the number of inactive drivers with attribute a', who arrive (if $W_{t+1,a'} > 0$) or depart (if $W_{t+1,a'} < 0$) between epochs t and $t+1$.

Next, we update the demand state vector $D_{t+1} = (D_{t+1,b})_{b \in \mathcal{B}_{t+1}}$. This is composed of carried forward orders D^x_{tb}, plus new orders that arrives between t and $t+1$, W^d_{t+1}, and is expressed as follows:

$$D_{t+1,b} = D^x_{tb} + W^d_{t+1,b}, \qquad \forall b \in \mathcal{B}_{t+1} \tag{5.13}$$

5.4 Guarantee Policies

The proposed guarantee policies differ mainly in the type of guarantee W, whether it is wage or utilization based, and the type and form of the penalty term $F(a)$. We consider three types of guarantee policies:

1. The platform maximizes the utilization of active drivers to reach a target utilization W for τ consecutive hours.

2. The platform guarantees active drivers a minimum pay W for τ consecutive hours. This is also referred to in the literature as the *effective wage*.

3. The platform guarantees drivers a minimum pay of W, while they are actively delivering orders. That is, drivers are compensated only for the time they are delivering orders and may have unpaid long breaks between consecutive orders that do not count towards their minimum pay. This is also referred to in the literature as the *nominal wage*.

We also model the base-case policy, where the platform does not offer any earning guarantees to drivers. This policy acts as a benchmark to compare the performance of the proposed policies against. In what follows, we formulate and explain the explicit cost function used under each policy.

Policy 1. Maximum Utilization During an Activity Window τ

The motivation behind this guarantee policy is to investigate a policy that is less dependent on the compensation structure employed by a particular platform, and thus gain generate insights that may extend to different types of platforms. In other words, since the results of the minimum pay guarantee depend on the assumption of the compensation of drivers, focusing on driver utilization while they are active enables us to examine the main trade-offs a platform encounters when driver utilization is prioritized.

Under this policy, y_a represents the utilization percentage of a driver with attributes a, and the penalty $F(a) = F(y_a)$ is a function of such utilization. Note that the penalty in this case is not actual dollars paid by the platform, but rather an artificial amount which forces the model to improve the utilization of drivers to reach the desired target W.

We formulate the cost function as follows.

- $R(b) = \theta_b w_b$, where w_b is the distance between the origin and destination of the order, and θ_b is the revenue per unit distance.
- $C(a, b) = \bar{\theta}_{ab} w_{ab}$, where w_{ab} is the distance between the origin of a driver and the origin of an order, and $\bar{\theta}_{ab}$ is a scalar, which represents the cost or disutility per unit distance.
- $F(a) = \zeta \rho_a$, where ρ_a is a piecewise linear function of the utilization of a driver, and ζ is a penalty score.

We note that the above cost and revenue functions are used in computational experiments examining different solution approaches under the utilization policy, which we present in Section 5.6.3. However, when comparing the different types of guarantee policies (Section 5.6.4), to be consistent, we use the cost and revenue functions proposed under policy 2.

Parametric Cost Function Approximation

Parametric cost function approximation refers to the use of parametrically modified deterministic models. The goal is to improve the performance of the myopic policy by parametrically adjusting the cost function and/or constraints so as to (partially or fully) capture the effect of uncertainty (Powell, 2019). To design the parametric cost function approximation (PCF), the objective function coefficients (5.1) are modified as follows.

- A *priority* score $L(a, b)$ is added to drivers with reduced utilization, and orders closer to expiry, where $L(a, b) = \alpha(\rho_a + \rho'_b)$.
- Case three is modified such that a penalty is incurred every time a driver falls below utilization W, and not just at the end of the activity window τ.

Thus, objective function π_{ab} is expressed as:

$$\pi_{ab} = \begin{cases} R(b) - C(a,b) + L(a,b), & \text{if } b \in \mathcal{B}_t, (h_a < \tau) \text{ or } (h_a \geq \tau \text{ and } y_a \geq W) \\ R(b) - C(a,b) + L(a,b) - F(a), & \text{if } b \in \mathcal{B}_t, h_a \geq \tau, y_a < W \\ -F(a), & \text{if } b \in \{0\},\ s_a = 1,\ y_a < W \\ 0, & \text{otherwise} \end{cases} \tag{5.14}$$

Although we use ρ_a in the expression of both $F(a)$ and $L(a,b)$, the penalty and priority scores are not equal because of ζ and α, and may be expressed using different functions.

Policy 2. Minimum Wage During an Activity Window τ

Under this policy, the platform guarantees a minimum compensation for drivers W for τ consecutive hours. If this minimum compensation is not earned from matching, the platform compensates drivers directly with the difference. The activity time window τ ensures that drivers get to work when they are ready, for a specified activity window, rather than leaving them unmatched too long between consecutive matchings. We compare this policy against policy 3, where drivers are only guaranteed a minimum wage while completing deliveries.

To estimate driver compensation and the platform's profit, we focus on the compensation structure used by restaurant meal delivery platforms. The revenue and cost functions are estimated using data from UberEats official website, as detailed in Section 5.6.4. Under this policy, the penalty $F(a) = F(y_a)$, is a function of the average wage earned during the τ activity time window. We modify the order attribute by adding a revenue per order, ϕ_b which is the fraction of order value that the platform earns from the restaurant for providing delivery service. We express the terms of the myopic objective function under the minimum wage policy as follows.

- $R(b) = \theta_b(w_b) + \phi_b$, where the first term is a delivery cost charged to the customer, that is based on the distance between the origin and destination of the order, and ϕ_b is the fraction of the value of the order that the platform collects from the restaurant/store.
- $C(a,b) = \bar{\theta}_{ab} w_{ab} + \varphi_b$, where $\bar{\theta}_{ab}$ is the variable per-unit-distance delivery cost, paid to drivers, and φ_b is the fixed cost per delivery, .
- $F(a) = (W - y_a) \cdot \tau$, where W is the minimum hourly wage, y_a is the hourly earning of a driver with attributes a, and τ is the number of hours in the guarantee time window.

Policy 3. Minimum Pay Guarantee while Delivering Orders

This policy is a modification of policy 2 that only guarantees a minimum pay to drivers while they are delivering orders. That is there is no activity time window τ, and the platform only ensures that the compensation received while delivering orders delivery is no less than a minimum pay W. Thus, drivers receive no earning guarantee if they are idle between two consecutive matchings. In this case, the myopic objective function changes to

$$\pi_{ab} = \begin{cases} R(b) - C(a,b), & \text{if } b \in \mathcal{B}_t,\ y_a \geq W \\ R(b) - C(a,b) - F(a), & \text{if } b \in \mathcal{B}_t, y_a < W \\ 0, & \text{otherwise} \end{cases} \tag{5.15}$$

where the revenue and cost functions are identical to those defined under policy 2. $F(a) = (W - y_a) \cdot h_a$, where h_a is the time spent delivering an order.

Base-Case Policy. No Driver Earning Guarantee

We model the base-case policy, where the platform does not offer any earning guarantees to drivers. The goal is to assess the benefit of compensation guarantees on drivers and the platform, and investigate the impact of driver earning guarantees on the quality of obtained assignments. Under this policy, the attribute vector of drivers reduces to the following:

$$a = (m_a, o_a)$$

where m_a and o_a denote the driver availability for matching and their location, respectively. Since the platform does not provide any minimum earning guarantee, it does not keep track of the *status* or *active time* of drivers. Intuitively, the demand attribute vector is unaffected by the platform's guarantee to drivers. Under the base-case policy, the immediate reward of assigning a driver a to an order b is expressed as follows:

$$\pi_{ab} = \begin{cases} R(b) - C(a,b), & \text{if } b \in \mathcal{B}_t \\ 0, & \text{if } b = 0 \end{cases} \tag{5.16}$$

That is, since no earning/matching guarantee is in place, the platform does not incur a cost/penalty for matches that compensate drivers poorly. The rest of the model follows from the MDP formulation detailed earlier.

5.5 Value Function Approximation

Bellman Equation (5.8) computes the value of being in a state S_t as the sum of the immediate myopic cost plus the discounted expected value of future states, i.e., the cost-to-go. Computing the value function (V_t) exactly for all possible states is computationally expensive, even with the reduction in outcome space enabled by the post-decision state representation (5.9), as discussed earlier. Thus, we aim to approximate the value functions $V_t(S_t^x)$ so as to obtain high quality solutions more efficiently, while still capturing the downstream impact of decisions. We refer to the approximation function as $\bar{V}_t(S_t^x)$. The optimization problem we solve is

$$Z_t(S_t) = \max_{x_t \in \mathcal{X}_t} \left\{ C_t(S_t, \mathbf{x}_t) + \gamma \bar{V}_t(S_t^x) \right\} \tag{5.17}$$

To compute approximation $\bar{V}_t(S_t^x)$, we simulate the system forward in time for N iterations. At iteration $n \in \{1, \ldots, N\}$, a sample path ω^n of the stochastic process is randomly selected, which includes realizations of the random variables (of orders and drivers) for the finite planning horizon, $t = 0, \ldots, T$. Given the sample path ω^n, the following optimization problem is solved for $t = 0, \ldots, T$, with the approximate value function computed from the previous iteration $n-1$, as follows:

$$Z_t(S_t^n) = \max_{x_t \in \mathcal{X}_t} \left\{ C_t(S_t^n, \mathbf{x}_t) + \gamma \bar{V}_t^{n-1}(S^{M,x}(S_t^n, \mathbf{x}_t)) \right\} \tag{5.18}$$

At a particular decision epoch t, we solve optimization problem (5.18) given state S_t^n. We obtain the optimal solution $\mathbf{x}_t^n$, then compute the post-decision state as $S_t^{x,n} = S^{M,x}(S_t^n, \mathbf{x}_t)$, and the subsequent pre-decision state $S_{t+1}^n = S^{M,W}(S_t^{x,n}, W_t(\omega^n))$. We then solve (5.18) again, and continue the process until we reach the end of the planning horizon T.

To design the approximate value function, similar to the idea presented in Simao et al. (2009), we take advantage of an important property of our problem. We notice that because of the detailed characteristics captured in the driver attribute vector, the value of R_{ta} typically does not exceed one. Thus, a linear approximation of $\bar{V}_t^n$ as a function of available drivers R_{ta} is proposed as follows:

$$\begin{aligned} \bar{V}_t^{n-1}(S_t^x) = \bar{V}_t^{n-1}(R_t^x) &= \sum_{a' \in \mathcal{A}_{t,x}} \bar{v}_{ta'} R_{ta'}^x \\ &= \sum_{a' \in \mathcal{A}_{t,x}} \bar{v}_{ta'} \sum_{a \in \mathcal{A}_t} \sum_{b \in \mathcal{B}^+} \delta_a'(a, \mathbf{x}_t) x_{tab} \\ &= \sum_{a \in \mathcal{A}_t} \sum_{b \in \mathcal{B}^+} \bar{v}^{n-1}_{t+\bar{\iota}(t,a,d), a^M(a, \mathbf{x}_t)} x_{tab} \end{aligned} \tag{5.19}$$

where $\bar{v}_{ta'}$ is the value coefficient of having a driver with attribute a' at epoch t, and $\bar{\iota}(t, a, d)$ is the travel time of driver with attribute a, when decision d is applied. Optimization problem

(5.18) is rewritten as

$$
\begin{aligned}
Z_t(S_t^n) &= \max_{x_t \in \mathcal{X}_t} \left\{ C_t(S_t^n, \mathbf{x}_t) + \gamma \sum_{a \in \mathcal{A}_t} \sum_{b \in \mathcal{B}^+} \bar{v}^{n-1}_{t, a^M(a, \mathbf{x}_t)} x_{tab} \right\} \\
&= \max \sum_{(a,b) \in \mathcal{D}_{ab}} (\pi_{ab} + \gamma \bar{v}^{n-1}_{t+\bar{\iota}(t,a,d), a^M(a,d)}) x_{tab} \\
& \sum_{b:(a,b) \in \mathcal{D}_{ab}} x_{tab} = R_{ta}, \qquad \forall a \in \mathcal{A}_t \\
& \sum_{a \in \mathcal{A}_t} x_{tab} \le D_{tb}, \qquad \forall b \in \mathcal{B}_t \\
& x_{tab} \ge 0
\end{aligned} \tag{5.20}
$$

Note that $\bar{v}^{n-1}_{ta'}$ can be interpreted as the marginal value of having an additional resource (driver) with attribute vector a', on the objective function value $Z_t(S_t^n)$. Therefore, we can also think of it as the slope of the objective function $Z_t(S_t^n)$ with respect to R^x_{ta}, which is computed as

$$
v^n_{t-1,a} = \frac{\partial Z_t(S_t)}{\partial R^x_{t-1,a}} = \sum_{a' \in \mathcal{A}_t} \frac{\partial Z_t(S_t)}{\partial R_{ta'}} \frac{\partial R_{ta'}}{\partial R^x_{t-1,a}}, \tag{5.21}
$$

where $\partial Z_t(S_t)/\partial R_{ta'}$ are the dual variables associated with Constraints (5.4) in LP (5.20), and $\partial R_{ta'}/\partial R^x_{t-1,a}$ is defined as

$$
\frac{\partial R_{ta'}}{\partial R^x_{t-1,a}} = \begin{cases} 1, & \text{if } a' = a^{M,W}(a^x_{t-1}) \\ 0, & \text{otherwise.} \end{cases}
$$

The estimated values $\bar{v}^n_{t-1,a}$ at iteration n are then updated to be a weighted average of the estimate from the previous iteration $\bar{v}^{n-1}_{t-1,a}$, and the new values $v^n_{t-1,a}$ as follows:

$$
\bar{v}^n_{t-1,a} = (1 - \alpha_{n-1}) \bar{v}^{n-1}_{t-1,a} + \alpha_{n-1} v^n_{t-1,a} \tag{5.22}
$$

Algorithm (1) details the steps of the forward simulation used for approximating the value functions.

5.5.1 Hierarchical Aggregation

The state space considered in this problem is vast, especially since the attribute vector of drivers stores highly detailed information about the driver status. When we consider the full attribute vector to approximate the value function, we face a statistical challenge; for attribute a and epoch t, we do not have enough samples to accurately learn its value function. We can create

Algorithm 1: Approximate Value Functions

```
Initialize $\bar{v}^0_{ta}, t \in \{0, \ldots, T\}, a \in \mathcal{A}_t,\ S^1_0$.
Set $n = 1$.
while $n \leq N$ do
    Randomly select a sample path $\omega^n$.
    for $t = 0, \ldots, T$ do
        Solve optimization problem (5.20). Let $x^n_t$ denote its solution, and $v^n_{ta}$
          denote the new estimate obtained from the dual of constraints (5.4).
        Update the value function as in Equation (5.22).
        Update the state:
            $S^{x,n}_t = S^{M,x}(S^n_t, x^n_t)$
            $S^n_{t+1} = S^{M,W}(S^{x,n}_t, W_{t+1}(\omega^n))$
    $n \leftarrow n + 1, S^n_0 \leftarrow S^{n-1}_T$
    return value functions $\{\bar{v}^n_{ta}, t = 1, \ldots, T, a \in \mathcal{A}_t\}$.
```

more samples for each attribute a at time t and iteration n, but since the state S is a vector, the number of possible states increases exponentially with the number of dimensions. Table 5.1 demonstrates the size of the outcome space for a single decision epoch t for a problem instance with 400 zones, and with y_a and h_a discretized to 20 values, where H is the number of possible outcomes.

To overcome this, hierarchical aggregation is implemented, as suggested by George et al. (2008). The authors argue that aggregating states for the purpose of approximating the value function results in better convergence of the VF, and thus better results. We train the model with and without hierarchical aggregation, and compare the results in the computational experiments.

Aggregation is used to create a hierarchy of state spaces, $\{\mathcal{A}^{(g)}, g = 0, 1, 2, \ldots, |\mathcal{G}|\}$, with successively fewer dimensions. One intuitive way to aggregate the state space is to use *spatial aggregation*. This is motivated by the fact that spatial correlation is likely to exist between neighboring zones, i.e., two zones that are close to each other are likely to have close value functions. Let Σ denote the number of zones in the model. We propose three levels of aggregation; the first, level 0, is the disaggregate level, where the number of zones is the same as in the model, i.e., $\sigma^{(g=0)} = \Sigma$. For each other aggregation level $g > 0$, we aggregate 2^{2g} neighboring zones into one area. Thus, for level one, we group each 4 zones together, i.e., $\sigma^{(g=1)} = \Sigma/4$ and at level two, we group every 16 zones together, $\sigma^{(g=2)} = \Sigma/16$. At each iteration, a disaggregate zone belongs to one and only one aggregate area, at each aggregation level. Note that the size of zones, whether at the disaggregate or aggregate level, need not be equal throughout the service region, as long as a disaggregate zone belongs to only one aggregate zone. For instance, a disaggregate zone in a sparsely populated region could be larger in size than one at a more densely populated region.

We aim to estimate the value functions at different levels of aggregation. The estimate of the function at the disaggregate level is assumed to be noisy but unbiased. Thus, we estimate the bias as the difference between the function at some level of aggregation and the disaggregate level 0. Choosing the right level of aggregation to approximate value functions includes a trade-off between statistical and structural errors. Let $\{\bar{v}_{ta}^{(g)}, g \in \mathcal{G}\}$ denote the estimate of the value function $\bar{v}_{ta}$ at different levels of aggregation. To reach a trade-off between statistical and structural errors, we compute an improved estimate as a weighted average of estimates at different aggregation levels as proposed by George et al. (2008). That is,

$$\bar{v}_{ta} = \sum_{g \in \mathcal{G}} w_{ta}^{(g)} \cdot \bar{v}_{ta}^{(g)}, \tag{5.23}$$

where $\{w_{ta}^{(g)}\}_{g \in \mathcal{G}}$ is a set of weights for each aggregation level, that add up to one. George et al. (2008) show that a good choice of weights can be obtained by using a simple formula that they call WIMSE, which assigns weights to each aggregate level such that they are inversely correlated to the estimate of the mean squared deviations from the true value at that level. The mean squared deviation is the sum of variance and bias estimates. We show how to compute those weights in what follows; interested readers are referred to George et al. (2008) for the complete derivation.

We first compute the following:

$$\begin{aligned} \bar{\beta}_{ta}^{(g,n)} &= \text{Estimate of bias due to smoothing a transient data series,} \\ &= (1-\eta_{n-1})\bar{\beta}_{ta}^{(g,n-1)} + \eta_{n-1}(v_{ta}^{n} - \bar{v}_{ta}^{(g,n-1)}). \\ \bar{\mu}_{ta}^{(g,n)} &= \text{Estimate of bias due to aggregation error,} \\ &= \bar{v}_{ta}^{(g,n)} - \bar{v}_{ta}^{(0,n)}. \\ \bar{\bar{\beta}}_{ta}^{(g,n)} &= \text{Estimate of total squared variation,} \\ &= (1-\eta_{n-1})\bar{\bar{\beta}}_{ta}^{(g,n-1)} + \eta_{n-1}(v_{ta}^{n} - \bar{v}_{ta}^{(g,n-1)})^2. \end{aligned} \tag{5.24}$$

Notice that in the proposed methodology we are using two step-size variables. η_n is used in updating bias and squared variation estimates, as shown in (5.24), and is typically a deterministic step size or a constant (e.g., $\eta_n = 0.1$). $\alpha_{ta}^{(g,n)}$ is the step size used to update the value function estimate $\bar{v}_{ta}^{n}$. The variance of the observations at a given aggregation level is estimated using

$$(s_{ta}^2)^{(g,n)} = \frac{\bar{\bar{\beta}}_{ta}^{(g,n)} - (\bar{\beta}_{ta}^{(g,n)})^2}{1 + \lambda_{ta}^{(g,n)}}, \tag{5.25}$$

where $\lambda_{ta}^{(g,n)}$ is computed as

$$\lambda_{ta}^{(g,n)} = \begin{cases} \left(\alpha_{ta}^{(g,n-1)}\right)^2 & n = 1, \\ (1-\alpha_{ta}^{(g,n-1)})^2 \lambda_{ta}^{(g,n-1)} + (\alpha_{ta}^{(g,n-1)})^2 & n > 1. \end{cases}$$

We then compute an estimate of the variance of $\bar{v}_{ta}^{(g,n)}$ as

$$(\bar{\sigma}_{ta}^2) = \mathrm{Var}[\bar{v}_{ta}^{(g,n)}] = \lambda_{ta}^{(g,n)} (s_{ta}^2)^{(g,n)}. \tag{5.26}$$

The weight we assign to each level of aggregation is given by

$$w_{ta}^{(g,n)} \propto \left((\bar{\sigma}_{ta}^2)^{(g,n)} + (\bar{\mu}_{ta}^{(g,n)})^2\right)^{-1}, \tag{5.27}$$

where the weights are normalized so they sum to one. The above estimates are easy to compute even for very large scale state spaces. This allows us to compute an estimate for all states, regardless of whether they were visited or not.

Table 5.1. Outcome state space size for a single decision epoch t.

No. of zones	Driver status (s_a)	Driver availability (m_a)	Percent completion of τ (h_a)	Progress to guarantee (y_a)	H
400	2	2	20	20	320,400

5.5.2 Structural Analysis

In this section, we derive some theoretical properties of the proposed model.

Relationship between the dual variables of (5.20) and value function estimates

We now show a result that formalizes the relationship between the dual variables of the LP (5.20) and the value function estimates. Let $\bar{v}_{ab}$ be the value function estimate when a driver with attributes a is matched to order with attributes b. Let $v_a, a \in \mathcal{A}$ denote the dual variables of constraint set (5.4) and $\kappa_b, b \in \mathcal{B}$ denote the dual variables of constraint set (5.5). Also, let the attribute vector $\mathcal{A} = \mathcal{A}^1 \cup \mathcal{A}^2 \cup \mathcal{A}^3$, where $\mathcal{A}^1$ is the subset of driver attribute such that $s_a = 1, h_a \geq \tau, y_a < W$, $\mathcal{A}^2 \subset \mathcal{A}$ is the subset of drivers satisfying $s_a = 1, (h_a < \tau)$ or $(h_a \geq$

τ and $y_a \geq W$), $\mathcal{A}^3$ is the set of remaining driver attributes, $\mathcal{A}^3 = \mathcal{A} \setminus \mathcal{A}^1 \cup \mathcal{A}^2$. The dual of LP (5.20) is

$$\min \sum_{a\in\mathcal{A}} R_{ta} v_a + \sum_{b\in\mathcal{B}} D_{tb}\kappa_b \tag{5.28}$$

$$\text{s.t. } v_a + \kappa_b \geq \theta_b w_b - \bar{\theta}_{ab} w_{ab} - \zeta\rho_a + \gamma\bar{v}_{ab} \qquad a \in \mathcal{A}^1, b : (a,b) \in \mathcal{D}_{ab} \tag{5.29}$$

$$v_a + \kappa_b \geq \theta_b w_b - \bar{\theta}_{ab} w_{ab} + \gamma\bar{v}_{ab} \qquad a \in \mathcal{A}^2, b : (a,b) \in \mathcal{D}_{ab} \tag{5.30}$$

$$v_a \geq -\zeta\rho_a + \gamma\bar{v}_{a0} \qquad a \in \mathcal{A}^1 \tag{5.31}$$

$$v_a \geq \gamma\bar{v}_{a0} \qquad a \in \mathcal{A}^2 \cup \mathcal{A}^3 \tag{5.32}$$

$$\kappa_b \geq 0 \qquad b \in \mathcal{B} \tag{5.33}$$

Lemma 2. The optimal values of the dual variables $v_a, a \in \mathcal{A}$, in the driver compensation guarantee model (5.20), have the following structure

$$v_a = \begin{cases} \max\left\{(-\zeta\rho_a + \gamma\bar{v}_{a0}), \max\limits_{b:(a,b)\in\mathcal{D}_{ab}} (\theta_b w_b - \bar{\theta}_{ab} w_{ab} - \zeta\rho_a + \gamma\bar{v}_{ab} - \kappa_b)\right\}, & a \in \mathcal{A}^1 \\ \max\left\{\gamma\bar{v}_{a0}, \max\limits_{b:(a,b)\in\mathcal{D}_{ab}} (\theta_b w_b - \bar{\theta}_{ab} w_{ab} + \gamma\bar{v}_{ab} - \kappa_b)\right\}, & a \in \mathcal{A}^2 \\ \gamma\bar{v}_{a0}, & a \in \mathcal{A}^3 \end{cases} \tag{5.34}$$

Proof.

- For $a \in \mathcal{A}^1$, the constraints imposed on a in the dual problem are (5.29) and (5.31). To minimize objective function (5.28) while satisfying (5.29) and (5.31), v_a takes the highest value of the RHS of the two constraint sets (5.29) and (5.31), i.e., $v_a = \max\left\{(-\zeta\rho_a + \gamma\bar{v}_{a0}), \max\limits_{b:(a,b)\in\mathcal{D}_{ab}} (\theta_b w_b - \bar{\theta}_{ab} w_{ab} - \zeta\rho_a + \gamma\bar{v}_{ab} - \kappa_b)\right\}$.

- For $a \in \mathcal{A}^2$, similar analysis holds. The constraints imposed on a in the dual problem are (5.30) and (5.32). To minimize objective function (5.28) while satisfying (5.30) and (5.32), v_a takes the greater value of the RHS of the two constraint sets, i.e., $v_a = \max\left\{\gamma\bar{v}_{a0}, \max\limits_{b:(a,b)\in\mathcal{D}_{ab}} (\theta_b w_b - \bar{\theta}_{ab} w_{ab} + \gamma\bar{v}_{ab} - \kappa_b)\right\}$.

- Finally, for $a \in \mathcal{A}^3$, only constraint set (5.32) is imposed, and thus to minimize the objective, $v_a = \gamma\bar{v}_{a0}$.

This concludes the proof. □

5.6 Computational Experiments

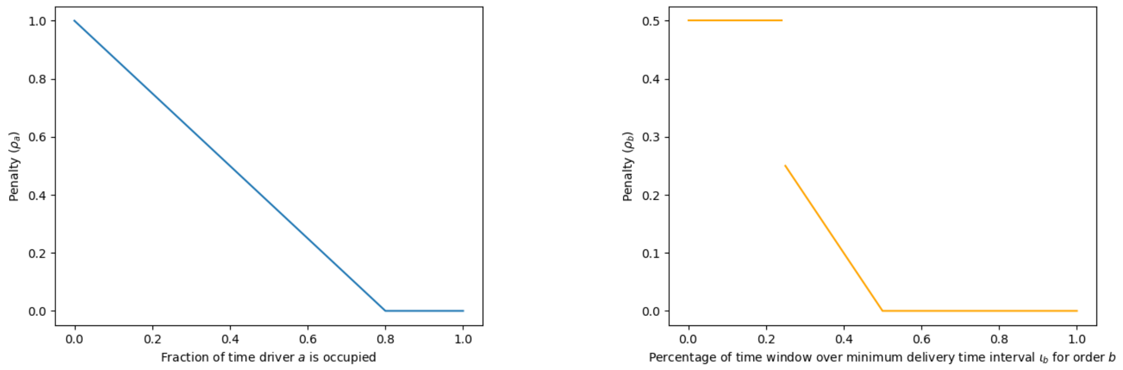
We test the proposed model and solution approach on multiple test instances to examine the effectiveness of the different computational policies and factors that affect solution quality. We first focus on the utilization guarantee, where we examine and compare multiple solution policies including the *myopic policy, parametric cost function approximation* (PCF), *value function approximation* (VFA) with and without heirarchical aggregation, as well as *VFA* when combined with *PCF*. We then compare the three proposed guranatee policies (utilization, minimum wage for activity window, and minimum wage while on-delivery) for restaurant delivery test instances, where we approximate the platform's profit and driver earning given data from the official website of UberEats (2021).

5.6.1 Data Generation

We test the solution approach using randomly generated instances. The number of stages or decision epochs is set to $T = 168$, with a duration of 5 minutes for each epoch, implying a 14-hour planning horizon. To ensure that driver utilization guarantees are considered as we approach the end of the horizon, the planning horizon is extended by 2 hours ($T = 192$) only when training the value functions. Drivers are initially uniformly distributed on a 10×10 grid network at $t = 0$, and continue arriving at rate $\mu = 2$ per decision epoch. One unit of distance in the grid network is assumed to be 3 km. Each disaggregate location is the center of 0.5×0.5 square, resulting in 400 disaggregate locations in the network. At a given decision epoch, the number of drivers exiting is the minimum of the random realization of the exit distribution with mean $\psi = 4$ per epoch, and the number of inactive drivers at t. New orders arrive following a Poisson process with mean $\lambda = 10$ per epoch. The delivery time window is set to 90 minutes from order announcement time. The origin and destination of each order is uniformally distributed in the network.

The parameters of the objective function of the utilization policy are set as follows. $\theta_b = 1$ implying that a unit of reward is obtained for each unit distance traveled between an origin and a destination. $\bar{\theta}_{ab} = \frac{1}{2\sqrt{200}}$, implying that the maximum disutility weight assigned to the distance between a driver and the origin of an order is 0.5. η, the scalar that converts the distance measure to a time estimate, is to equal 0.8. This means that it takes 4 minutes to travel 3 km, implying a vehicle speed of 45 km/hr.

The utilization guarantee W is set to 0.8, and the activity time window τ is set to 2 hours (24 consecutive decision epochs). Penalty functions ρ_a and ρ'_b are as shown in Figure 5.1. Figure 5.1a plots the shape of ρ_a. A driver meeting the minimum guaranteed utilization rate of 80% has a penalty of 0; drivers below this threshold are assigned a penalty that increases as utilization decreases. Figure 5.1b plots the shape of the function ρ'_b. An order is assigned a penalty (priority) score that is a function of the minimum delivery time ι_b it takes to travel from the origin to the destination of the order. Below a particular threshold (0.25 in the figure), the order is assigned a high priority as there is a high likelihood of losing it in consecutive decision epochs if it is not assigned. Over that threshold, the order is also assigned a priority, that decreases as the time window increases. Beyond another threshold (0.5 in the figure), the order is assigned a priority of 0, indicating that moving it forward to subsequent time periods is less risky. α, a parameter that scales the priority scores with the matching reward, is set to 1.



(a) Shape of driver matching penalty function ρ_a.

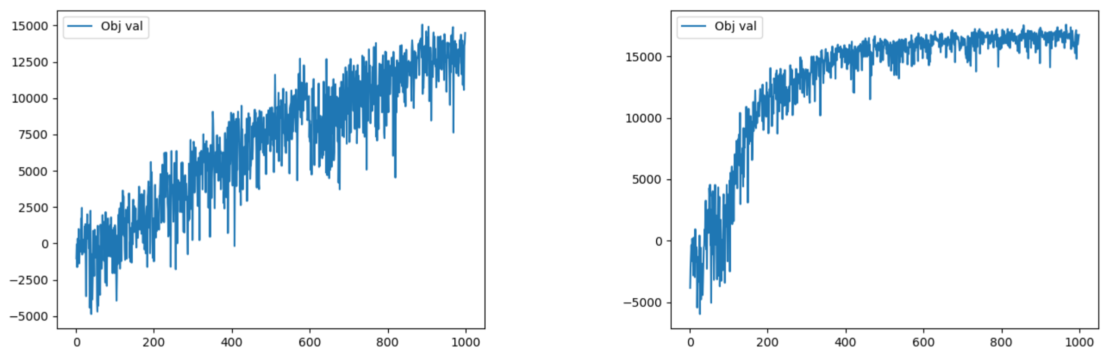
(b) Shape of demand loss risk penalty function ρ'_b.

Figure 5.1. Shape of penalty functions ρ_a and ρ'_b.

5.6.2 Value Function Approximation

In this section, we train the value functions as described by Algorithm (1) with and without spatial aggregation. A lookup table representation is used for approximating the value function. This means that in the value function approximation routine, we can estimate only the value of states that have been visited, at the disaggregate or aggregate levels. We start by initiating value functions for all $a \in \mathcal{A}$, $t \in \{0, 1, \ldots, T\}$ to 0, then run $N = 1000$ iterations to train the value functions using forward sample paths as outlined in Algorithm (1). Only when an attribute a at time t is encountered do we update its value function. Hierarchical aggregation is used to provide a better estimate on states that *may* be visited, not only states that have been visited.

Figure 5.2 shows the progression of the value function over iterations for the VFA algorithm with and without hierarchical aggregation. We observe that using hierarchical aggregation improves the convergence of the value function estimates. This figure shows the solution of a single instance, solved N times, each with the most updated value function at $n \in \{1, \ldots, N\}$. Notice how, when using aggregation, the total objective (sum of the objective function values over the planning horizon) has an upward trend as we learn the value functions, then levels off approximately when $N > 500$. This implies that the value functions converge, and further iterations will likely not change the estimates. For the disaggregate level, on the other hand, though we see an upward trend, it is apparent that there is high variability. This indicates that the N iterations are insufficient for learning the value functions, since driver attributes with disaggregate locations are not visited often enough.



(a) Total objective value per iteration without aggregation.

(b) Total objective value per iteration with aggregation.

Figure 5.2. Value function convergence with and without hierarchical aggregation.

5.6.3 Utilization Guarantee: Comparing Computational Policies

We test the proposed model and VFA solution approach (with and without aggregation) over multiple instances. We compare the performance of the propose solution policies against two benchmarks: the myopic policy (with and without PCF) and the base-case policy. Table 5.2 summarizes solution statistics for the average of 10 instances, solved using different computational policies. The table columns show the percentage of active drivers (i.e., those who receive the guarantee), the percentage of drivers that met the desired utilization and those who missed it, the average utilization of all active drivers. It also shows the percent deviation from the desired guarantee for drivers with a utilization below W, the percentage of drivers that *timed-out*, i.e., did not complete their activity window τ by the end of the planning horizon, the average total distance per decision epoch for all matches, and finally, the percentage of met demand. Note that in this set of instances, that the penalty parameter ζ is set to 1.

We notice that the aggregate VFA algorithm achieves the highest number of drivers meeting the desired utilization W, 30.99%, followed by the disaggregate VFA, 18.44%. Although under the myopic policy, only 4.46% of active drivers have a utilization of at least W and the average utilization is slightly higher than the that of the base-case policy, incorporating *parametric cost function approximation* (PCF) results in an average utilization (71.33%) comparable to that of the aggregate VFA algorithm (71.86%). This is interesting since the PCF algorithm simply modifies the cost function of the myopic policy, and does not require training value function estimates, which is an additional computational advantage.

Table 5.2 also shows that the aggregate VFA algorithm results in less efficient matches. Relative to the base-case policy, the average distance more than triples. It is also apparent that effectiveness of the VFA algorithm in reaching the desired utilization needs to be further improved, since having only 30.99% of drivers meeting the desired utilization is quite low. In what follows, we discuss the effect of varying the penalty parameter ζ, as well as the effect of combining

Table 5.2. Comparison of solution statistics of different computational policies (average of 10 instances).

Policy Type	Active Drivers (%)	Guarantee Met (%)	Guarantee Missed (%)	Avg Utilization (%)	Deviation (%)	Timed-out Drivers (%)	Avg Dist	Met Demand (%)
Disagg. VFA	55.47	18.44	68.18	68.28	-20.02	13.38	25.21	99.64
Agg. VFA	58.75	30.99	55.53	71.86	-18.24	13.48	37.08	99.87
Myopic	59.75	4.46	81.97	60.76	-25.74	13.57	15.42	99.81
Myopic - PCF	48.87	6.85	80.70	71.33	-12.19	12.44	20.89	99.83
Base case	62.56	3.72	82.16	57.82	-29.28	14.12	12.48	99.67

VFA with parametric cost function approximation.

Varying the penalty parameter ζ of missing the desired utilization

We investigate the benefit of increasing the penalty coefficient in the penalty function $F(a)$ on reaching the desired utilization threshold W. Based on the results in Table 5.2 and Figure 5.2, it is apparent that aggregating value functions improves the convergence of their estimates. Thus, in this section, only VFA with aggregation is considered. We also experiment with combining parametric cost function approximation within the VFA algorithm. Each row in Table 5.3 reports the results for the average of 10 instances.

We notice a significant improvement in the percentage of drivers that meet utilization threshold W as the value of ζ increases. Just increasing ζ to 10 from the initial value of 1, results in an additional 20.79% of active drivers meeting utilization W. The percentage of drivers meeting the utilization target continues to increase slightly with increases in ζ. Another significant improvement is achieved when VFA is combined with PCF as we see in the last row. The percentage of drivers meeting W increases to 86.67%. Considering only drivers that complete their activity window τ (i.e., excluding timed-out drivers), this translates to 99.74% of drivers meeting utilization W. However, this comes at an apparent cost in trip efficiency as the average distance per decision epoch is 45.79, which is more than triple that of the base-case policy, 12.48. Thus, we further examine the trade-off between utilization and trip efficiency for the different computational policies.

Table 5.3. Comparison of solution statistics of VFA with aggregation, varying ζ (average of 10 instances).

Penalty (ζ)	Active Drivers (%)	Guarantee Met (%)	Guarantee Missed (%)	Avg Utilization (%)	Deviation (%)	Timed-out Drivers (%)	Avg Dist	Met Demand (%)
$\zeta = 1$	58.75	30.99	55.53	71.86	-18.24	13.48	37.08	99.87
$\zeta = 10$	57.08	51.78	34.47	75.28	-19.10	13.75	39.41	99.89
$\zeta = 50$	56.59	58.47	28.51	76.24	-19.41	13.02	40.38	99.87
$\zeta = 100$	56.85	59.45	27.07	76.45	-19.50	13.48	40.20	99.87
$\zeta = 250$	56.80	61.02	25.45	76.81	-19.31	13.53	40.33	99.87
VFA-PCF, $\zeta = 250$	52.38	86.67	0.23	84.56	-2.19	13.10	45.79	99.89

Trade-off between utilization and trip efficiency

We compare the expected utilization of drivers for different matching policies and investigate the relationship between utilization and trip efficiency. Trip efficiency is defined as the expected distance traveled between a driver's origin and the origin of the order they are matched with. That is, when a driver is matched with a nearby order, the trip is more efficient than if they are matched with a farther order. Figure 5.3 plots the distribution of expected utilization, under different computational policies, for drivers that complete their activity window τ.

Figure 5.4 plots the total distance travelled per epoch for 10 instances of each of the matching policies. We observe that although combining aggregate VFA with PCF results in the highest fraction of drivers meeting utilization W, and much lower variability in drivers' utilization relative to the other policies, it also has the highest expected total trip distance. In general, we notice a inverse relationship between utilization and trip efficiency; as the mean utilization of drivers increases, the trip efficiency decreases.

Active drivers over the planning horizon

Figure 5.5 plots the number of active drivers throughout the planning horizon, under the VFA-PCF policy, the myopic PCF policy and the base case policy. We observe that the base policy keeps a somewhat steady number of active drivers throughout most of the planning horizon, oscillating between 25 and 30 drivers. The PCF policy (Fig. 5.5a), on the other hand, follows an approximate cyclical pattern, in the number of active drivers. This pattern repeats approximately every 24 epochs (or 2 hours), which is the duration of the activity time window τ. This indicates

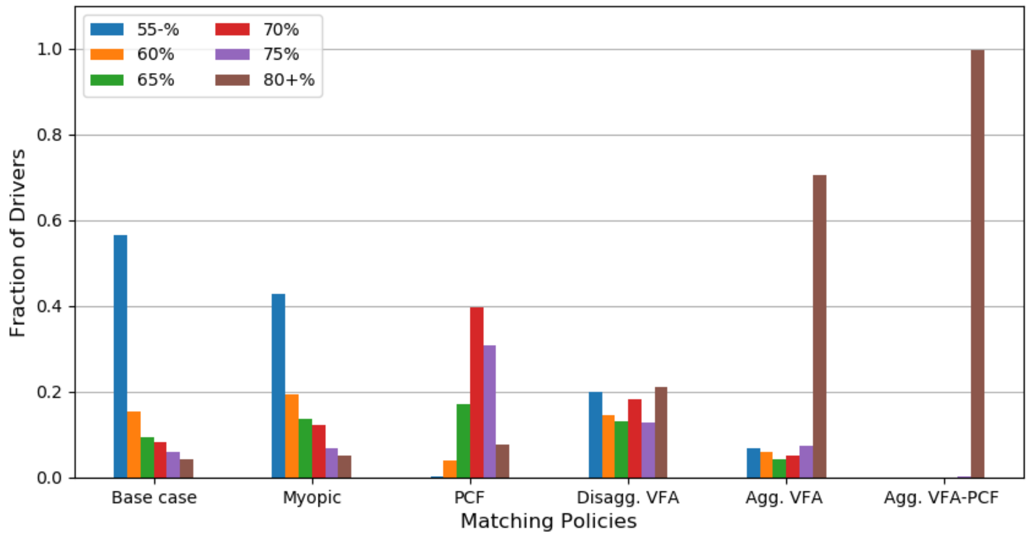


Figure 5.3. Driver utilization for different computational policies (average of 10 instances).

that the policy follows some kind of schedule for inviting active drivers, such that once the desired number of drivers is reached, no additional drivers are invited until a portion of active drivers complete their activity time window τ.

Examining cost function (5.14) and this pattern more closely, we notice that this happens because of the penalty imposed on missing the desired utilization and the fact that the PCF algorithm does not accurately capture the effect of decisions made now on the future. This penalty increases as the gap between the desired and actual utilization increases. The algorithm keeps inviting new drivers, favoring trip efficiency, until a particular threshold where the penalty cost becomes higher than the efficiency cost saving. At this point the supply of drivers levels off for a few decision epochs, then decreases as some drivers complete their activity window τ. When drivers' utilization improves, and after another approximate threshold point (lower points in the graph), the algorithm again favors efficiency since utilization of active drivers is high enough, and thus invites more drivers, and the cycle repeats.

Combining PCF with VFA allows us to reduce this oscillating effect, as observed in Figure 5.5b. The supply of drivers varies less significantly than that of the PCF policy. This indicates that the approximate value functions are helpful in predicting the downstream penalty of the low utilization of drivers.

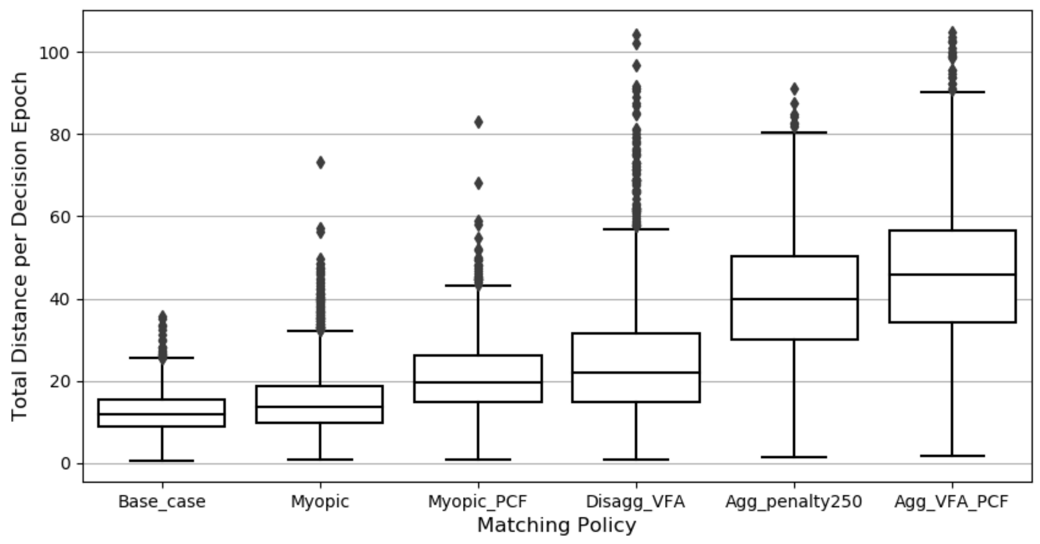
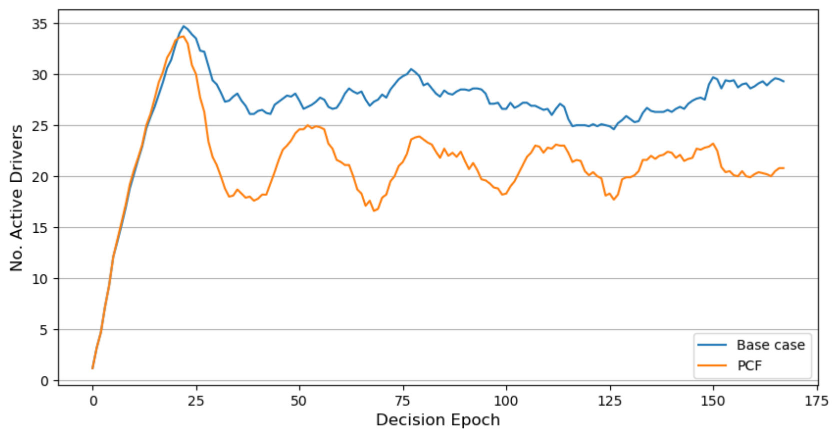


Figure 5.4. Total distance travelled per epoch for different computational policies.

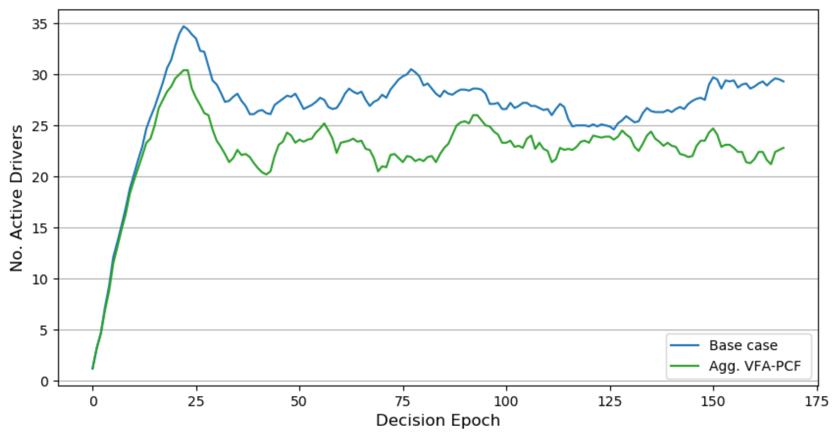
The effect of changes to τ and W

We now examine the effect of changes to activity time window τ and the target utilization level W on the solution. Each row in Table 5.4 shows the average of 10 instances solved for each combination of τ and W using the aggregate VFA combined with PCF. We first note that the duration of the activity window τ seems to have a more significant effect on the solution than the utilization target W. That is, for a fixed τ, varying W results in slight changes to the average utilization of drivers, the average distance and the percentage of demand met. Interestingly, when τ is set to 1 hour (12 epochs), we see a significant drop in percentage of demand met. This could be due to the following. First, in order to meet utilization target W in a short amount of time, the algorithm invites fewer drivers, which results in more uninvited drivers exiting the system. Second, the penalty of missing driver utilization guarantee in a shorter time is higher than the priority score set to prioritize expiring demand in parametric cost function (5.14), and thus the algorithm improves driver utilization at the expense of demand fulfillment.

Another interesting observation from the results in Table 5.4 is that when τ increases to 3 hours, trip efficiency improves (i.e., average distance decreases), while average utilization slightly decreases. This shows that more efficient driver-order matching can be achieved when the time window of meeting the utilization target increases.



(a) Parametric Cost Function.



(b) VFA - PCF.

Figure 5.5. Number of active drivers over the planning horizon.

5.6.4 Comparison of Guarantee Policies: Utilization vs. Wage Guarantee

We test and compare the three guarantee policies, and assess their impact on the platform and drivers. Though the proposed model could extend to other platforms, in this section we focus on platforms offering restaurant delivery. The revenue and cost functions are approximated from the information provided on UberEats official website.

The revenue collected by the platform is earned from two sources: the restaurants from which orders originate, and the consumers that place orders. Restaurants are charged 30% of the value of the meal, whereas consumers pay a delivery fee, a service fee, a small order fee, and delivery adjustment fee. For simplicity we focus on the first two types of consumer fees, since they are incurred by every order. Delivery fee depends on the location of the customer relative to the restaurant from which they are ordering, while service fee is 15% of the value of the meal, with a minimum of \$2.5 and a maximum of \$4.5 (UberEats, 2021).

Table 5.4. Varying activity time window τ and utilization target W.

τ	W	Active Drivers (%)	Guarantee Met (%)	Guarantee Missed (%)	Avg Utilization (%)	Deviation (%)	Timed-out Drivers (%)	Avg Dist	Met Demand (%)
1 hr	70%	16.55	88.81	0.00	97.97	0.00	11.19	13.80	51.49
	80%	14.20	84.97	0.00	97.53	0.00	15.03	11.69	46.61
	90%	11.89	89.93	0.00	97.22	0.00	10.07	10.26	40.17
2 hrs	70%	52.48	87.10	0.63	82.19	-2.05	12.27	44.33	99.86
	80%	52.38	86.67	0.23	84.56	-2.19	13.10	45.79	99.89
	90%	52.74	86.84	0.22	84.47	-0.63	12.94	46.09	99.90
3 hrs	70%	46.53	71.60	16.99	80.09	-7.04	11.41	30.34	96.73
	80%	47.82	82.73	4.84	81.31	-8.19	12.44	33.61	96.77
	90%	48.18	82.00	6.43	80.81	-6.70	11.57	33.93	96.75

The revenue that a platform earns is a function of the order attributes b, and depends on the distance between order origin and destination, w_b, and the value of the order, ϕ_b. To compute the variable cost of delivery as a function of distance, we use a step function. The variable cost of delivery for a distance no more than 2 units is \$4, between 2 and 5 units is \$6, and greater than 6 is \$7. The value of orders is uniformally distributed in the range [10,40], of which the platform receives 30% of the order value. The cost of completing an order is a function of order attributes b and driver attributes a to whom the order is assigned. According to Manzocco (2019), UberEats Toronto offers the following payment structure to drivers: \$1.5 per pickup, \$1 per delivery, \$0.49 per km from pickup to drop-off, and \$0.28 per minute spent fulfilling the order. Thus, the cost is a function of the distance between driver a and order b, w_{ab}, which is used to compute the variable per mileage and per minute payment, and the fixed pickup and drop-off cost per order, φ_b. The minimum wage for drivers is set to \$15 per hour.

As noted earlier, two types of wage guarantee policies are considered: wage guarantee during an activity window τ (policy 2), and wage guarantee only while delivering orders (policy 3). Table 5.5 compares the performance of those two policies with the utilization policy and the base case policies. Each row displays the average of 10 instances. To compute the average driver earning, we subtract 0.5 times the mileage compensation (\$0.245/km) from the revenue earned. This, however, does not factor in other vehicle expenses such as maintenance and depreciation costs.

We note from Table 5.5 that the wage guarantee policy (P2) results in lower average distance than the utilization policy. The average utilization of drivers is lower under policy 2, but the

Table 5.5. Comparison of guarantee policies (average of 10 instances).

Guarantee Policy Type	Perc. Active	Avg Utilization (%)	Avg Driver Earning/hr ($)	Avg Platform Pay/hr ($)	Timed-out Drivers (%)	Avg dist	Perc. Met Demand
Utilization (P1)	48.21	85.87	31.97	0.00	12.80	34.01	99.89
Wage (P2)	54.08	68.17	26.08	0.04	13.15	23.01	99.87
Wage - when Active (P3)	83.79	37.70	15.38	0.00	0.00	19.32	99.85
Base case	82.98	37.70	15.54	0.00	0.00	19.38	99.82

average driver profit under both policies is well over the minimum wage of $15.

The wage guarantee policy while drivers are delivering orders (policy 3), without an activity window τ, results in a very similar solution to that of the base case policy. Driver average profit ($15.38 and $15.54, respectively) and average utilization (37.7%) is much lower under those policies relative to policies 1 and 2.

Are drivers better off under guarantee policies? Is the platform worse off?

Figure 5.6 plots the average total distance per decision epoch vs. the average hourly earning for drivers under the four considered policies. We note that, although the utilization policy (P1) results in more inefficient routes in terms of total distance travelled, driver earning under this policy is the highest. The minimum wage guarantee policy (P2) reduces both average earning and distance, indicating that the lower compensation is for the sake of limiting matching drivers with farther orders. On the other hand, policy 3, where drivers are guaranteed the minimum wage only while delivering orders, does not provide any higher compensation to drivers relative to the base policy.

Figure 5.7 plots the average platform profit per decision epoch vs. the average hourly earning per driver for the four considered policies. We see that under policy 3, where drivers receive a minimum wage while on delivery, the platform's profit is similar to that of the base case, and so is average driver earning. We also note that although policy 1 is the best for drivers in terms of hourly earnings, it results in the worst average profit for the platform. Policy 2, on the other hand, results in a small expected profit reduction for the platform ($2.48 per epoch), relative to the base policy, but a considerably higher hourly rate increase for drivers (an increase of $10.54).

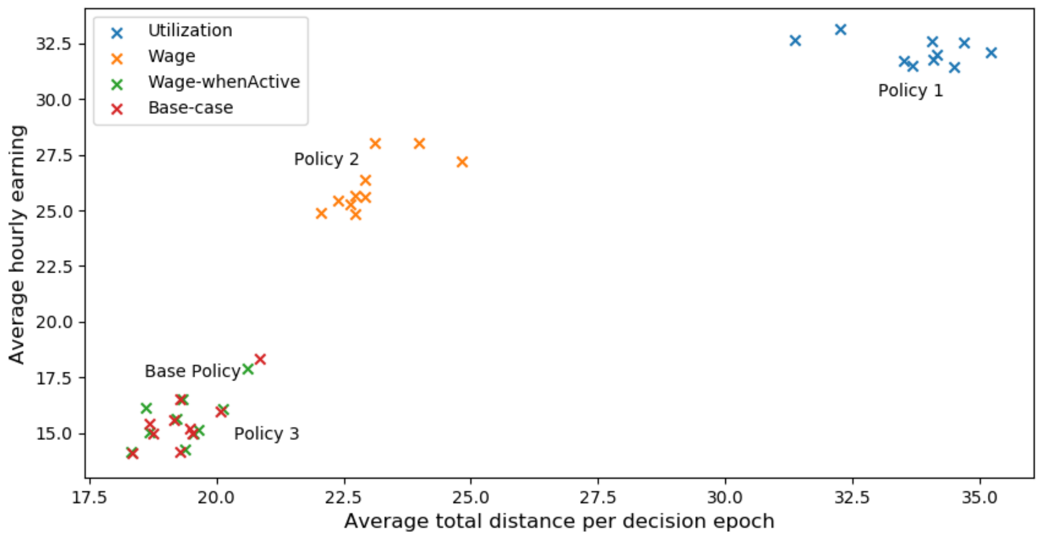


Figure 5.6. Total average distance travelled per epoch vs. average driver hourly rate.

5.7 Concluding Remarks

This research studies the integration of compensation guarantees for sharing economy workers within a platform's dynamic matching framework. Those compensation guarantees aim to provide earning protection to gig economy workers, while maintaining the work hour flexibility promoted by the sharing economy. Drivers are self-scheduling and do not pre-announce their schedules. Once a driver is active in the platform, i.e., they are available and they receive their first invitation to deliver an order, the platform offers them an earning guarantee. We propose, model and test three types of earning guarantees: (i) a maximum utilization guarantee up to a utilization target W, for an activity window of length τ, (ii) an hourly minimum wage guarantee for an activity window of τ consecutive hours, and (iii) an hourly minimum wage only while drivers are delivering orders.

To capture the dynamic and stochastic nature of matching drivers to orders in a sharing economy platform, we model the problem as a Markov decision process. The detailed driver and order attributes needed to accurately capture dynamics of the system, and the spatial and temporal features of the problem, make the problem high dimensional. Thus, solving it to optimality using Bellman's equation is intractable for reasonable size problems. We thus turn to approximate dynamic programming, particularly, value function approximation. The value function estimates mainly focus on driver attributes, and take advantage of the sparsity of the drivers attribute vector, thus utilize a linear approximation representation. To further enhance the convergence of the value function estimates, we use spatial hierarchical aggregation to exploit the spatial correlation between value functions. This helps us increase the number of samples for the value function

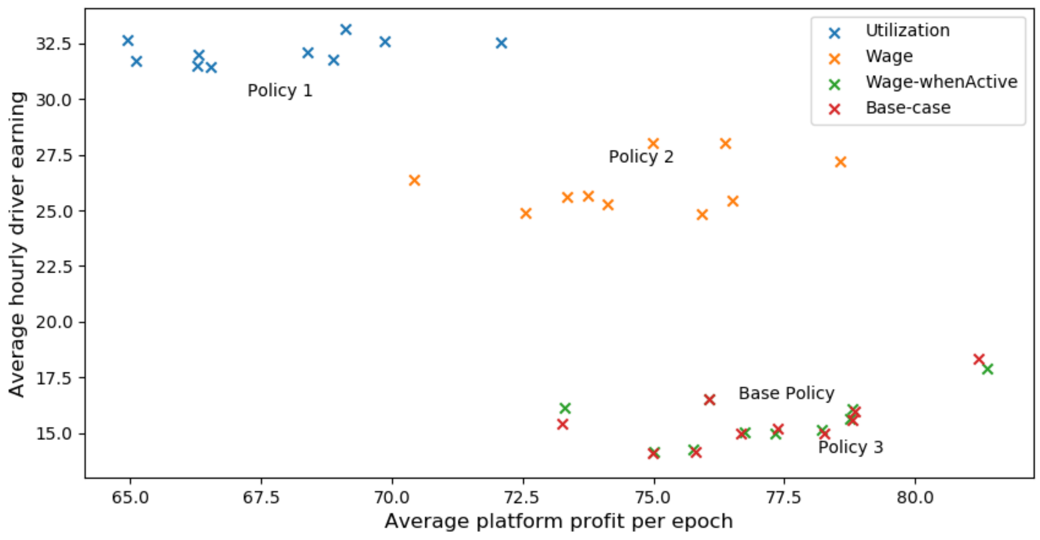


Figure 5.7. Average platform profit per epoch vs. average driver hourly rate.

estimates, thus improving their estimation.

We conduct extensive computational testing to assess the performance of the proposed value function approximation, and to compare the different types of guarantee policies. For the utilization guarantee policy, we observe that combining parametric cost function approximation with value function approximation results in the highest number of drivers meeting the target utilization level. However, this comes at a clear cost in trip efficiency; the higher the utilization of active drivers, the longer the distance between drivers and the origins of their matched orders. To compare the different guarantee policies, we extract data from a crowdsourced restaurant delivery platform (UberEats) to estimate the platform's matching profit and driver compensation. We observe that drivers receive the highest average earning under the utilization policy, although trip efficiency is significantly worse than all other policies. The wage policy for an activity window (policy 2) results in a slight increase in average distance relative to the base case policy, but an hourly wage increase of \$10.54 for active drivers. The wage policy when delivering orders (policy 3) results in almost identical average driver earning, utilization and average distance, relative to the base case. We find that policy 2, the wage guarantee for an activity window, results in the best trade-off in improving driver earnings, while causing only a slight reduction to the platform's profit.

Some interesting directions for future work include extending the proposed model to other sharing economy operations, e.g., ridehailing, where the time window for demand fulfillment is very short. From a methodological perspective, tuning the parametric cost function approximation policy may further improve its performance. Furthermore, experimenting with other methods of approximating the value function, such as using regression and other parametric models, is a promising direction. Such approaches help us obtain estimates of unvisited states by fitting a parametric function based the features of visited states.

Chapter 6

Conclusion and Future Research

The rapid development of e-retailing in the last decade necessitates the creation of efficient and responsive distribution plans, that account for different sources of uncertainty within the supply chain. This thesis investigates three stochastic distribution planning problems, faced by e-retailers at different stages of the supply chain.

Chapter 2 presents a distribution planning problem in a transshipment network under stochastic customer demand. This work addresses the gap in the literature concerning the limited work that accounts for randomness in freight distribution planning with intermediate facilities. We study the problem from the perspective of a 3PL, and formulate a two-stage stochastic programming model with recourse that minimizes transportation, inventory and outsourcing costs. The model is then solved using sample average approximation (SAA), which results in reasonable optimality gaps for problem instances of varying sizes. Computational experiments highlight the robustness and cost-effectiveness of the proposed stochastic model compared to its deterministic counterpart. They also provide insights on the conditions under which the model achieves notable cost savings.

We then examine a later stage in the supply chain, in particular, last-mile same-day delivery from distribution centers, stores or restaurants to the customer's doorstep. We focus on *crowd-sourced delivery*, an emergent last-mile delivery system where freelance self-scheduling drivers complete last mile deliveries with their own vehicles. Chapter 3 presents a comprehensive review of this system in terms of both academic work and industry practice. It also highlights new challenges that this system brings about, relative to more traditional transportation and delivery systems. The following two chapters study two important research questions faced by crowdsourced delivery platforms.

Chapter 4 introduces, models and assesses the use of heatmaps as a control lever to balance supply of drivers and demand in a crowdsourced delivery system. Heatmaps communicate to drivers regions with driver shortage, where drivers have a higher probability of being matched with a revenue-producing order. We formulate an MDP model to dynamically choose heatmap and matching decisions that maximize demand fulfillment within a planning horizon. A stochastic lookahead policy, based on approximate dynamic programming, is proposed to obtain good quality solutions in a reasonable amount of time. We then propose a computational enhancement method that decomposes heatmap and matching decisions without affecting solution quality. Based on computational experiments extracted from the Chicago ridehailing dataset, we find that heatmaps are most effective in improving demand fulfillment when the supply of drivers exceeds expected demand, and when the demand patterns in the service network are imbalanced.

Chapter 5 investigates the integration of driver compensation guarantees with dynamic matching decisions of a crowdsourced delivery platform. This addresses the common criticism of the lack of protection for workers in the sharing economy, by proposing compensation guarantees to drivers, while maintaining the work hour flexibility of the sharing economy. We propose and model three types of compensation guarantees, that are either utilization-based or wage-based. The problem of dynamically matching drivers and orders, while ensuring a particular type of driver compensation guarantee is met, is formulated as an MDP model. The problem is then solved by value function approximation, so as to obtain good quality solutions efficiently. Extensive computational testing is presented to assess the proposed solution approach, and evaluate the impact of the different types of guarantees on both the platform and the drivers.

While we presented several possible extensions to our work in previous chapters, there are many other interesting directions in the broad area of stochastic distribution planning. For instance, considering crowdsourcing of distribution activities in other phases of the supply chain, and not just the last mile. Under this setting, managing the stochastic supply of crowdsourced workers is very critical, as failure to accurately meet the supply needs could result in a greater downstream impact on the supply chain. Thus, determining the best trade-off between work flexibility and maintaining a reliable level of supply that accurately fulfills the needs of the supply chain is essential.

Another interesting direction is examining and modeling other innovations in transportation and delivery. For example, this can include the use of autonomous trucks and delivery droids. Those innovative delivery methods could potentially pose issues of endogenous uncertainty, i.e., demand of customers may be affected by the delivery method employed, especially if deliveries are not made to the customer doorstep as a result. Therefore, studying the effectiveness of the partial or full adoption of such delivery technologies on fulfilling orders with short time windows, while also capturing the effect of possible demand pattern changes as a result of this adoption is an interesting area of research.

References

Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450 – 1464, 2011.

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

Aliaa Alnaggar, Fatma Gzara, and James H Bookbinder. Crowdsourced delivery: A review of platforms and academic literature. *Omega*, page 102139, 2019.

Aliaa Alnaggar, Fatma Gzara, and James H Bookbinder. Distribution planning with random demand and recourse in a transshipment network. *EURO Journal on Transportation and Logistics*, 9(1):100007, 2020.

Amazon.com. Amazon flex. https://flex.amazon.com/, 2019. Accessed on 2019-04-08.

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

Alp M Arslan, Niels Agatz, Leo Kroon, and Rob Zuidwijk. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 2018.

R. Bai, S. W. Wallace, J. Li, and A. Y. Chong. Stochastic service network design with rerouting. *Transportation Research Part B: Methodological*, 60:50 – 65, 2014.

Siddhartha Banerjee, Carlos Riquelme, and Ramesh Johari. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258*, 2015.

Jonathan F Bard, Canan Binici, et al. Staff scheduling at the united states postal service. *Computers & Operations Research*, 30(5):745–771, 2003.

Saif Benjaafar and Ming Hu. Operations management in the age of the sharing economy: what is old and what is new? *Manufacturing & Service Operations Management*, 22(1):93–101, 2020.

Saif Benjaafar, Jian-Ya Ding, Guangwen Kong, and Terry Taylor. Labor welfare in on-demand service platforms. *Available at SSRN 3102736*, 2019.

Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

O. Berman and Q. Wang. Inbound logistic planning: Minimizing transportation and inventory cost. *Transportation Science*, 40(3):287–299, 2006.

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

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

J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

S. Bock. Real-time control of freight forwarder transportation networks by integrating multimodal transport chains. *European Journal of Operational Research*, 200(3):733 – 746, 2010.

J. H. Bookbinder, Q. Cai, and Q. He. Shipment consolidation by private carrier: The discrete time and discrete quantity case. *Stochastic Models*, 27(4):664–686, 2011.

Nondita Bose. Walmart trials grocery delivery to rival Amazon Flex, 2018. URL https://www.reuters.com/article/us-walmart-delivery/walmart-trials-grocery-delivery-to-rival-amazon-flex-idUSKCN1LL1Q1.

BuddyTruk. https://www.buddytruk.com/, 2019. Accessed on 2019-04-09.

Reinhard Bürgy, Hélène Michon-Lacaze, and Guy Desaulniers. Employee scheduling with short demand perturbations and extensible shifts. *Omega*, 2018.

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

X. Cai, J. Chen, Y. Xiao, X. Xu, and G. Yu. Fresh-product supply chain management with logistics outsourcing. *Omega*, 41(4):752–765, 2013.

Valentina Carbone, Aurélien Rouquet, and Christine Roussat. Carried away by the crowd": what types of logistics characterise collaborative consumption. In *1st International Workshop on Sharing Econom, Utrecht, Netherlands*, 2015.

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

Peter Congdon. *Bayesian statistical modelling*, volume 704. John Wiley & Sons, 2007.

T. G. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

Teodor Gabriel Crainic, Mike Hewitt, and Walter Rei. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research*, 43:90 – 99, 2014.

K. L. Croxton, B. Gendron, and T. L. Magnanti. Models and methods for merge-in-transit operations. *Transportation Science*, 37(1):1–22, 2003.

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.

Deliv. https://www.deliv.co/, 2019. Accessed on 2019-04-09.

Aashwinikumar Devari, Alexander G Nikolaev, and Qing He. Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers. *Transportation Research Part E: Logistics and Transportation Review*, 105:105–122, 2017.

DHL. Dhl crowd sources deliveries in Stockholm with MyWays, 2013. URL http://www.dhl.com/en/press/releases/releases_2013/logistics/dhl_crowd_sources_deliveries_in_stockholm_with_myways.html#.XKywEphKiHs.

DoorDash. https://www.doordash.com/, 2019. Accessed on 2019-04-09.

DoorDash. https://help.doordash.com/consumers/s/article/What-fees-do-I-pay?language=en_US, 2021. Accessed on 2021-06-29.

Ezzeddine Fatnassi, Jouhaina Chaouachi, and Walid Klibi. Planning and operating a shared goods and passengers on-demand rapid transit system for sustainable city-logistics. *Transportation Research Part B: Methodological*, 81:440–460, 2015.

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

Katarzyna Gdowska, Ana Viana, and João Pedro Pedroso. Stochastic last-mile delivery with crowdshipping. *Transportation research procedia*, 30:90–100, 2018.

Abraham George, Warren B Powell, and Sanjeev R Kulkarni. Value function approximation using multiple aggregation for multiattribute resource management. *Journal of Machine Learning Research*, 9(Oct):2079–2111, 2008.

Veaceslav Ghilas, Emrah Demir, and Tom Van Woensel. Integrating passenger and freight transportation: model formulation and insights. In *Beta Working Papers WP 441*. Technische Universiteit Eindhoven, 2013.

Veaceslav Ghilas, Emrah Demir, and Tom Van Woensel. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12–30, 2016.

Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

G. Guastaroba, M. G. Speranza, and D. Vigo. Intermediate facilities in freight transportation planning: A survey. *Transportation Science*, 2016.

Mohammad Reza Habibi, Alexander Davidson, and Michel Laroche. What managers should know about the sharing economy. *Business Horizons*, 60(1):113 – 121, 2017.

Sarah Halzack. Amazon Flex: the retailer's Uber-like effort to bring you packages, 2015. URL https://www.washingtonpost.com/news/business/wp/2015/09/29/amazon-flex-the-retailers-uber-like-effort-to-bring-you-packages/?noredirect=on&utm_term=.13ed1725b857.

J. K. Higginson and J. H. Bookbinder. Policy recommendations for a shipment-consolidation program. *Journal of Business Logistics*, 15(1):87–112, 1994.

Hitch. http://www.hitchit.co/, 2019. Accessed on 2019-04-09.

Arild Hoff, Arnt-Gunnar Lium, Arne Løkketangen, and Teodor Gabriel Crainic. A metaheuristic for stochastic service network design. *Journal of Heuristics*, 16(5):653–679, 2010.

Bin Hu, Ming Hu, and Han Zhu. Surge pricing and two-sided temporal responses in ride hailing. *Manufacturing & Service Operations Management*, 2021.

Ming Hu and Yun Zhou. Price, wage and fixed commission in on-demand matching. *Available at SSRN 2949513*, 2019.

Instacart. https://www.instacart.com/, 2019. Accessed on 2019-04-09.

Nabin Kafle, Bo Zou, and Jane Lin. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation research part B: methodological*, 99:62–82, 2017.

Kanga. https://www.getkanga.com/, 2019. Accessed on 2019-04-09.

A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502, February 2002.

M. A. Krajewska and H. Kopfer. Transportation planning in freight forwarding companies: Tabu search algorithm for the integrated operational transportation planning problem. *European Journal of Operational Research*, 197(2):741 – 751, 2009.

Y. E. Kılıç and U. R. Tuzkaya. A two-stage stochastic mixed-integer programming approach to physical distribution network design. *International Journal of Production Research*, 53(4): 1291–1306, 2015.

C. J. Langley. *2018 Third-Party Logistics Study*. Infosys, Penske, Korn Ferry, PennState Smeal, 2017. URL http://www.3plstudy.com/.

Alan Lee and Martin Savelsbergh. Dynamic ridesharing: Is there a role for dedicated drivers? *Transportation Research Part B: Methodological*, 81:483–497, 2015.

Yanzhe Murray Lei, Stefanus Jasin, Jingyi Wang, Houtao Deng, and Jagannath Putrevu. Dynamic workforce acquisition for crowdsourced last-mile delivery platforms. 2020.

Baoxiang Li, Dmitry Krushinsky, Hajo A Reijers, and Tom Van Woensel. The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1): 31–40, 2014.

Baoxiang Li, Dmitry Krushinsky, Tom Van Woensel, and Hajo A Reijers. An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research*, 66:170–180, 2016.

A. G. Lium, T. G. Crainic, and S. W. Wallace. A study of demand stochasticity in service network design. *Transportation Science*, 43(2):144–157, 2009.

Alice Lu, Peter Frazier, and Oren Kislevt. Surge pricing moves uber's driver partners. *Available at SSRN 3180246*, 2018.

Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Demetrio Laganà. The vehicle routing problem with occasional drivers and time windows. In *International Conference on Optimization and Decision Science*, pages 577–587. Springer, 2017.

D. P. Mak, W. K.and Morton and R. K. Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations research letters*, 24(1-2):47–56, 1999.

Natalia Manzocco. What it's like working for a food delivery app in toronto, 2019. URL https://nowtoronto.com/food-and-drink/food/food-delivery-toronto-pay.

Neda Masoud and R Jayakrishnan. A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological*, 106: 218–236, 2017.

Renaud Masson, Anna Trentini, Fabien Lehuédé, Nicolas Malhéné, Olivier Péton, and Houda Tlahig. Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 6(1):81–109, 2017.

J. M. Masters. The effects of freight consolidation on customer service. *Journal of Business Logistics*, 2(1):55–74, 1980.

Abood Mourad, Jakob Puchinger, and Chengbin Chu. A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*, 2019.

Kianoush Mousavi, Merve Bodur, and Matthew J Roorda. Stochastic last-mile delivery with crowd-shipping and mobile depots. 2020.

F. Mutlu, S. Çetinkaya, and J. H. Bookbinder. An analytical model for computing the optimal time-and-quantity-based policy for consolidated shipments. *Iie Transactions*, 42(5):367–377, 2010.

Nimber. https://www.nimber.com/, 2019. Accessed on 2019-04-09.

Sara O'Brien. Uber, lyft prices go up in nyc as new driver minimum wage law takes effect, 2019. URL https://www.cnn.com/2019/02/01/tech/uber-nyc-rates/index.html.

PiggyBee. https://www.piggybee.com/, 2019. Accessed on 2019-04-09.

Postmates. https://postmates.com/, 2019. Accessed on 2019-04-09.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality, Second Edition*, volume 1. John Wiley & Sons, 2011.

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

PricewaterhouseCoopers. The sharing economy - sizing the revenue opportunity. http://www.pwc.co.uk/issues/megatrends/collisions/sharingeconomy/the-sharing-economy-sizing-the-revenue-opportunity.html, 2015. Accessed on 2019-04-08.

Alissa Pump. *City of Chicago Community Areas and Sides*, 2012. City of Chicago Data Portal. Projection: SPCS83 Illinois East zone. (US Survey feet) (Transverse Mercator) Datum: North American Datum 1983/Illinois East.

Wei Qi, Lefei Li, Sheng Liu, and Zuo-Jun Max Shen. Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact. *Manufacturing & Service Operations Management*, 2018.

H. Qin, Z. Zhang, Z. Qi, and A. Lim. The freight consolidation and containerization problem. *European Journal of Operational Research*, 234(1):37 – 48, 2014.

María I Restrepo, Bernard Gendron, and Louis-Martin Rousseau. A two-stage stochastic programming approach for multi-activity tour scheduling. *European Journal of Operational Research*, 262(2):620–635, 2017.

Roadie. https://www.roadie.com/, 2019. Accessed on 2019-04-09.

Jean-François Rougès and Benoit Montreuil. Crowdsourcing delivery: New interconnected business models to reinvent delivery. In *1st international physical internet conference*, pages 1–19, 2014.

Andrzej Ruszczyński. Commentary—post-decision states and separable approximations are powerful tools of approximate dynamic programming. *INFORMS Journal on Computing*, 22(1): 20–22, 2010.

A. Shapiro and A. Philpott. A tutorial on stochastic programming, March 2007. URL https://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf.

Shipt. https://www.shipt.com/, 2019. Accessed on 2019-04-09.

Hugo P Simao, Jeff Day, Abraham P George, Ted Gifford, John Nienow, and Warren B Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197, 2009.

H. Song, V. N. Hsu, and R. K. Cheung. Distribution coordination between suppliers and customers with a consolidation center. *Operations Research*, 56(5):1264–1277, 2008.

David Soto Setzke, Christoph Pflügler, Maximilian Schreieck, Sven Fröhlich, Manuel Wiesche, and Helmut Krcmar. Matching drivers and transportation requests in crowdsourced delivery systems. *Twenty-third Americas Conference on Information Systems*, 2017.

Brooke Staggs. California's gig worker law faces push back even from some it's intended to help, 2020. URL https://www.ocregister.com/2020/02/03/californias-gig-worker-law-faces-push-back-even-from-some-its-intended-to-help/.

Statista. Retail e-commerce sales in the United States from 2017 to 2023 (in million U.S. dollars). https://www.statista.com/statistics/272391/us-retail-e-commerce-sales-forecast/, 2019. Accessed on 2019-04-08.

M. SteadieSeifi, N.P. Dellaert, W. Nuijten, T. V. Woensel, and R. Raoufi. Multimodal freight transportation planning: A literature review. *European Journal of Operational Research*, 233 (1):1 – 15, 2014.

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

Jeremiah Titone and Robert Goch. The economics of quick service restaurant delivery partnerships. *Proceedings of the Northeast Business & Economics Association*, 2018.

TNP-Chicago-Trips. Transportation network providers - trips. https://data.cityofchicago.org/Transportation/Transportation-Network-Providers-Trips/m6dm-c72p, 2019. Accessed on 2021-06-20.

P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Number 18 in MOS-SIAM Series on Optimization. SIAM, 2014.

Truxx. https://truxxit.com/, 2019. Accessed on 2019-04-09.

Uber-UK. Uber 'willing to change' as drivers get minimum wage, holiday pay and pensions, 2021. URL https://www.bbc.com/news/business-56412397.

UberEats. https://www.ubereats.com/, 2019. Accessed on 2019-04-09.

UberEats. https://restaurants.ubereats.com/ca/en/, 2021. Accessed on 2021-04-07.

UberFreight. https://www.uberfreight.com/, 2019. Accessed on 2019-04-09.

Marlin Ulmer and Martin Savelsbergh. Workforce scheduling in the era of crowdsourced delivery. *Transportation Science*, 2020.

Marlin W Ulmer, Barrett W Thomas, and Dirk C Mattfeld. Preemptive depot returns for dynamic same-day delivery. *EURO journal on Transportation and Logistics*, 8(4):327–361, 2019.

Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European journal of operational research*, 226(3):367–385, 2013.

Stacy A Voccia, Ann Melissa Campbell, and Barrett W Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

Walmart. Spark delivery. https://www.drive4spark.com/, 2019. Accessed on 2019-04-09.

X. Wang and S. W. Wallace. Stochastic scheduled service network design in the presence of a spot market for excess capacity. *EURO Journal on Transportation and Logistics*, 5(4):393–413, Dec 2016.

X. Wang, H. Kopfer, and M. Gendreau. Operational transportation planning of freight forwarding companies in horizontal coalitions. *European Journal of Operational Research*, 237(3):1133 – 1141, 2014.

Xin Wang, Teodor Gabriel Crainic, and Stein W Wallace. *Stochastic scheduled service network design: The value of deterministic solutions.* CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise ..., 2016.

N. Wieberneit. Service network design for freight transportation: a review. *OR spectrum*, 30(1): 77–112, 2008.

Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization.* John Wiley & Sons, 1999.

Jiayi Joey Yu, Christopher S Tang, Zuo-Jun Max Shen, and Xiqun Chen. Should on-demand ride services be regulated? an analysis of chinese government policies. *An Analysis of Chinese Government Policies (June 16, 2017)*, 2017.

S. Çetinkaya and J. H. Bookbinder. Stochastic models for the dispatch of consolidated shipments. *Transportation Research Part B: Methodological*, 37(8):747 – 768, 2003.

APPENDICES

Appendix A

Chapter 2 Appendix

A.1 Data generation

We explain below the specific procedure used to generate data for our computational experiments.

A.1.1 The network of suppliers and customers

Locations of suppliers and customers are randomly generated within a radius of 1,000 miles from the consolidation center. We locate suppliers and customers on opposite sides of the 1,000-mile-radius circle, such that the consolidation center is a natural middle point. We do that since, from a practical point of view, suppliers in some long-haul freight transportation applications are clustered in a different geographical area, which may be overseas. We conducted some experiments with suppliers and customers randomly located throughout the 1,000-mile-radius circle; we observed very similar results to when they are located on opposite sides. This is attributed to the fact that our network is a pure transshipment network where direct deliveries between suppliers and customers are not possible. In order to justify the need for consolidation, supplier-customer shipments (i, j) are selected on the network such that the distance between supplier i and the consolidation center plus the distance from the consolidation center to customer j is at most 1.25 times the direct distance from i to j.

A.1.2 Distribution of demand, holding cost, and transportation option capacity

Each shipment (i, j) has a uniform demand distribution; the lower bound of the distribution is generated in the range U[300,450] and the width of the distribution is set at 30%. For example, if shipment (i, j) has a lower bound of 350, with a 30% width the demand follows a uniform distribution U[350,455]. The holding cost of shipments from a given supplier i is a variable cost per volumetric unit, per time unit, generated uniformly as h_i=U[0.005,0.01].

The capacity of inbound and outbound transportation options is determined as follows. For a given inbound transportation option q, the average demand of all customer orders for which option q is feasible, denoted as $\bar{d}_{iq}$, is calculated. Option $q \in Q_i$ is feasible for customer j if at least one outbound option $l \in L_j$ for customer j leaves the consolidation center after inbound option q arrives there. Capacity C_{iq} of option q is then generated as $C_{iq} = \gamma \bar{d}_{iq}$, where γ is in between $[1.0, 1.3]$; the exact value of γ is specified when generating data sets in Section 2.5.1. This capacity is then rounded up to be in multiples of 10 units. The capacity of an outbound option is generated in a similar manner, $C_{jl} = \gamma \bar{d}_{jl}$.

A.1.3 Supplier and Customer Data

For supplier $i \in I$, a release time υ_i is generated in the range [0,100]. A supplier has a number of inbound transportation options $q \in Q_i$ with arrival times $\tau_{iq} \in X_i$ between [100,500]. The number of options and their arrival times are specified in the data sets in Section 2.5.1.

For option q with arrival time τ_{iq}, the transportation cost is expressed as $f_i(\tau_{iq}) = \theta_i \rho_i(\tau_{iq}) \frac{C_{iq}}{\xi}$, where θ_i is the scale factor of supplier i and is randomly generated in U[0.5,1.5]. $\rho_i(\tau_{iq})$ is the transportation rate corresponding to arrival time τ_{iq}, C_{iq} is the capacity of option q, and ξ is the baseline capacity that is assumed to be 4000 units. This capacity level corresponds to approximately an average consolidated demand of 8 customers, following the demand distributions outlined above.

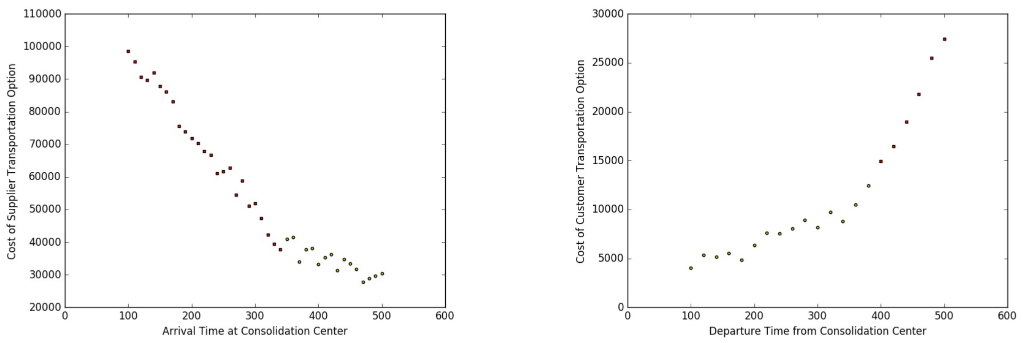
We generate $\rho_i(\tau_{iq})$ as follows. First, a baseline transportation rate $\bar{\rho}_i$ is generated as $\bar{\rho}_i = \delta_i \cdot \rho$, where δ_i is the distance and ρ is the average unit rate that is uniformly distributed in the range U[30,50]. A baseline transportation time ι_i is then generated as $\iota_i = \delta_i/v$, where v is the average speed per time unit, generated in the range U[2,3]. If the transportation time is shorter than the baseline, i.e., $\tau_{iq} - \upsilon_i \leq \iota_i$, the transportation rate is higher than the baseline rate; $\rho_i(\tau_{iq}) = \bar{\rho}_i + 2 \cdot \bar{\rho}_i \cdot [1 - ((\tau_{iq} - \upsilon_i)/\iota_i)] + \bar{\rho}_i \cdot \epsilon$, where the first term denotes the baseline rate, the second represents the extra cost to make the transportation time shorter than the baseline time, and the third term is some random perturbation in which ϵ is generated in the range U[-0.1,0.1]. On the other hand, if the transportation time is longer than the baseline, i.e., $\tau_{iq} - \upsilon_i > \iota_i$, we set $\rho_i(\tau_{iq}) = \bar{\rho}_i - 0.1 \cdot \bar{\rho}_i \cdot [1 - (\iota_i/(\tau_{iq} - \upsilon_i))] + \bar{\rho}_i \cdot \epsilon$. So, transportation options that need less time to reach the consolidation center, once the consolidated shipment is released, are faster options, and therefore have higher rates. The supplier cost function for the *baseline* capacity ξ is plotted in Figure A.1a.

Finally, we generate the cost of shipping through a spot-market carrier π_i, for each supplier $i \in I$. As mentioned earlier, this is a per unit cost composed of two elements: (a) the expected inbound spot market rate per unit and (b) the 3PL's disutility to ship through a spot market carrier. For each supplier i the cost is generated as $\pi_i = \frac{f_i(0.5\iota_i)}{\xi} r$, where a spot market carrier is assumed to be a fast option with 0.5 the baseline speed, and r is the 3PL's disutility factor of using a spot market carrier. The exact value of r is specified when generating data sets in Section 2.5.1.

Random customer data is obtained in a similar manner as supplier data. For customer $j \in J$, we generate due date κ_j uniformly in range [500,600]. Each customer has a number of outbound transportation options $l \in L_j$, with dispatch times $\tau_{jl} \in Y_j$ between [100,500]. The number of options and their dispatch times are specified in Section 2.5.1.

Outbound transportation cost is expressed as $g_j(\tau_{jl}) = \theta_j \rho_j(\tau_{jl}) \frac{C_{jl}}{\xi}$, where θ_j and ξ are generated in the same ranges as in supplier data. For $\rho_j(\tau_{jl})$, a baseline transportation rate $\bar{\rho}_j$ and a baseline transportation time ι_j are generated. The expressions are $\bar{\rho}_j = \delta_j \cdot \rho$, and $\iota_j = \delta_j/v$, where ρ and v are generated in the same uniform ranges as in supplier data. If the transportation time is shorter than the baseline, i.e., $\kappa_j - \tau_{jl} < \iota_j$, the transportation price $\rho_j(\tau_{jl})$ is calculated as $\rho_j(\tau_{jl}) = \bar{\rho}_j + 2 \cdot \bar{\rho}_j \cdot [1 - ((\kappa_j - \tau_{jl})/\iota_i)] + \bar{\rho}_i \cdot \epsilon$. However, if transportation time is longer than the baseline, i.e., $\kappa_j - \tau_{jl} \geq \iota_j$, the price equals $\rho_j(\tau_{jl}) = \bar{\rho}_j - 0.1 \cdot \bar{\rho}_j \cdot [1 - (\iota_j / (\kappa_j - \tau_{jl}))] + \bar{\rho}_j \cdot \epsilon$. Likewise, faster transportation options, i.e., ones that require less time to reach the customer from the time they depart the consolidation center, cost more than slower ones. The customer cost function for the *baseline* capacity ξ is plotted in Figure A.1b.

The cost of spot-market carrier shipping π_j, for each customer $j \in J$, is generated as $\pi_j = \frac{f_j(0.5\iota_j)}{\xi} r$, where a spot market carrier is assumed to be a fast option with 0.5 the baseline speed.



(a) Supplier cost function **(b)** Customer cost function

Figure A.1. Transportation options cost function for suppliers and customers.

A.2 Deterministic distribution planning with consolidation (DDPC)

The *deterministic distribution planning with consolidation - path based formulation* (DDPC-PF) is modeled below, using the same notation and decision variables defined in Section 2.3. We use the path formulation as opposed to the flow formulation, to avoid the nonlinearity in the objective function and since the number of paths is small when there is only a single scenario. In this model, the mean demand of each shipment (i, j), which we denote as $\bar{d}_{ij}$, is used instead of samples from the demand distribution.

$$[\textbf{DDPC-PF}] \ \min \sum_{i\in I}\sum_{q\in Q} f(x_{iq}) + \sum_{j\in J}\sum_{l\in L} g(y_{jl}) + \sum_{i\in I}\sum_{j\in J(i)}\sum_{p\in P_{ij}} c_p \beta_{ijp} \tag{A.1}$$

$$\text{subject to} \sum_{p\in P_{ij}} \beta_{ijp} = 1 \qquad \forall i \in I, j \in J(i) \tag{A.2}$$

$$\sum_{p\in P_{ij}} a_{iqp}\beta_{ijp} \le x_{iq} \qquad \forall i \in I, j \in J(i), q \in Q \tag{A.3}$$

$$\sum_{p\in P_{ij}} b_{jlp}\beta_{ijp} \le y_{jl} \qquad \forall j \in J, i \in I(j), l \in L \tag{A.4}$$

$$\sum_{p\in P_{ij}}\sum_{j\in J(i)} a_{iqp}\bar{d}_{ij}\beta_{ijp} \le C_{iq} \qquad \forall i \in I, q \in Q \tag{A.5}$$

$$\sum_{p\in P_{ij}}\sum_{i\in I(j)} b_{jlp}\bar{d}_{ij}\beta_{ijp} \le C_{jl} \qquad \forall j \in J, l \in L \tag{A.6}$$

$$x_{iq}, y_{jl} \in \{0, 1\}, \qquad i \in I, q \in Q, j \in J, l \in L$$

$$\beta_{ijp} \in \{0, 1\}, \qquad i \in I, j \in J(i), p \in P_{ij} \tag{A.7}$$

Similar to SDPC-PF, the objective function (A.1) minimizes the total transportation cost and the cost of allocating shipments to paths. Constraints (A.2) ensure that exactly one path is chosen for each shipment (i, j) in the network. Constraints (A.3) and (A.4) guarantee that shipment (i, j) traverses a path only if both the inbound transportation option q of supplier i and the outbound transportation option l of customer j are open. Constraints (A.5) and (A.6) ensure that the total demand that traverses a given path does not exceed the capacity of the inbound and outbound transportation options of that path. Finally, Constraints (A.7) impose the binary requirement on the variables.

A.3 Detailed SAA results example

In this section we provide and analyze the detailed result of one problem instance; instance 3 from set 9A, with 20 suppliers, 20 customers and 100 shipments. We exhibit the results of the 10 SAA runs in Table A.1. The first column shows the run number and the second shows the objective value of the run, which is the expected transportation and holding cost based on the 10 scenarios within the run as defined by objective function (2.22). This objective value is the sum of the first stage and the second stage costs, shown in columns 3 and 4, respectively. We evaluate each of those runs on a 1000-scenario tree and report the upper bound estimate and standard deviation in columns 5 and 6. We also report the *expected outsourcing* and *expected utilization* for each run in columns 7 and 8, respectively.

In Table A.2, we show statistics of the instance. The upper bound is the minimum of the upper bound estimates reported in column 5 of Table A.1, which is the upper bound of run 9 in this instance. We also report the lower bound and its standard deviation; the lower bound is the mean of the objectives in column 2 of Table A.1, minus $t_{\alpha=5,v=9}\hat{\sigma}_{N,M}$, as shown in Equation (2.25). The absolute gap of the SAA algorithm, i.e., the difference between upper and lower bounds, as well as the relative gap are also shown. We compute the relative gap as $\frac{UB-LB}{UB}$, since our goal is to evaluate the quality of the UB; the expected cost of the *true* problem. The cost values reported in columns 2 to 6 are all in units of 1000s of dollars.

We note that the SAA run with the optimal transportation plan, which is run 5 in this instance, resulted in relatively low *expected outsourcing* and *expected utilization* values, compared to other runs. This suggests that this plan reserved a higher level of capacity compared to plans of other SAA runs.

To further analyze the results, we provide a summary of the distribution plan of each SAA run in Table A.3. That table summarizes, for each run, the total number of inbound and outbound reserved options with different speed and capacity levels. The breakdown of the optimal transportation plan, run 5, is also plotted in Figure A.2. We note that the different runs result in somewhat similar transportation plans. This low variability in the solutions of the SAA runs indicates the stability of sampling among the different runs. We also notice that for some suppliers and customers, more than one transportation option is reserved. Run 5, for instance, has a total of 28 inbound reserved options for 20 suppliers, and 22 outbound options for 20 customers. Since this instance is from experimental setting A, with the high expected wait time at the consolidation center, the results suggest that in some cases, overbooking capacity is justifiable to reduce expected holding cost.

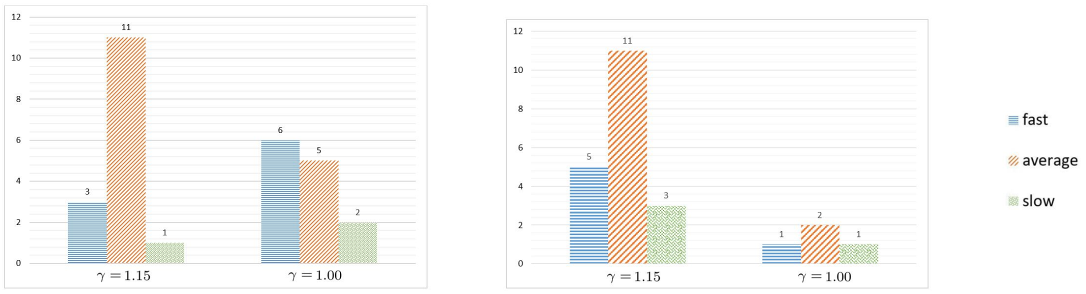
Table A.3 also shows that average-speed options are the most reserved, especially with higher capacity level. This is explained by the fact that the arrival and dispatch times for inbound and outbound options in this instance are independent, meaning that slow inbound options do not necessarily create feasible paths with fast outbound options, making reserving fast options with higher costs less justifiable.

Table A.1. Detailed SAA solution of Instance 3 - Set 9A.

SAA run	Objective	First stage cost	Second stage cost	Upper bound (UB)	UB standard deviation	Expected outsourcing (%)	Expected utilization (%)
1	515.087	463.319	51.767	525.381	0.344	1.51	72.30
2	520.095	470.902	49.193	523.708	0.233	1.22	70.38
3	517.050	467.325	49.725	525.101	0.283	1.41	71.52
4	518.735	469.991	48.744	521.468	0.171	1.21	71.95
5	519.759	473.242	46.516	**520.006**	0.039	1.03	71.85
6	518.214	471.784	46.430	522.902	0.171	1.25	70.24
7	516.361	461.241	55.119	531.877	0.444	1.85	72.21
8	519.626	472.740	46.886	522.398	0.102	1.10	72.94
9	516.499	470.524	45.975	522.383	0.216	1.58	76.76
10	521.323	473.636	47.687	520.549	0.041	0.92	72.79

Table A.2. SAA solution statistics of Instance 3 - Set 9A.

Upper bound	520.006
Std. dev. upper bound	0.039
Lower bound	517.126
Std. dev. lower bound	0.626
Absolute Gap	2.880
Relative Gap	0.55%



(a) Inbound reserved options **(b)** Outbound reserved options

Figure A.2. Breakdown of the transportation plan of the best SAA run (run 5).

Table A.3. Transportation plans for each SAA run, Instance 3 - Set 9A.

SAA run	Inbound Options						Outbound Options					
	fast		average		slow		fast		average		slow	
	$\gamma =$		$\gamma =$		$\gamma =$		$\gamma =$		$\gamma =$		$\gamma =$	
	1.00	1.15	1.00	1.15	1.00	1.15	1.00	1.15	1.00	1.15	1.00	1.15
1	6	3	5	11	2	1	3	3	3	10	0	3
2	6	3	6	10	2	1	1	5	2	11	1	3
3	6	3	5	11	1	1	1	5	3	10	1	3
4	7	2	5	11	2	1	1	5	2	11	0	3
5	**6**	**3**	**5**	**11**	**2**	**1**	**1**	**5**	**2**	**11**	**0**	**3**
6	7	2	5	11	2	1	1	5	2	11	1	3
7	7	2	6	10	2	1	2	4	3	10	0	3
8	5	1	6	13	1	1	1	6	2	11	0	2
9	7	4	3	9	1	0	0	3	2	12	1	4
10	5	1	5	14	1	1	1	6	2	11	0	2

Appendix B

Chapter 4 Appendix

B.1 Proofs

B.1.1 Proof of Proposition 1

Proposition 1. *For a given choice of heatmap h at t, and a given realization of exogenous information W_{t+1}, $v^{\pi}(S_{t+1}|S_t, x_t = \mathcal{X}_{tij}, y_t = h) \geq v^{\pi}(S_{t+1}|S_t, x_t = \mathcal{X}'_{tij}, y_t = h)$.*

Proof. We prove Proposition 1 by induction. Since $v^{\pi}(S_T) = 0$, for all S_T, we show that the result holds for $T-2$ and $T-1$. Let $\mathcal{D}_{lkij}$ be the set of active orders at epoch l with deadline at epoch $l+k$, $l \in \{t, \ldots, T\}, k \in \{l+\tau_{ij}, l+\omega\}$, $|\mathcal{D}_{lkij}| = d_{lkij}$. Suppose at $t = T-2$, we choose the set of orders to fulfill $\mathcal{X}'_{T-2ij} \subseteq \mathcal{D}_{(T-2)Tij}$ and assume that $\mathcal{D}_{(T-2)(T-1)ij} \neq \emptyset$ (without loss of generality, we assume that $\tau_{ij} = 1$). Since the choice of heatmap h and the number of fulfilled orders for each (i,j) pair, x_{tij}, is fixed and does not affect the set $\mathcal{X}_{tij}$, the number and locations of drivers at $t = T-1$, m_{T-1}, is independent of the choice of subset of orders to match. So m_{T-1} is constant for a given realization of the binomial random variable of repositioning, m^u_{T-1}, regardless of the elements of the set $\mathcal{X}'_{T-2ij}$. Since $\mathcal{X}'_{T-2ij}$ is a subset of $\mathcal{D}_{(T-2)Tij}$, the carried forward demand at $T-1$ is $d_{(T-1)Tij} = d_{(T-2)Tij} - x_{T-2ij}$, $\forall i,j \in N$. If for every $i \in N$, $m_{T-1i} \leq \sum_{j \in N} d_{(T-1)Tij}$, then the maximum number of possible matches at node i is m_{T-1i}. Therefore,

$v^{\pi}(S_{T-1}|S_{T-2}, x_{T-2} = \mathcal{X}_{T-2ij}, y_{T-2} = h) = v^{\pi}(S_{T-1}|S_{T-2}, x_{T-2} = \mathcal{X}'_{T-2ij}, y_{T-2} = h)$, and any arbitrary choice of the set $\mathcal{X}'_{T-2ij}$ would result in the maximum possible reward at $t = T-1$. However, if for a given node $i \in N$, $m_{T-1i} > \sum\limits_{j \in N} d_{(T-1)Tij}$, then supply at node i exceeds demand, and a higher number of matches could have been obtained if $\mathcal{X}'_{T-2ij}$ consisted of orders from the set $\mathcal{D}_{(T-2)(T-1)ij}$, since more orders from the set $\mathcal{D}_{(T-2)Tij}$ would be carried forward to $T-1$, resulting in a similar matching at $t = T-2$, but a higher matching at $t = T-1$. Therefore, $v^{\pi}(S_{T-1}|S_{T-2}, x_{T-2} = \mathcal{X}_{T-2ij}, y_{T-2} = h) > v^{\pi}(S_{T-1}|S_{T-2}, x_{T-2} = \mathcal{X}'_{T-2ij}, y_{T-2} = h)$.

We have now shown that the result of Proposition 1 holds for $T-2$ and $T-1$. By the induction assumption, assume that the result holds for $t = 2, \ldots, T-3$. Next, we show that the result holds for $t = 1$. We choose $\mathcal{X}'_{1ij} \subseteq \mathcal{D}_{13ij}$ and we assume that $\mathcal{D}_{12ij} \neq \emptyset$. As noted earlier, m_{2i} is independent of the choice of the elements of the set $\mathcal{X}'_{1ij}$. Since we set $\mathcal{X}'_{1ij}$ to be a subset of $\mathcal{D}_{13ij}$, $d_{23ij} = d_{13ij} - x_{1ij}$, $\forall i, j \in N$. If for every $i \in N$ $m_{2i} \leq \sum\limits_{j \in N} (d_{23ij} + \sum_{k=4}^{2+\omega} d_{2kij})$, then the maximum number of possible matches at $t = 2$, for every $i \in N$, is m_{2i}. Therefore, $v^{\pi}(S_2|S_1, x_1 = \mathcal{X}_{1ij}, y_1 = h) = v^{\pi}(S_2|S_1, x_1 = \mathcal{X}'_{1ij}, y_1 = h)$, and any arbitrary choice of the set $\mathcal{X}'_{1ij}$ would result in the maximum possible reward at $t = 1$.

However, if for a given $i \in N$, $m_{2i} > \sum\limits_{j \in N} (d_{23ij} + \sum_{k=4}^{2+\omega} d_{2kij})$, then supply exceeds demand at node i, and a higher number of matches at $t = 2$ could have been obtained if $\mathcal{X}'_{1ij}$ contained elements from the set $\mathcal{D}_{1,2ij}$, since more orders could have been carried forward to $t = 2$, resulting in a similar matching at $t = 1$, but a higher number of matches at $t = 2$. So, $v^{\pi}(S_2|S_1, x_1 = \mathcal{X}_{1ij}, y_1 = h) > v^{\pi}(S_2|S_1, x_1 = \mathcal{X}'_{1ij}, y_1 = h)$, which concludes the proof. □

B.1.2 Proof of Lemma 1

Lemma 1. *MCDRP is a relaxation of MHSP.*

Proof. Proof. To show that MCDRP is a relaxation of MHSP, we start by showing that a feasible solution to Model (B.1) is also feasible to Model (4.2), but the opposite is not true.

Notice that Constraints (4.2b - 4.2d) and (4.3b - 4.3g) are shared in the first and second stage problems of (B.1) and (4.2). Thus, it suffices to show that Constraints (B.1b - B.1g) and (B.2b

- B.2i) give a feasible solution with respect to Constraints (4.2e - 4.2g) and (4.3h - 4.3k). For ease of exposition, we drop the s subscript for the second stage variables, so that, for example, y_{lh} refers to both first stage y_{lh} and second stage y^s_{lh} variables. Recall that a particular choice of heatmap, denoted by variable y_{lh}, determines the transition probability $P_{ij}(h)$, which in turn determines the number of drivers that reposition from i to j, denoted by variables u_{lijh} and u'_{li}. Since the number of drivers that reposition (u_{lijh} and u'_{li}) in (B.1) is guaranteed to add up to the number of unmatched drivers at i because of Constraints (B.1d) and (B.2e-B.2f), it follows that the value of $\sum_{h\in H}\sum_{j:(i,j)\in A} u_{ljih} + u'_{li}$ equals $\sum_{j:(i,j\in A)} u_{lij}$ because of (4.2e) and (4.3h-4.3i). Thus, any feasible values of u_{ljih} and u_{li} satisfy constraints (4.2e, 4.3h-4.3i) and (4.2f, 4.3j), and therefore, any solution of (B.1) is feasible with respect to (4.2).

Next, we show that the opposite is not true; not all solutions of (4.2) are feasible for (B.1). Recall that $\sum_{h\in H} u_{lijh}$ determines the number of drivers that reposition from i to j, for all $(i,j)\in A$. This is equivalent to u_{lij} in model (4.2). We can easily see that any value of u_{lij} that meets the following two conditions (a) $u_{lij} > (m_{li} - \sum_{j\in N} x_{lij})P_{ij}(h)$ and (b) $u_{lij} \leq (m_{li} - \sum_{j\in N} x_{lij})$ is feasible to (4.2). That is because such a value meets Constraints (4.2e, 4.3h-4.3i), but is infeasible to (B.1) since it violates Constraints (B.1c, B.2c-B.2d). So model (B.1) is more constrained than model (4.2), and thus (4.2) is a relaxation of (B.1). □

B.2 Additional Formulations

B.2.1 Matching and Heatmap Selection Problem (MHSP)

To formulate the problem, we first define the following additional notation. Binary variables y_{lh}, y^s_{lh} equal 1 if heatmap $h \in H$ is chosen at epoch l, and 0 otherwise, for first and second stage problems, respectively. u_{lijh}, u^s_{lijh} are the number of drivers that transition from node i to node j, between decision epochs l and $l+1$, given heatmap h, in the first and second stages. u'_{li}, u'^s_{li} is the number of drivers that stay at node i at decision epoch l in the first and second stages. This is needed to ensure the flow balance after transition, i.e., the total number of drivers stays the same when transitioning between l and $l+1$. M is a very large real number.

The problem is formulated as follows.

[MHSP]

$$\max \sum_{l=t}^{t+\Delta} \sum_{i\in N} \sum_{j\in N} x_{lij} + \frac{1}{|\mathbb{S}|} \sum_{s\in\mathbb{S}} Q^s(m, d, x^s, y^s, m^s, u^s, d^s) \tag{B.1a}$$

s.t. (4.2b – 4.2d)

$$\sum_{h\in H} y_{lh} = 1 \qquad l \in \{t, \ldots, t+\Delta\} \tag{B.1b}$$

$$u_{lijh} \le (m_{li} - \sum_{j\in N} x_{lij}) P_{ij}(h) \qquad \forall (i,j) \in A, h \in H, l \in \{t, \ldots, t+\Delta\} \tag{B.1c}$$

$$u'_{li} = (m_{ti} - \sum_{j\in N} x_{lij}) - \sum_{h\in H} \sum_{j:(i,j)\in A} u_{lijh} \qquad \forall i \in N, l \in \{t, \ldots, t+\Delta\} \tag{B.1d}$$

$$u_{lijh} \le M y_{lh} \qquad \forall (i,j) \in A, h \in H, l \in \{t, \ldots, t+\Delta\} \tag{B.1e}$$

$$m_{li} = \sum_{j\in N} x_{(l-\tau_{ji})ji} + \sum_{h\in H} \sum_{j:(j,i)\in A} u_{(l-1)jih} + u'_{(l-1)i} \qquad \forall i \in N, l \in \{t+1, \ldots, t+\Delta+1\} \tag{B.1f}$$

$$x_{lij}, u'_{li} \in \mathbb{Z}_{\ge 0} \qquad \forall i,j \in N, l \in \{t, \ldots, t+\Delta\}$$

$$u_{lijh} \in \mathbb{Z}_{\ge 0}, y_{lh} \in \{0,1\} \qquad \forall (i,j) \in A, h \in H, l \in \{t, \ldots, t+\Delta\}$$

$$m_{li} \in \mathbb{Z}_{\ge 0} \qquad \forall i \in N, l \in \{t+1, .., t+\Delta+1\}$$

$$d_{lkij} \ge 0 \qquad \forall (i,j) \in A, l \in \{t+1, .., t+\Delta+1\}, \; k \in \{l+\tau_{ij}, .., l+\omega-1\} \tag{B.1g}$$

where the second-stage cost function is defined as

$$Q^s(m, d, x^s, y^s, m^s, u^s, d^s) = \sum_{l=t+\Delta+1}^{\Gamma} \sum_{i\in N} \sum_{j\in N} \alpha^{(l-t)} x^s_{lij} \tag{B.2a}$$

s.t. ($4.3b - 4.3g$)

$$\sum_{h\in H} y^s_{lh} = 1 \qquad \forall l \in \{t+\Delta+1, \ldots, t+\Gamma\} \tag{B.2b}$$

$$u^s_{lijh} \leq (m_{li} - \sum_{j\in N} x^s_{lij}) P_{ij}(h) \qquad \forall i, j \in N, l \in \{t+\Delta+1\}, h \in H \tag{B.2c}$$

$$u^s_{lijh} \leq (m^s_{li} - \sum_{j\in N} x^s_{lij}) P_{ij}(h) \qquad \forall i, j \in N, l \in \{t+\Delta+2, \ldots, t+\Gamma\}, h \in H \tag{B.2d}$$

$$u'^s_{li} = (m_{li} - \sum_{j\in N} x^s_{lij}) - \sum_{h\in H} \sum_{j:(i,j)\in A} u^s_{lijh} \qquad \forall i \in N, l \in \{t+\Delta+1\} \tag{B.2e}$$

$$u'^s_{li} = (m^s_{li} - \sum_{j\in N} x^s_{lij}) - \sum_{h\in H} \sum_{j:(i,j)\in A} u^s_{lijh} \qquad \forall i \in N, l \in \{t+\Delta+2, \ldots, t+\Gamma\} \tag{B.2f}$$

$$u^s_{lijh} \leq M y^s_{lh} \qquad \forall (i,j) \in A, h \in H, l \in \{t+\Delta+1, \ldots, t+\Gamma\} \tag{B.2g}$$

$$m^s_{li} = \sum_{j\in N} x^s_{(l-\tau_{ji})ji} + \sum_{h\in H} \sum_{j:(i,j)\in A} u^s_{(l-1)jih} + u'^s_{(l-1)i} \qquad \forall i \in N, l \in \{t+\Delta+2, \ldots, t+\Gamma\} \tag{B.2h}$$

$$x^s_{lij}, u'^s_{li} \in \mathbb{Z}_{\geq 0} \qquad \forall i, j \in N, l \in \{t+\Delta+1, \ldots, t+\Gamma\}$$

$$u^s_{lijh} \in \mathbb{Z}_{\geq 0}, \qquad \forall (i,j) \in A, h \in H, l \in \{t+\Delta+1, \ldots, t+\Gamma\}$$

$$m^s_{li} \in \mathbb{Z}_{\geq 0} \qquad \forall i \in N, l \in \{t+\Delta+2, .., t+\Gamma\}$$

$$d^s_{lkij} \geq 0 \qquad \forall (i,j) \in A, l \in \{t+\Delta+2, .., t+\Gamma\}, \; k \in \{l+\tau_{ij}, .., l+\omega-1\} \tag{B.2i}$$

Similar to MCDRP, the first and second stages are linked by the number of drivers and carried forward demand at epoch $t+\Delta+1$, reflected here in the additional constraints (B.2c and B.2e). Objective function (B.1a) maximizes the number of matches of active orders at decision epochs t to $t+\Delta$, plus the expectation of matched forecasted orders for decision epochs $\{t+\Delta+1, \ldots, t+\Gamma\}$. Constraints (B.1b) and (B.2b) guarantee that a single heatmap is selected for epoch t and for each subsequent epoch l. Constraints (B.1c) and (B.2c-B.2d) limit the number of drivers that transition from nodes i to j, given heatmap h, to the unmatched drivers at epoch t multiplied by the transition probability $P_{ij}(h)$. Since the right-hand-side of this equation can be fractional and the number of drivers on the left-hand-side is rounded down, we add Constraints (B.1d) and (B.2e-B.2f) to ensure the flow balance between the number of unmatched drivers at epoch l and the number of available drivers at $l+1$. We assume that if the number of unmatched drivers at l is not equal to the total number of drivers that transition to other nodes δ_{lijh} due to the rounding error in equations (B.1c) and (B.2c-B.2d), the difference (which cannot exceed 1) stays at the origin node

i. We relax this assumption through the proposed solution methodology discussed in Section 4.4.

Constraints (B.1e) and (B.2g) ensure that drivers move according to transition probability matrix $P(h)$ only if heatmap h is chosen. Constraints (B.1f) and (B.2h) compute the number of drivers at decision epoch $l+1$ for each node. Constraints (B.1g) - (B.2i) assign only integer values to all variables except y_{lh}, y^s_{lh}, which are binary, and d_{lkij}, d^s_{lkij}, which are continuous.

B.2.2 Matching with No Driver Relocation (MNDRP) - a Lower Bound

In this section, we simplify MHSP by considering only matching decisions, without heatmaps to influence the repositioning decisions of drivers. We assume that if drivers are not matched, they simply stay at their origin node. The goal is to maximize the number of fulfilled orders through matching drivers with orders in a way that considers their availability in subsequent time periods. We use this model as a benchmark to compare the effectiveness of heatmaps against. That is, this model enables us to quantify the additional matching that we're able to achieve when heatmaps are used, compared to when balancing supply and demand is done only through matching decisions.

[MNDRP]

$$\max \sum_{l=t}^{t+\Delta} \sum_{i \in N} \sum_{j \in N} x_{lij} + \frac{1}{|\mathbb{S}|} \sum_{s \in \mathbb{S}} Q^s(x, d, x^s, m^s, d^s) \tag{B.3a}$$

s.t. $(4.2b - 4.2d)$

$$m_{li} = \sum_{j \in N} x_{(l-\tau_{ji})ji} + (m_{(l-1)i} - \sum_{j \in N} x_{(l-1)ij}) \qquad \forall i \in N, l \in \{t+1, \ldots, t+\Delta\} \tag{B.3b}$$

$$x_{lij} \in \mathbb{Z}_{\geq 0} \qquad \forall i, j \in N, l \in \{t+1, \ldots, t+\Delta\}$$

$$m_{li} \in \mathbb{Z}_{\geq 0}, d_{lkij} \geq 0 \qquad \forall i \in N, l \in \{t+1, \ldots, t+\Delta+1\}$$

$$k \in \{l+\tau_{ij}, \ldots, l+\omega-1\} \tag{B.3c}$$

where

$$Q^s(x,d,x^s,m^s,d^s) = \sum_{l=t+\Delta+1}^{\Gamma} \sum_{i\in N} \sum_{j\in N} \alpha^{(l-t)} x^s_{lij} \tag{B.4a}$$

s.t. ($4.3b - 4.3g$)

$$m^s_{li} = \sum_{j\in N} x^s_{(l-\tau_{ji})ji} + (m_{(l-1)i} - \sum_{j\in N} x^s_{(l-1)ij}) \qquad \forall i \in N, l \in \{t+\Delta+2\} \tag{B.4b}$$

$$m^s_{li} = \sum_{j\in N} x^s_{(l-\tau_{ji})ji} + (m^s_{(l-1)i} - \sum_{j\in N} x^s_{(l-1)ij}) \qquad \forall i \in N, l \in \{t+\Delta+3,\dots,t+\Gamma\} \tag{B.4c}$$

$$x^s_{lij} \in \mathbb{Z}_{\geq 0} \qquad \forall i,j \in N, l \in \{t+\Delta+1,\dots,t+\Gamma\}$$

$$m^s_{li} \in \mathbb{Z}_{\geq 0}, d^s_{lkij} \geq 0 \qquad \forall i,j \in N, l \in \{t+\Delta+2,\dots,t+\Gamma\}, \; k \in \{l+\tau_{ij},\dots,l+\omega-1\} \tag{B.4d}$$

Constraints (B.3b) and (B.4b-B.4c) compute the number of drivers at subsequent decision epochs, where unmatched drivers do not relocate, but rather stay in their origin node. The following lemma demonstrates the relationship between MNDRP and MHSP.

Lemma 3. MNDRP is a restriction of MHSP, if there exists a heatmap $h \in H$, such that $\forall i \in N$, $p_{ii}(h) = 1$.

The proof is similar to that of Lemma 1; any solution of model (B.3) is feasible with respect to model (B.1), but the opposite is not true.

B.3 Repositioning Probabilities

In the computational experiments, we set the pairwise repositioning probabilities as expressed in (B.5).

$$\rho_{\epsilon_i\epsilon_j} = \begin{cases} 0.9, \text{ if } \epsilon_j = 2 \text{ and } \epsilon_i = 0 \\ 0.4, \text{ if } \epsilon_j = 2 \text{ and } \epsilon_i = 1 \\ 0.55, \text{ if } \epsilon_j = 1 \text{ and } \epsilon_i = 0 \\ 0.2, \text{ if } \epsilon_j = 2 \text{ and } \epsilon_i = 2 \\ 0.1, \text{ if } \epsilon_j = 1 \text{ and } \epsilon_i = 1 \\ 0, \text{ otherwise} \end{cases} \quad (i,j) \in A. \tag{B.5}$$

We define the repositioning probability between nodes $(i, j) \in A$ as shown in (4.6). The number of drivers that reposition from any node $i \in N$ is a multinomial random variable, with m^u_{ti} trials and probabilities of success as expressed in equations (4.6).

We now extend the definition of repositioning probabilities to case 2, where drivers can exit the system and new drivers may join. We define the pairwise probabilities for a pair of nodes (i, j) as shown in (B.6). Recall that node 0 is the auxiliary node where drivers enter/exit the system. For each pair of nodes (i, j), there are 3 associated probabilities: $\rho_{\epsilon_i\epsilon_j}$: the probability of moving from i to j given the heat levels of each node, $\rho_{(\epsilon_i\epsilon_j,i)}$: the probability of staying at node i given the heat levels of nodes i and j, and $\rho_{(\epsilon_i\epsilon_j,0)}$: the probability of exiting the system given heat levels ϵ_i and ϵ_j. Thus, for each pair of nodes (i, j) we define a vector $\boldsymbol{\rho_{\epsilon_i\epsilon_j}} = \begin{bmatrix}\rho_{\epsilon_i\epsilon_j} & \rho_{\epsilon_i\epsilon_j,i} & \rho_{\epsilon_i\epsilon_j,0}\end{bmatrix}$. In the computational testing, we use the following values of $\boldsymbol{\rho_{\epsilon_i\epsilon_j}}$ for each pair of heat levels.

$$\boldsymbol{\rho_{\epsilon_i \epsilon_j}} = \begin{cases} \begin{bmatrix} 0.0 & 0.1 & 0.9 \end{bmatrix} & \text{if } \epsilon_j = 0 \text{ and } \epsilon_i = 0 \\ \begin{bmatrix} 0.0 & 0.5 & 0.5 \end{bmatrix} & \text{if } \epsilon_j = 0 \text{ and } \epsilon_i = 1 \\ \begin{bmatrix} 0.0 & 0.8 & 0.2 \end{bmatrix} & \text{if } \epsilon_j = 0 \text{ and } \epsilon_i = 2 \\ \begin{bmatrix} 0.3 & 0.1 & 0.6 \end{bmatrix} & \text{if } \epsilon_j = 1 \text{ and } \epsilon_i = 0 \\ \begin{bmatrix} 0.1 & 0.5 & 0.4 \end{bmatrix} & \text{if } \epsilon_j = 1 \text{ and } \epsilon_i = 1 \\ \begin{bmatrix} 0.0 & 0.8 & 0.2 \end{bmatrix} & \text{if } \epsilon_j = 1 \text{ and } \epsilon_i = 2 \\ \begin{bmatrix} 0.5 & 0.1 & 0.4 \end{bmatrix} & \text{if } \epsilon_j = 2 \text{ and } \epsilon_i = 0 \\ \begin{bmatrix} 0.3 & 0.4 & 0.3 \end{bmatrix} & \text{if } \epsilon_j = 2 \text{ and } \epsilon_i = 1 \\ \begin{bmatrix} 0.1 & 0.7 & 0.2 \end{bmatrix} & \text{if } \epsilon_j = 2 \text{ and } \epsilon_i = 2 \end{cases} \quad (i,j) \in A, i,j \neq 0. \tag{B.6}$$

We also define ρ_{0,ϵ_i} as the probability of entering the system from the auxiliary node, given the heat level of node i. The following are the values set in the testing:

$$\rho_{0,\epsilon_i} = \begin{cases} 0.5, & \text{if } \epsilon_i = 2 \\ 0.3, & \text{if } \epsilon_i = 1 \\ 0, & \text{if } \epsilon_i = 0 \end{cases} \tag{B.7}$$

The repositioning probabilities for the whole network are then obtained by equations (4.6).

B.4 Additional Computational Results

Tables B.1 and B.2 report the solution summary of different test settings, solution policies, and supply levels, for cases 1 and 2, respectively. Each cell is the average of 10 problem instances with varying demand realizations, for the complete planning horizon T. We report on two main statistics: (1) percentage demand fulfillment throughout the planning horizon, which quantifies the expected service level under each policy, and (2) percentage of matched drivers in the network, the complement of which indicates the percentage of drivers that may relocate as a response to a heatmap. That is, when a high number of drivers is matched, only a small percentage can relocate as a response to a heatmap. The test settings we use are (1) the original dataset as described in Section 4.5.1, (2) demand data scaled down by a factor of 4, (3)

Table B.1. Solution statistics for different testing setting, varying computational policies and levels of initial driver supply, **case 1** (no enter/exit).

Supply (% of Avg Demand)	**Policy**	**Fulfilled demand (%)**				**Matched drivers (%)**			
		Original Data	Scaled-down Demand	Imbalanced Network	P-sensitivity	Original Data	Scaled-down Demand	Imbalanced Network	P-sensitivity
80	MCDRP	78.65	78.95	57.84	57.88	96.21	94.85	86.5	86.86
	SLA	78.09	78.44	56.79	54.2	93.09	91.82	82.93	81.58
	simple	77.45	77.69	50.93	49.96	89.67	88.48	76.58	75.79
	MNDRP	77.13	77.36	48.16	48.3	88.34	87.36	73.84	74.61
90	MCDRP	86.32	85.82	62.57	62.66	95.98	93.37	82.67	83.46
	SLA	85.89	85.27	60.72	57.12	92.92	90.04	77.38	75.8
	simple	85.27	84.39	52.08	50.68	89.92	87.12	69.29	69.54
	MNDRP	84.73	84.07	48.36	48.48	87.86	86.17	66.89	68.57
100	MCDRP	94.29	91.09	67.75	67.79	96.21	90.74	80.39	81.2
	SLA	90.8	90.31	64.87	60.18	89.97	86.72	72.88	71.21
	simple	90.19	88.74	53.2	51.44	87.85	82.29	65.49	66.38
	MNDRP	89.26	87.62	48.51	48.6	84.76	79.88	62.61	65.5
120	MCDRP	98.62	98.22	77.33	77.39	87.03	83	77.66	78.31
	SLA	97.51	97.1	73.14	66.13	77.76	75.47	68.21	66.34
	simple	95.06	93.96	55.97	53.23	76.83	71.85	58.64	59.45
	MNDRP	92.3	91.2	48.77	48.74	72.69	68.16	57.09	59.6
150	MCDRP	99.06	99.06	91.3	91.28	61.24	59.69	71.63	72.24
	SLA	99.02	98.78	83.44	74	57.56	56.27	57.62	57.3
	simple	97.82	96.76	58.47	55.01	59.9	57.43	51.43	51.7
	MNDRP	93.03	92.12	49.1	49.13	56.33	56.12	48.29	50.55

imbalanced network described in Section 4.5.4 where we reduce demand inflow of two nodes by a factor of 10, while keeping the rest of the demand distributions the same, and (4) sensitivity on heatmap transition probabilities as detailed in Section 4.5.5.

B.5 Notation Summary

Table B.2. Solution statistics for different testing setting, varying computational policies and levels of initial driver supply, **case 2** (enter/exit).

Supply (% of Avg Demand)	**Policy**	**Fulfilled demand (%)**				**Matched drivers (%)**			
		Original Data	Scaled-down Demand	Imbalanced Network	P-sensitivity	Original Data	Scaled-down Demand	Imbalanced Network	P-sensitivity
80	MCDRP	94.04	93.5	70.21	70.24	94.41	90.62	80.51	81
	SLA	92.3	91.3	62.5	62.38	94.46	91.3	83.85	80.91
	simple	91.4	89.5	57.04	56.57	92.7	89.85	76.29	72.11
	MNDRP	77.13	77.36	48.16	48.3	88.34	87.36	73.84	74.61
90	MCDRP	97.66	97.15	75	75.02	91.15	87.02	78.81	79.37
	SLA	95.8	94.86	65.6	65.47	92.41	89	81.73	78.63
	simple	93.41	91.79	58.86	58.2	87.38	85.36	73.03	68.34
	MNDRP	84.73	84.07	48.36	48.48	87.86	86.17	66.89	68.57
100	MCDRP	98.98	98.83	80.1	80.14	84.21	80.47	77.44	78.08
	SLA	98.27	97.35	68.9	68.78	89.1	85.34	80.47	77.49
	simple	94.55	93.33	60.71	60.29	82.24	79.86	70.22	64.94
	MNDRP	89.26	87.62	48.51	48.6	84.76	79.88	62.61	65.5
120	MCDRP	99.06	99.06	89.27	89.24	68.6	65.8	74.21	74.71
	SLA	98.98	98.73	75.24	74.97	75.88	73.67	78.35	75.07
	simple	96.06	94.88	64.5	63.88	75.75	72.57	65.81	60.36
	MNDRP	92.3	91.2	48.77	48.74	72.69	68.16	57.09	59.6
150	MCDRP	99.06	99.07	98.07	98.01	52.86	51.8	66.22	66.73
	SLA	99.05	99.03	84.58	84.26	60.95	59.88	73.9	70.52
	simple	97.33	96.37	69.88	69.54	69.89	66.94	60.92	55.44
	MNDRP	93.03	92.12	49.1	49.13	56.33	56.12	48.29	50.55

Table B.3. Notation Summary.

Notation	Meaning	Notation	Meaning
MDP Notation			
Notation	**Meaning**	**Notation**	**Meaning**
N	set of nodes in the network	A	set of arcs connecting adjacent nodes
t	time index (decision epoch) of planning horizon	T	the length of the planning horizon
$\mathcal{M}_t$	set of drivers at epoch t	$\mathcal{D}_t$	set of orders at epoch t
$\mathcal{D}_{tij}$	set of orders with o-d pair (i, j), ordered in increasing order of delivery deadline	m_{ti}	number of drivers at location i, epoch t
d_{tlij}	number of orders with o-d pair (i, j) at epoch t with delivery deadline at l	τ_{ij}	travel time (in epochs) between nodes i and j
ω	delivery time window (in epochs) of order fulfillment	γ	maximum heat level
h	heatmap	H	set of all heatmaps
$P(h)$	Transition probability matrix given heatmap h	p_{ij}	probability of transitioning from node i to j
S_t	state of the system at epoch t	$\boldsymbol{x_t}$	matching decision vector at epoch t
$\boldsymbol{y_t}$	heatmap decision vector at epoch t	m^u_{ti}	number of unmatched drivers at node i
$X^\pi(S_t)$	decision function for mapping state to action	π	MDP policy
W_t	random information revealed at epoch t	$\mathcal{X}_{tij}$	set of fulfilled orders at t with o-d pair (i, j)
$C(S_t, (x_t, y_t))$	contribution (reward) of making decision vectors x_t, y_t when in state S_t	$v^\pi(S_0)$	value function of state S_0 when following policy π
Solution Methodology Notation			
N_i	set of nodes adjacent to i	Γ	duration of forecast window
l	time index within forecast window $l \in \{t+1, \ldots, t+\Gamma\}$	$\mathbb{S}$	set of random samples of demand considered in SLA policy
d^s_{lkij}	number of forecasted orders with o-d pair (i, j) at epoch l with delivery deadline at $l + k$	Δ	service time duration for preparing order
x_{lij}	decision variable: number of matched active orders with o-d pair (i, j)	x^s_{lij}	decision variable: number of matched *forecasted* orders with o-d pair (i, j)
m_{li}, m^s_{li}	decision variable: number of drivers at i in epoch l, for fist and second stages	u_{lij}	decision variable: number of drivers transitioning from i in epoch l
α^{l-t}	discount parameter	$Q^s(\cdot)$	objective function value of second stage given scenario s
y_h	binary decision variable for choice of heatmap	δ_{ti}	shortage of drivers at node i, epoch $t+1$ given known demand
ϵ_i	heat level at node i	$\rho_{\epsilon_i\epsilon_j}$	pairwise probability of transitioning from a node with heat level ϵ_i to an adjacent node with heat level ϵ_j

Appendix C

Chapter 5 Appendix

C.1 Notation Summary

Table C.1. Notation Summary 1.

Notation	**Meaning**	**Notation**	**Meaning**
λ	order arrival rate per epoch	μ	driver arrival rate per epoch
W	guarantee (utilization or wage based)	τ	activity time window in hours
t	time index (decision epoch) of planning horizon	T	the length of the planning horizon
S_t	state of the system at epoch t	a	driver attribute vector
s_a	binary indicator, driver active	m_a	binary indicator, driver available
o_a	location of driver	h_a	active time up to t
y_a	progress towards desired guarantee (utilization or earning amount)	$\mathcal{A}_t$	set of attributes of available drivers at epoch t
$\mathcal{A}'_t$	set of attributes of en-route drivers at epoch t	b	demand attribute vector
$(o_b,\, d_b)$	origin and destination of order	$[t_b^{min}, t_b^{max}]$	time window of order fulfillment
φ_b	fraction of order value that platform earns	$\mathcal{B}_t$	set of all demand attributes at t
R_{ta}	number of drivers with attribute a at t	R_t	$R_t = (R_{ta})_{a \in \mathcal{A}_t}$ vector of drivers
D_{tb}	number of orders with attribute b at t	D_t	$D_t = (D_{tb})_{b \in \mathcal{B}_t}$ vector of demand
x_{tab}, x_{ta0}	number of drivers with attribute a assigned to orders with attribute b, and not assigned	$\mathcal{B}^+$	$\mathcal{B}^+ = \mathcal{B}_t \cup \{0\}$
$\mathcal{D}_{ab}$	set of time feasible assignments between driver a and order b	$R(b)$	revenue as a function of demand attribute b
$C(a,b)$	cost, function of demand and driver attributes	$F(a)$	penalty term as a function of driver attribute
$L(a,b)$	priority score in parametric cost function approximation	η	scalar that converts distance to time estimate
ι_{ab}	time it takes driver a to fulfill order b	w_b	distance between the origin and destination of an order b
θ_b	revenue per unit distance	w_{ab}	distance between the origin of a driver a and the origin of an order b
$\bar{\theta}_{ab}$	per unit distance delivery cost	ζ	penalty coefficient
ϕ_b	fraction of value of the order the platform collects	φ_b	fixed cost per delivery
ρ_a	driver matching penalty function	ρ'_b	demand loss penalty function

Table C.2. Notation Summary 2.

Notation	**Meaning**	**Notation**	**Meaning**
$V_t(S_t)$	the value of being in state S_t	$\mathcal{X}_t$	the feasible region at t
$C_t(S_t, x_t)$	myopic objective value of taking action vector x_t when in state S_t	W_t	exogenous information arriving between $t-1$ and t.
N	number of iterations in VFA algorithm	ω^n	sample path at iteration n
$\bar{V}_t$	approximate value function	$\bar{v}_{ta'}$	value function coefficient of having a driver with attribute a' at t
$\bar{\iota}(t, a, d)$	travel time of driver a when decision d is applied at t	$v^n_{t,a}$	new value function estimate at iteration n obtained from dual variables
γ	discount factor in MDP objective	ψ	rate of drivers exiting the system
Σ	number of zones in the model	$\mathcal{G}$	set of aggregation levels
σ^g	number of zones in aggregate level g	$w^{(g)}_{ta}$	weight of aggregate level g
$\bar{\beta}^{(g,n)}_{ta}$	bias estimate due to smoothing	$\bar{\mu}^{(g,n)}_{ta}$	bias estimate due to aggregation error
$\bar{\bar{\beta}}^{(g,n)}_{ta}$	total squared variation estimate	η_n	step size for updating bias and squared variation estimates
α^n_{ta}	step size for updating value function estimate	s^2_{ta}	variance estimate