Approximate Dynamic Programming for Crowd-shipping with In-store Customers

Kianoush Mousavi
Department of Civil and Mineral Engineering, University of Toronto, Toronto, Ontario M5S 1A4, Canada,
kianoush.mousavichashmi@mail.utoronto.ca

Merve Bodur
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada,
bodur@mie.utoronto.ca

Mucahit Cevik
Department of Mechanical and Industrial Engineering, Ryerson University, Toronto, Ontario M5B 2K3, Canada,
mcevik@ryerson.ca

Matthew J. Roorda
Department of Civil and Mineral Engineering, University of Toronto, Toronto, Ontario M5S 1A4, Canada,
matt.roorda@utoronto.ca

September 27, 2021

Crowd-shipping has gained significant attention as a last-mile delivery option over the recent years. However, crowd-shipping operations are subject to a high degree of uncertainty due to stochastic arrival of online orders and availability of crowd-shippers. In this study, we propose a variant of dynamic crowd-shipping model with in-store customers as crowd-shippers to deliver online orders within few hours. We formulate the problem as a Markov decision process and develop an approximate dynamic programming (ADP) policy using value function approximation for obtaining a highly scalable and real-time decision making strategy on matching orders to crowd-shippers while considering temporal and spatial uncertainty in arrival of online orders and crowd-shipper. We consider several algorithmic enhancements to the ADP algorithm such as employing hierarchical aggregation and imposing the monotonicity of the value functions, which significantly improve the convergence. We also propose an optimization-based myopic policy and compare it with the ADP policy using various performance measures including operational cost, percentage of served orders and average order postponement. Our numerical analysis with varying parameter settings show that ADP policies can lead to up 25.2% cost savings and 9.8% increase in the number of served orders. Accordingly, our findings demonstrate the viability of ADP for addressing the real-time decision making aspect of the dynamic crowd-shipping problem.

Key words: Crowd-shipping; Last-mile delivery; Markov decision process; Approximate dynamic programming; Value function approximation

1. Introduction

Online shopping has grown rapidly in the last few years, and this growth has accelerated with Covid-19 restrictions across the world. New data from IBM retail shows that the Covid-19 pandemic has pushed forward the e-commerce trends by five years (World Economic Forum 2021). Online shoppers order from anywhere and at any time of the day, often expecting their deliveries at their doors within a few hours. This has resulted in an increasing demand for quick and cost-efficient last-mile delivery operation. This observation is also evident from a survey of logistic providers, in which 78% of the respondents expect to offer same-day delivery within five years (Zebra 2019).

Crowd-shipping, in which individuals offer their excess capacity and time for making deliveries, is playing a growing role in delivering goods and groceries to online customers. A survey of logistic providers and retailers shows that 87% of the respondents are expected to use crowd-shipping in their last-mile delivery operations by 2028 (Zebra 2019). Many crowd-shipping companies, such as Instacart and Cornershop, rely solely on crowd-shippers for grocery delivery to online shoppers. Due to the proximity of brick-and-mortar stores to online customers, the fulfillment of online orders from stores enables faster delivery to customers, either by a company's delivery fleet or crowd-shipping. Furthermore, brick-and-mortar stores have access to a large pool of crowd-shippers. For instance, stores can employ in-store customers as crowd-shippers for delivering some online orders on their trip back home in exchange for a small compensation or store credit (Archetti, Savelsbergh, and Speranza 2016). However, the crowd-shipping operation is marred by uncertainty, primarily in crowd-shippers' availability and capacity, resulting in high operation cost and low reliability on service quality. Studies that incorporated the uncertainty in crowd-shipping operations are limited, and only few of those consider a dynamic problem setting (e.g., see (Dayarian and Savelsbergh 2020), (Raviv and Tenzer 2018), and (Yıldız 2021)).

We present a dynamic crowd-shipping operation employing in-store customers as crowd-shippers, who have a destination location and a maximum number of orders they are willing to serve. The online orders and crowd-shippers arrive randomly throughout the operation horizon. Online orders should be delivered within a few hours to their delivery location, and the orders that are not delivered within their delivery deadline are subject to a cost penalty, which reflects the cost of cancelling the orders or serving them late with other delivery options. With these information, our model aims to find an efficient assignment of online orders to crowd-shippers while minimizing the total operational cost of compensation to crowd-shippers and the penalty of not serving some online orders within their delivery deadline. Recent works by Arslan et al. (2019) and Dayarian and Savelsbergh (2020) are the closest studies to our problem. However, they either do not consider future uncertainty or suffer from scalability for real-time decision making. In contrast, our study mainly focuses on developing scalable policies that consider spatial and temporal uncertainty in

the arrival of crowd-shippers and online orders. Furthermore, our solution approach does not have any limit on incorporating attributes for online orders and crowd-shippers, and their associated uncertainties in a real-time decision making setting.

In determining the assignment of crowd-shippers to online orders throughout the rolling horizon, the main questions are as follows: (1) To whom should we assign the online orders? (2) Should we postpone some delivery in the hope for a crowd-shipper with a lower compensation in the future? (3) Should we wait for a crowd-shipper with a higher capacity to bundle some deliveries? (4) Can we increase the total number of served orders by making smarter assignment decisions? (5) How can we consider the future uncertainty and downstream cost in real-time decision making? In this research, we address this decision complexity by proposing an approximate dynamic programming (ADP) approach based on value function approximation, that is scalable to large instances, making it unique compared to previous studies in crowd-shipping with in-store customers.

The main contributions of the paper are as follows.

- We introduce a new variant of dynamic crowd-shipping operation with in-store customers.
- We formulate the problem as a Markov Decision Process (MDP).
- We develop an ADP algorithm for our problem setting with various enhancements (including hierarchical aggregation, stochastic step-size, monotonicity property, and alternative duals). To the best of our knowledge, this is the first use of an ADP policy for matching online orders to crowd-shippers in crowd-shipping. In addition, we develop an optimization-based myopic policy, which is intuitive and fast to generate policies for the crowd-shipping problem.
- We present an extensive numerical study by developing a comprehensive simulated instance for the City of Toronto downtown, and conduct a detailed computational analysis on the performance of the ADP algorithm and the resulting policy considering several enhancements. We also provide a detailed sensitivity analysis on various model parameters for comparing the ADP and myopic policies based on several performance measures (such as operational cost, percentage of served orders, average assigned orders per crowd-shippers, distance deviation of crowd-shippers, and average postponement per order).
- Our computational experiments provide important algorithmic and managerial insights. We show that the algorithmic enhancements are crucial for faster convergence of the ADP algorithm. Furthermore, our results indicate that imposing the monotonicity property with hierarchical aggregation yields a better quality of the ADP policy compared to using hierarchical aggregation in isolation. Compared to the myopic policy for various parameter settings, we observe that the ADP policy can yield up to 25.2% cost-saving, 9.8% increase in the number of served orders, 21.1% increase in the number of orders assigned per crowd-shipper, and 66.2% decrease in crowd-shippers distance deviation.

The remainder of this paper is structured as follows. We review the relevant literature in crowd-shipping in Section 2. We present the formal problem description in Section 3, and our MDP model along with the myopic policy in Section 4. We explain the proposed ADP algorithm and enhancements in Section 5. We discuss our thorough computational experiments in Section 6. Lastly, we summarize our main findings and future research directions in Section 7.

2. Literature Review

In the last few years, crowd-shipping has gained significant attention in the academic literature. For general overview about the crowd-shipping and industry practices, we refer the reader to the studies by Alnaggar, Gzara, and Bookbinder (2021) and Cleophas et al. (2019). Our literature review is mostly dedicated to operational planning and optimization algorithms in crowd-shipping with a special focus on stochastic and dynamic models. Moreover, we provide an extended discussion on the most relevant studies to our work, to better position our paper in the literature.

Many studies rely on deterministic models for crowd-shipping problems. Archetti, Savelsbergh, and Speranza (2016) introduced a vehicle routing problem with crowd-shipping by employing in-store customers as crowd-shippers to undertake a part of the last-mile delivery operation in exchange for a small compensation. Kafle, Zou, and Lin (2017) presented a two-tier delivery operation with trucks and crowd-shipping, and introduced a heuristic solution approach. Dai and Liu (2020) proposed a deterministic workforce scheduling and order allocation problem by considering in-house, full-time, and part-time crowd-shipping workforce. Boysen, Emde, and Schwerdfeger (2021) developed a crowd-shipping model that uses employees of distribution centers for delivering orders on their way back home with a compensation scheme based on number of delivered parcels.

Crowd-shipping operation involves a high degree of uncertainty since people participate in this delivery operation voluntarily, based on their free time and need for income, and without any commitment to the service provider. Few researches have made an attempt to address stochastic and dynamic aspects of crowd-shipping in their modelling and operational decision making. Gdowska, Viana, and Pedroso (2018) extended the work of Archetti, Savelsbergh, and Speranza (2016) to a single-stage stochastic model which incorporates the probability of crowd-shippers' willingness to accept or reject assigned delivery tasks. Dahle, Andersson, and Christiansen (2017) developed a two-stage stochastic vehicle routing model with dynamically appearing crowd-shippers. Their model decides on routing of company vehicles in the first stage, then assigns customers' packages to crowd-shippers or company vehicles, or postpone some deliveries in the second stage, after the appearance of any crowd-shippers. Skålnes et al. (2020) extended that work by developing a multi-stage stochastic model, and compared their solutions to those from its two-stage counterpart. Mousavi, Bodur, and Roorda (2020) proposed a two-tier delivery operation with mobile depots and

crowd-shipping, which incorporates uncertainty in crowd-shippers availability. They developed a two-stage stochastic model with mobile depot stopping location decisions in the first stage, and crowd-shipper-to-customer package assignment decisions in the second stage.

Raviv and Tenzer (2018) proposed a dynamic crowd-shipping delivery model based on series of service points as package drop-off, pickup, and intermediate locations. They developed a stochastic dynamic model that provides an optimal package routing policy through the network by crowd-shippers. As an extension, Yıldız (2021) proposed a simulation-based optimization approach based on calculating shadow cost of capacity utilization at service points. Allahviranloo and Baghestani (2019) proposed a dynamic optimization model for capturing the effect of crowd-shipping on urban areas by incorporating the travel activity of crowd-shippers and the customers.

Ulmer and Savelsbergh (2020) introduced the tactical problem of workforce scheduling in crowd-shipping. They incorporated continuous approximation and value function approximation methods for scheduling of delivery drivers. They showed that the obtained schedules obtained from the integration of these methods are superior to those by each method applied in isolation.

Arslan et al. (2019) introduced a dynamic crowd-shipping pick up and delivery problem that matches delivery tasks to crowd-shippers (referred to as ad-hoc drivers) and backup drivers in real time. Their model aims to minimize the compensation to crowd-shippers and routing cost of backup drivers. They developed a rolling-horizon framework that re-optimizes the decisions of their model each time a new information about crowd-shippers and delivery tasks is revealed. Their matching policy is a myopic policy based on assigning tentative matches to delivery tasks, and updating the matches in subsequent decision epochs in case of finding a better match, until the latest delivery time. Their proposed algorithm requires building routes of crowd-shippers and back-up drivers for their matching problem, which results in high computational time for online decision making, especially when multiple deliveries to crowd-shippers and backup drivers are allowed.

The most relevant work to our paper is by Dayarian and Savelsbergh (2020). They considered a problem setting where online orders are served by company's fleet and in-store customer (i.e., crowd-shippers) on their way back home. In their problem, the crowd-shippers and online orders arrive dynamically through the operation horizon. The problem decides on routing of the company vehicles and in-store customers with primary objective of minimizing violation in late delivery (i.e., after the orders delivery deadline) and secondary objective of minimizing total cost. They use a tabu search metaheuristic for solving the static vehicle routing problem. They proposed two policies for the dynamic decision making: (i) a myopic policy that does not consider any information about arrival of crowd-shippers and online orders, and (ii) a policy based on sample scenario planning that incorporates stochastic arrival of in-store customers and online orders. Our study differs in

terms of modelling and solution method from the previous study. Our model solely relies on in-store customer for serving orders and we do not consider routing of company owned vechicles. Many companies do not have a company owned vehicles for delivering customer packages, and they solely rely on third-party logistics which usually charge companies a fixed amount per package in urban and suburban areas (Boysen, Emde, and Schwerdfeger 2021). We do not let late delivery and any unserved orders within their delivery deadline are subject to penalty of not serving. We formally define an MDP formulation for crowd-shipping using in-store customers and focus on algorithmic challenges for real-time dynamic decision making, for which we propose an ADP policy. Our model is capable of considering multiple attributes for online orders and crowd-shippers without significantly affecting solution times for online decision making.

The mathematical modelling language of our crowd-shipping problem is based on the work by Spivey and Powell (2004), who formally proposed a general class of dynamic assignment problems and adaptive learning strategies for decision making. Their work is a cornerstone for modelling and solution strategy of dynamic fleet management problems, most notably the work of Simao et al. (2009), who used a forward-pass ADP algorithm for approximating the marginal value of fleet drivers. Due to the very large attribute space of the drivers, they incorporated hierarchical aggregation to improve the estimation of marginal values with a limited number of observations. A similar methodology was used by Al-Kanj, Nascimento, and Powell (2020) who proposed a dynamic ride-hailing system using autonomous vehicles. They also imposed monotonicity properties to improve the convergence of the ADP algorithm. Our problem shares similarities with the fleet management problem in terms of uncertainty related to order/demand arrivals, and the large scale attribute space. Furthermore, their ADP algorithm is made quite powerful with numerous algorithmic enhancements incorporated and tested over years of research. Therefore, we are inspired by the ADP algorithmic framework in the fleet management problem and tailored it for our problem for offline learning of online order attributes' marginal values. To the best of our knowledge, our work constitutes the first attempt to employ this algorithmic framework for a crowd-shipping problem. The policies derived from the ADP algorithm enable generating solutions for large-scale instances while considering downstream costs in our online decision-making setting.

3. Problem Description

We present a dynamic crowd-shipping model which employs in-store customers as crowd-shippers for delivering online orders. In this model, online orders and crowd-shippers show up in a highly dynamic fashion with spatial and temporal demand patterns through a multi-period decision horizon. We assume that the arrival rates of the crowd-shippers and the online orders are known through the operation horizon for each geographic area. We represent crowd-shipper destinations and online

order locations by small zones (e.g., with a width of 500 meters); therefore, each location can be attributed to multiple online orders' locations and crowd-shippers' destinations. The online orders arrive based on a stochastic process, expected to be delivered within their deadline (e.g., a few hours) to their delivery location via crowd-shippers (i.e., in-store customers). Orders that are not delivered by their deadline are subject to a penalty, e.g., as the cost of cancellation or late delivery.

Crowd-shippers are in-store customers who are willing to deliver online orders on their way back to their destinations. They have a destination location, a maximum number of orders they are able to serve, and a maximum allowable detour for accepting deliveries. In our model, crowd-shippers can make multiple deliveries (e.g., more than two deliveries) but only to two zones: the one of their own destination and another intermediate one. This is a reasonable assumption since (i) crowd-shippers participate in this activity for a small compensation rather than as a part-time job, and (ii) they usually travel back home after shopping, therefore, they might not be interested in making numerous stops at different delivery zones due to their time constraints and limited capacity. Relaxing this restriction is possible, however, the resulting model would fall into the vehicle routing problem class thus add to the problem complexity significantly. The maximum allowable detour for crowd-shippers is proportional to the distance from the store to their destinations. Crowd-shippers might declare their participation in the checkout or during their shopping in the store through an app, and usually leave the store immediately after shopping. Therefore, we assume that crowd-shippers only stay for a short amount of time in the store after declaring their interest for participating in crowd-shipping, which implies that a new set of crowd-shippers can be considered for every decision epoch. Crowd-shippers reimbursement follows a two-part compensation scheme which has a fixed compensation for every served package and a deviation compensation for the detour caused by deliveries. As mentioned by Boysen, Emde, and Schwerdfeger (2021), the latter is fair to the crowd-shippers; a crowd-shipper with a longer detour spends more time on the road, and pays higher running cost (e.g., gas), thus should expect a higher compensation.

Our model aims to minimize the total operational cost including crowd-shippers compensation and penalty of not serving online orders by their deadline. Decisions in each epoch include assignment of online orders to crowd-shippers and postponement of online orders to subsequent epochs, by considering future uncertainties and the impact of current decisions on downstream costs. In this research, we address this decision complexity by proposing an MDP model, and designing an ADP policy to overcome the challenge of real-time decision making under uncertainty.

4. Problem Formulation

We formulate an MDP model for the dynamic crowd-shipping problem, which employs in-store customers as crowd-shippers for delivering online orders. In that regard, we discretize the (finite)

planning horizon into time intervals, and assume decisions are made at the beginning of each interval and exogenous information is observed throughout the interval. Then, based on the decisions and observed exogenous information, the state of the system is updated at the end of the interval. We denote the set of decision-making epochs by $\mathcal{T} := \{1, 2, \ldots, T\}$. For ease of exposition, without loss of generality, we assume that the time between consecutive epochs is fixed, denoted by δ, i.e., the discretized time intervals are of equal length. Similarly, we divide the operational area into locations (e.g., geographical zones of user-defined size), $\mathcal{L} := \{0, 1, \ldots, L\}$, and assume without loss of generality that the store belongs to location 0. Given two locations $\ell, \ell' \in \mathcal{L}$, we denote the travel time and distance between them by $\texttt{time}(\ell, \ell')$ and $\texttt{dist}(\ell, \ell')$, respectively, which can be calculated/estimated by using a reference point per location, e.g., the "center" of a zone, and average travel speed. All the notation introduced in this section can be found in Table 4 of Appendix A.

In what follows, we explain the components of the MDP model, namely state variables, decision variables, cost function, exogenous information, and transition function. We conclude the section by introducing the optimal policy, ADP policies, and a myopic policy as possible solutions.

4.1. State Variables

The state of the system is characterized by available in-store crowd-shippers and online orders:

- An in-store crowd-shipper is described by two-dimensional attribute vector $c = (c_{\texttt{loc}}, c_{\texttt{cap}})$ capturing their destination location, $c_{\texttt{loc}}$, and capacity, $c_{\texttt{cap}}$. The latter refers to the maximum number of orders that can be served by the crowd-shipper, and belongs to the set $\mathcal{Q} := \{1, 2, \ldots, Q\}$. We denote the set of all possible crowd-shipper attribute vectors by $\mathcal{C} := \mathcal{L} \times \mathcal{Q}$.
- An online order is characterized by two-dimensional attribute vector, $o = (o_{\texttt{loc}}, o_{\texttt{due}})$, capturing their delivery location and deadline. The deadline attribute belongs to the set $\mathcal{D} := \{0, 1, \ldots, D\}$ and corresponds to the number of epochs left to dispatch the order from the depot for its on-time delivery. In other words, each order is given at most D epochs for dispatching, thus the maximum time of $D\delta$ for delivery considering the assumption of equi-spaced decision epochs, since joining the system. More specifically, when an online order with delivery point in location $o_{\texttt{loc}}$ is placed between decision epochs $t-1$ and t, it joins the system at the beginning of epoch t with

$$o_{\texttt{due}} = \min \left\{ T - t + 1,\ D - \left\lceil \frac{\texttt{time}(0, o_{\texttt{loc}})}{\delta} \right\rceil \right\}, \tag{1}$$

  i.e., $o_{\texttt{due}}$ is determined considering the number of epochs left to the end of horizon, and the largest dispatch epoch for its on-time delivery considering the travel time from the depot to its delivery location, $\texttt{time}(0, o_{\texttt{loc}})$. Note that this can be further adjusted considering the parcel packing and preparation time. We denote the set of all possible order attribute vectors by $\mathcal{O} := \mathcal{L} \times \mathcal{D}$.

Note that the destination of a crowd-shipper and an order might be the location 0, which does not mean that they are destined for the depot (which is assumed to belong to the location 0), since locations correspond to aggregated regions in the operational area such as traffic zones in a city.

Let $S_{tc}^{\texttt{Crowd}}$ and $S_{to}^{\texttt{Order}}$ be the number of in-store crowd-shippers with attribute vector $c \in \mathcal{C}$ and the number of online orders with attribute vector $o \in \mathcal{O}$ at epoch $t \in \mathcal{T}$, respectively. Then, the system state vector at $t \in \mathcal{T}$, before decisions are made, is $S_t := (S_t^{\texttt{Crowd}}, S_t^{\texttt{Order}})$, which consists of the crowd-shippers state vector, $S_t^{\texttt{Crowd}} = (S_{tc}^{\texttt{Crowd}})_{c \in \mathcal{C}}$, and the online orders state vector, $S_t^{\texttt{Order}} = (S_{to}^{\texttt{Order}})_{o \in \mathcal{O}}$.

4.2. Decision Variables

At each decision epoch $t \in \mathcal{T}$, given the current state of the system, we decide on a partition of the available orders into two sets. The first set consists of orders to be postponed to future decision epochs, with the hope that it can be served in a more cost-efficient manner (e.g., by a crowd-shipper who requires a smaller distance deviation, or when bundled with some remaining and/or future orders for a suitable crowd-shipper). For these decisions, we define for each $o \in \mathcal{O}$ the variable p_o^t denoting the number of online orders with attribute o that are postponed to next decision epoch.

For the second partition set, consisting of the orders to be dispatched, we decide on an assignment to available crowd-shippers. In that regard, we introduce two types of decision variables to distinguish orders that are served by crowd-shippers' first stop location after they leave the depot and their potential second stop location (which would be their own destination location) in delivering assigned orders: (i) y_{co}^t, the number of orders with attribute o served by crowd-shippers with attribute c, (ii) $z_{c\ell o}^t$, the number of orders with attribute o served by crowd-shippers with the same destination location while the crowd-shipper had a stop to serve orders with a different location ℓ.

We define the y_{co}^t variables over a subset of all possible order attribute vectors based on a concept of eligibility. This is due to crowd-shippers delivery flexibility, which we parametrize by $\zeta \geq 1$, the maximum ratio of distance a crowd-shipper is willing to travel to the direct distance between the store and their own destination. We define the set of locations eligible to visit as

$$\mathcal{L}_c^{\texttt{Elig}} := \left\{ \ell \in \mathcal{L} : \frac{\texttt{dist}(0,\ell) + \texttt{dist}(\ell, c_{\texttt{loc}})}{\texttt{dist}(0, c_{\texttt{loc}})} \leq \zeta \right\} \tag{2}$$

for crowd-shippers having attribute vector c. In other words, crowd-shippers with destination $c_{\texttt{loc}}$ are willing to stop by at location ℓ if the extra distance they travel, $\Delta_{c\ell} := \texttt{dist}(0,\ell) + \texttt{dist}(\ell, c_{\texttt{loc}}) - \texttt{dist}(0, c_{\texttt{loc}})$, is no more than $\zeta - 1$ times their original route length, $\texttt{dist}(0, c_{\texttt{loc}})$. Similarly, we define the set of orders eligible for crowd-shippers as those with an eligible delivery location:

$$\mathcal{O}_c^{\texttt{Elig}} := \left\{ o \in \mathcal{O} : o_{\texttt{loc}} \in \mathcal{L}_c^{\texttt{Elig}} \right\}. \tag{3}$$

As such, we only need to determine the values for $\{y_{co}^t\}_{c \in \mathcal{C}, o \in \mathcal{O}_c^{\texttt{Elig}}}$ variables.

For $c \in \mathcal{C}$, we define the $z^t_{c\ell o}$ variables over subsets of all possible order attribute vectors and locations, namely, $o \in \mathcal{O}_c^{\texttt{SameLoc}}$ and $\ell \in \mathcal{L}_{co}^{\texttt{EligMid}}$, which we explain next. For convenience, we define the set of order attribute vectors having the same location as a given crowd-shipper attribute and a given location, respectively, as follows:

$$\mathcal{O}_c^{\texttt{SameLoc}} := \{o \in \mathcal{O} : o_{\texttt{loc}} = c_{\texttt{loc}}\},\ \mathcal{O}_\ell^{\texttt{SameLoc}} := \{o \in \mathcal{O} : o_{\texttt{loc}} = \ell\}. \tag{4}$$

By definition of the z variables, the considered orders should have the same destination as crowd-shippers, thus only $o \in \mathcal{O}_c^{\texttt{SameLoc}}$ is taken into account for $c \in \mathcal{C}$. Moreover, such deliveries must follow deliveries at a different (which we call "Mid" standing for middle) location. While defining those middle locations, we also impose eligibility in terms of not only the distance deviation for crowd-shippers as before but also the order delivery windows:

$$\mathcal{L}_{co}^{\texttt{EligMid}} := \{\ell \in \mathcal{L}_c^{\texttt{Elig}} \setminus \{o_{\texttt{loc}}\} : \texttt{time}(0,\ell) + \mu + \texttt{time}(\ell, o_{\texttt{loc}}) \leq o_{\texttt{due}}\delta + \texttt{time}(0, o_{\texttt{loc}})\}. \tag{5}$$

The inequality constraint ensures that an order with attribute vector o can be delivered by its deadline despite crowd-shipper travels to a different location ℓ and spends the service time of μ before going to the order location $o_{\texttt{loc}}$. More specifically, if such an order is dispatched at epoch t, which corresponds to time $t\delta$, it will reach to its destination $o_{\texttt{loc}}$ at time

$$t\delta + \texttt{time}(0,\ell) + \mu + \texttt{time}(\ell, o_{\texttt{loc}}) \leq (t + o_{\texttt{due}})\delta + \texttt{time}(0, o_{\texttt{loc}}) \leq \text{actual delivery due time}$$

where the first and second inequalities follow from (5) and from the definition of $o_{\texttt{due}}$ given in (1), respectively. We note that the set of eligible middle locations is built in a conservative manner due to the first inequality above, i.e., some other locations might be possible to visit before delivering an order at a second stop on time. This is because we define the order attribute $o_{\texttt{due}}$ in terms of number of epochs rather than time in the continuous space, to have a finite domain. Nevertheless, our conservative middle eligible location set would be exact if locations are sufficiently far from each other, or can be made exact by increasing the granularity of the time horizon discretization.

Due to crowd-shippers capacity consideration, we define $x^t_{c\ell}$ to denote the number of assigned crowd-shippers with attribute c to the delivery location ℓ that is visited by the crowd-shippers immediately after leaving the store. Due to crowd-shippers delivery flexibility, only the locations in $\mathcal{L}_c^{\texttt{Elig}}$ are eligible for each crowd-shipper with attribute c. Therefore, we only define $\{x^t_{c\ell}\}_{c\in\mathcal{C}, \ell\in\mathcal{L}_c^{\texttt{Elig}}}$.

With above definitions, we define the decision tuple $\boldsymbol{a}^t = (x^t, y^t, z^t, p^t)$ at epoch t, where x^t,y^t,z^t, and p^t are decision vectors of variables $x^t_{c\ell}$, y^t_{co}, $z^t_{c\ell o}$, and p^t_o, respectively. We denote the feasible

decision set $\mathcal{A}^t(S_t)$ given the current state of the system $S_t = (S_t^{\texttt{Crowd}}, S_t^{\texttt{Order}})$. The tuple $\boldsymbol{a}^t \in \mathcal{A}^t(S_t)$ has to satisfy the following constraints:

$$\sum_{\ell \in \mathcal{L}_c^{\texttt{Elig}}} x_{c\ell}^t \leq S_{tc}^{\texttt{Crowd}} \qquad \forall c \in \mathcal{C} \tag{6a}$$

$$\sum_{c \in \mathcal{C}: o \in \mathcal{O}_c^{\texttt{Elig}}} y_{co}^t + \sum_{c \in \mathcal{C}\,:\, o \in \mathcal{O}_c^{\texttt{SameLoc}}} \sum_{\ell \in \mathcal{L}_{co}^{\texttt{EligMid}}} z_{c\ell o}^t + p_o^t = S_{to}^{\texttt{Order}} \qquad \forall o \in \mathcal{O} \tag{6b}$$

$$\sum_{o \in \mathcal{O}_\ell^{\texttt{SameLoc}}} y_{co}^t + \sum_{o \in \mathcal{O}_c^{\texttt{SameLoc}}\,:\, \ell \in \mathcal{L}_{co}^{\texttt{EligMid}}} z_{c\ell o}^t \leq c_{\texttt{cap}} \cdot x_{c\ell}^t \qquad \forall c \in \mathcal{C},\ \ell \in \mathcal{L}_c^{\texttt{Elig}} \tag{6c}$$

$$\sum_{o \in \mathcal{O}_\ell^{\texttt{SameLoc}}} y_{co}^t \geq x_{c\ell}^t \qquad \forall c \in \mathcal{C},\ \ell \in \mathcal{L}_c^{\texttt{Elig}} \tag{6d}$$

$$x_{c\ell}^t \in \mathbb{Z}_+ \qquad \forall c \in \mathcal{C},\ \ell \in \mathcal{L}_c^{\texttt{Elig}} \tag{6e}$$

$$y_{co}^t \in \mathbb{Z}_+ \qquad \forall c \in \mathcal{C},\ o \in \mathcal{O}_c^{\texttt{Elig}} \tag{6f}$$

$$z_{c\ell o}^t \in \mathbb{R}_+ \qquad \forall c \in \mathcal{C},\ o \in \mathcal{O}_c^{\texttt{SameLoc}},\ \ell \in \mathcal{L}_{co}^{\texttt{EligMid}} \tag{6g}$$

$$p_o^t \in \mathbb{R}_+ \qquad \forall o \in \mathcal{O} \tag{6h}$$

Constraints (6a) indicate the maximum number of crowd-shippers with attribute c that can be assigned to eligible locations for serving online orders. Constraints (6b) are the online orders conservation constraints, in which online orders with attribute o have three possible decision cases: (i) being served by a crowd-shipper who immediately visits their location after leaving the store, (ii) being served by a crowd-shipper who serves their destination after having a stop at another location, and (iii) being postponed to next epoch. Constraints (6c) ensure that the maximum number of orders that can be served by these crowd-shippers are equal to the crowd-shippers' total capacity. Therefore, for crowd-shippers with attribute c, that are assigned to visit location ℓ, the number of served online orders with attribute o in location ℓ, or in the crowd-shippers destination cannot exceed the total available capacity of the assigned crowd-shippers. Constraints (6d) indicate that the number of served online orders by crowd-shippers with attribute c at location ℓ are at least equal to the number of crowd-shippers with attribute c that are assigned to location ℓ.

We note that by keeping integrality restrictions for variables x and y, we can relax the integrality restrictions for variables p and z. Let $\mathbb{V} = \begin{bmatrix} \mathbb{V}_{(6b)} \\ \mathbb{V}_{(6c)} \end{bmatrix}$ be the coefficient matrix of variable z and p for Constraints (6b) and (6c), where every element in $\mathbb{V}$ is either 0 or +1. The matrix $\mathbb{V}$ satisfies the Hoffman's sufficient totally unimodularity condition (e.g., see (Wolsey and Nemhauser 1999, Corollary 2.8)) since every column of $\mathbb{V}$ has at most two +1 elements, and for columns with two +1 elements, each element belongs to either $\mathbb{V}_{(6b)}$ and $\mathbb{V}_{(6c)}$.

4.3. Cost Function

The total operational cost incurred as a result of actions $\boldsymbol{a}^t$ at epoch $t \in \mathcal{T}$, denoted by $C(\boldsymbol{a}^t)$, is

$$\sum_{c\in\mathcal{C}} \sum_{\ell\in\mathcal{L}_c^{\texttt{Elig}}} f_{c\ell}^{\texttt{Dev}} \Delta_{c\ell} x_{c\ell}^t + \sum_{o\in\mathcal{O}} \sum_{c\in\mathcal{C}:o\in\mathcal{O}_c^{\texttt{SameLoc}}} \sum_{\ell\in\mathcal{L}_{co}^{\texttt{EligMid}}} f_o^{\texttt{Fix}} z_{c\ell o}^t + \sum_{o\in\mathcal{O}} \sum_{c\in\mathcal{C}:o\in\mathcal{O}_c^{\texttt{Elig}}} f_o^{\texttt{Fix}} y_{co}^t + \sum_{o\in\mathcal{O}:o_{\texttt{due}}=0} f^{\texttt{NotServ}} p_o^t. \quad (7)$$

The first term is the compensation paid to crowd-shippers based on deviation from their original route, the second and third terms represent the fixed cost compensation to crowd-shippers for each served package, and the last term shows the cost of not serving online orders within their delivery deadline (only applied for the orders that are failed to be served at their last eligible epoch).

4.4. Exogenous Information

We have two types of exogenous information: random arrival of online orders and crowd-shippers. We assume that arrivals follow a known stochastic process (e.g., Poisson process). We define the crowd-shippers new information arrival variable vector as $W_t^{\texttt{Crowd}} = (W_{tc}^{\texttt{Crowd}})_{c\in\mathcal{C}}$, where $W_{tc}^{\texttt{Crowd}}$ is the number of crowd-shippers with attribute c between decision epochs t and $t+1$. Analogously, we define the online order new information arrival variable vector as $W_t^{\texttt{Order}} = (W_{to}^{\texttt{Order}})_{o\in\mathcal{O}}$, where $W_{to}^{\texttt{Order}}$ includes the new orders that have recently arrive to the system (in this case order due date is defined as in (1)) and existing orders that are cancelled, or whose delivery locations are changed by e-shoppers. Accordingly, the exogenous information arrival vector for epoch $t \in \{1,\dots,T-1\}$ between decision epochs t and $t+1$ is defined as $W_t = (W_t^{\texttt{Crowd}}, W_t^{\texttt{Order}})$.

4.5. Transition Function

The transition function from state S_t to S_{t+1} depends on the decision tuple, $\boldsymbol{a}^t$, and arrival of online orders and crowd-shippers between time periods t and $t+1$. By introducing the concept of post-decision state (see Powell (2011)), which is the state obtained immediately after making decisions and before arrival of new information, we can break the transition function in two parts: (i) transition to post-decision state, $S^{\texttt{Order-Post}}$, via action $\boldsymbol{a}^t$, and (ii) transition from the post-decision state to the next state based on the random information W_t, respectively:

$$S_t^{\texttt{Order-Post}} = \texttt{statepost}(S_t, \boldsymbol{a}^t), \quad S_{t+1} = \texttt{statenext}(S_t^{\texttt{Order-Post}}, W_t) \quad (8)$$

As mentioned in Section 3, it is assumed that in each time period there is a new set of crowd-shippers. That is, all crowd-shippers available at the beginning of each epoch t will leave the system by start of the time epoch $t+1$, either being assigned some orders or leaving empty-handed. Accordingly the post-decision state only depends on the number of postponed orders p^t, where the delivery deadline of online orders is reduced by one:

$$S_{to}^{\texttt{Order-Post}} = \begin{cases} 0 & \text{if } o_{\texttt{due}} = D \\ p^t_{(o_{\texttt{loc}}, o_{\texttt{due}}+1)} & \text{otherwise} \end{cases} \quad (9)$$

The online orders at epoch $t+1$ is obtained by combining the post-decision state at epoch t and the new arrivals between epochs t and $t+1$:

$$S^{\texttt{Order}}_{t+1,o} = S^{\texttt{Order-Post}}_{to} + W^{\texttt{Order}}_{to} \qquad \forall o \in \mathcal{O} \tag{10}$$

For crowd-shippers, since we assume that crowd-shippers leave the store within a short time after declaring their willingness for crowd-shipping, only the new arrivals affect the next state:

$$S^{\texttt{Crowd}}_{t+1,c} = W^{\texttt{Crowd}}_{tc} \qquad \forall c \in \mathcal{C} \tag{11}$$

4.6. Policies

Optimal policy. In our problem, the objective is to minimize the expected total cost over the operation horizon, which consists of the total compensation paid to crowd-shippers and the total penalty of not serving customers within their delivery deadline:

$$\min_{\pi \in \Pi} \mathbb{E}_{W=(W_1,\ldots,W_T)} \left[\sum_{t \in \mathcal{T}} C\Big(\mathtt{A}^{\pi}_t\big(\mathtt{S}^{\pi}_t(W)\big)\Big) \middle| S_1 \right].$$

Given initial state of the system, S_1, we aim to find a policy π from set of feasible policies Π such that when the actions suggested by the policy, $\mathtt{A}^{\pi}_t(S_t)$, are sequentially implemented at realized states, $\mathtt{S}^{\pi}_1(W) = S_1$ and $\mathtt{S}^{\pi}_{t+1}(W) = \mathtt{statenext}\big(\mathtt{statepost}(\mathtt{S}^{\pi}_t, \mathtt{A}^{\pi}_t(\mathtt{S}^{\pi}_t)), W_t\big)$ for $t \in \{1,\ldots,T-1\}$, the minimum cost is obtained (on expectation with respect to the underlying stochastic process described by W). Note that although the total cost depends on the realization of the full vector W, the actions taken or the state visited at an epoch only depend on the random variables revealed up to that point, i.e., they are nonanticipative.

An optimal policy can be obtained by backward dynamic programming (Powell 2011) using

$$V_t(S_t) = \min_{\boldsymbol{a}^t \in \mathcal{A}^t(S_t)} C(\boldsymbol{a}^t) + \mathbb{E}_{W_t}\left[V_{t+1}(S_{t+1})\right], \tag{12}$$

where $S_{t+1} = \mathtt{statenext}(\mathtt{statepost}(S_t, \boldsymbol{a}^t), W_t)$ as mentioned before. $V_t(S_t)$ is referred to as the *value function* of state S_t. The procedure starts from the final epoch, $t = T$, setting all the values to zero, and then moves backward in time to update the state value functions based on the cost of optimal decisions and the probability of transition from one state to another. The recursion continues until reaching the first stage, $t = 1$. However, this approach requires enumeration of all possible outcomes and actions, which is impractical even for small instances.

Approximate policies. Due to computational intractability of the exact value function $V_t(S_t)$ for large-scale problems, a common technique is to replace the model in (12) with a more tractable one. In this work, we propose an ADP policy based on value function approximation in terms of post-decision states. We refer to this policy as `ADP`, whose details are presented in Section 5.3.

Myopic policy. We also design a myopic policy, denoted by `Myopic`, as a benchmark for `ADP`. Myopic policies are commonly used in practice, and usually designed based on intuition about the problem. They are called myopic since they do not consider cost of downstream decisions. In our problem, we are interested in increasing the number of served orders, and subsequently avoiding the cost of not serving. Hence, we propose a myopic policy that requires serving orders by crowd-shippers as soon as possible. For each location, it prioritizes orders that have a smaller delivery dispatch time. Also, this policy assigns an order only if it costs less than the cost of not serving.

We can impose this policy by modifying the cost function , and introducing additional constraints to decision constraint set . First, in the last term of the cost function (7), we include the cost of not serving for all order attributes (not just $o_{\texttt{due}} = 0$). Second, we add the following constraints to (6) along with a new set of decision variables, e, for orders with non-zero delivery deadline:

$$p_o^t \leq M_o^{(13a)} e_o^t \qquad \forall o \in \mathcal{O} : o_{\texttt{due}} > 0 \qquad (13a)$$

$$\sum_{c \in \mathcal{C}: o \in \mathcal{O}_c^{\texttt{Elig}}} y_{co}^t + \sum_{c \in \mathcal{C}\,:\, o \in \mathcal{O}_c^{\texttt{SameLoc}}} \sum_{\ell \in \mathcal{L}_{co}^{\texttt{EligMid}}} z_{c\ell o}^t \leq M_o^{(13b)} (1 - e^t_{(o_{\texttt{loc}}, o_{\texttt{due}}-1)}) \qquad \forall o \in \mathcal{O} : o_{\texttt{due}} > 1 \qquad (13b)$$

$$\sum_{o' \in \mathcal{O}: o'_{\texttt{loc}} = o_{\texttt{loc}}, o'_{\texttt{due}} > o_{\texttt{due}}} (1 - e_{o'}^t) \leq M_o^{(13c)} (1 - e_o^t) \qquad \forall o \in \mathcal{O} : o_{\texttt{due}} > 0 \qquad (13c)$$

$$e_o^t \in \{0, 1\} \qquad \forall\, o \in \mathcal{O} : o_{\texttt{due}} > 0 \qquad (13d)$$

Constraints (13a) define the e variables value, 1 if an order is postponed to the next epoch, 0 otherwise. Constraints (13b) ensure that for orders with attribute o are not assigned to crowd-shippers if the orders with one less delivery deadline is postponed to the next epoch. Constraints (13c) force to postpone all orders orders with higher delivery deadline if an order with a lower delivery deadline is postponed to the next epoch. In (13a) and (13b), for each row with order attribute o, the minimum (big-M) values of $M_o^{(13a)}$ and $M_o^{(13b)}$ are equal to $S_{to}^{\texttt{Order}}$. Similarly, for each row with order attribute o in (13c), the minimum value is $M_o^{(13c)} = (D - o_{\texttt{due}})$.

5. Solution Methodology

The presented dynamic problem is plagued with intractability in outcome space, state space, and action space, even for a moderately sized instances. Powell (2011) refers to this phenomena as "three curses of dimensionality", and proposes an ADP framework as a solution strategy for overcoming solution complexity of these type of problems. In this paper, we propose a look-ahead policy obtained by using value function approximation as a solution. The proposed algorithm is based on offline learning of value functions through simulation of the problem's Markov decision process. The algorithmic framework adopted for this work combines mathematical programming, reinforcement leaning, and machine learning techniques for learning value functions. In this section, we first discuss

how we overcome each of the three curses of dimensionality, then provide the steps of the ADP algorithm, and conclude with incorporating three enhancement techniques, namely, hierarchical aggregation, bias-adjusted Kalman filter step-size, and monotonicity property to speed up value function learning in the offline phase.

5.1. Post-decision State

In dynamic programming, for solving the Bellman optimality equation (12) for state S_t, we require the expected value of downstream cost obtained by multiplying the value function of all possible outcome of S_{t+1} with the probability of occurrence of S_{t+1} based on random information arrival W_t. For large-scale problems, outcome space is very large, which leads to the first curse of dimensionality. To avoid enumerating the whole outcome space and computing the expectation future value, Powell (2011) introduced the concept of the post-decision state (see Section 4.5 for more details). This concept enables us to break the dynamic programming recursion equation (12) into two parts. First, we define a *deterministic* optimality equation based on post-decision state as

$$V_t(S_t) = \min_{\boldsymbol{a}^t \in \mathcal{A}^t(S_t)} \{C(\boldsymbol{a}^t) + V_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}})\}. \tag{14}$$

Second, we express *post-decision state value function* as the expected value of downstream cost as

$$V_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}}) = \mathbb{E}_{W_t}\left[V_{t+1}(S_{t+1}) \middle| S_t^{\texttt{Order-Post}}\right] \tag{15}$$

where $S_{t+1} = \texttt{statenext}(S_t^{\texttt{Order-Post}}, W_t)$. Solving the Bellman optimality equation based on post-decision state, described in (14), requires the knowledge of value functions for all possible post-decision states. Due to the high dimensionality of the post-decision state space, calculating the value functions of the post-decision states is impractical with the current approach. As such, we apply a post-decision state value function approximation as explained next.

5.2. Value Function Approximation

In our problem, the post-decision state is based on online order state vector, whose state space set grows exponentially as we increase the number of order locations, delivery deadlines, or decision epochs. This complexity is the second curse of dimensionality, i.e., dimensionality of state space. To overcome this complexity, we propose to approximate the post-decision state value function $V_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}})$ by a function $\bar{V}_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}})$ which is a linear function of online order vector attributes in the post-decision state $\{S_{to}^{\texttt{Order-Post}}\}_{o \in \mathcal{O}}$. Due to the fine discretization of the operation horizon for the purpose of real-time decision making, the number of orders and crowd-shippers for each attribute in state S_{t+1} is highly likely to be only 0 or 1 at each decision epoch. This form of approximation is also proposed for fleet management problems, e.g, in the studies by Simao et al.

(2009) and Al-Kanj, Nascimento, and Powell (2020). The linear approximation of post-decision value function is formally defined as

$$\bar{V}_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}}) = \sum_{o \in \mathcal{O}} \bar{v}_{to}^{\texttt{Post}} S_{to}^{\texttt{Order-Post}} \tag{16}$$

where $\bar{v}_{to}^{\texttt{Post}}$ is the expected cost of serving an online order with attribute o from the next epoch $t+1$ until the end of operation horizon. The proposed approximation has a significant computational advantage, namely, instead of obtaining value functions of all possible post-decision states at epoch t, we only require $|\mathcal{O}|$ variables, $\bar{v}_{to}^{\texttt{Post}}$. In Section 5.4, we outline an ADP algorithm for obtaining a statistical estimate of $\bar{v}_{to}^{\texttt{Post}}$ using Monte Carlo sampling of the random information.

5.3. ADP Policy

We address the third curse of dimensionality, i.e., action space, by introducing the ADP policy, `ADP`. First, we write the post-decision state value function as the function of postponement decisions p^t using the post-decision state definition in (9) as follows:

$$\bar{V}_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}}) = \sum_{o \in \mathcal{O}: o_{\texttt{due}} > 0} \bar{v}^{\texttt{Post}}_{t,(o_{\texttt{loc}}, o_{\texttt{due}}-1)} p_o^t. \tag{17}$$

Furthermore, by replacing the value function of the post-decision state in (14) with the linear approximation introduced in (17), `ADP` decisions are obtained via

$$\mathtt{A}_t^{\texttt{ADP}}(S_t) \in \underset{\boldsymbol{a}^t \in \mathcal{A}^t(S_t)}{\arg\min} \{C(\boldsymbol{a}^t) + \bar{V}_t^{\texttt{Post}}(S_t^{\texttt{Order-Post}})\}. \tag{18}$$

`ADP` suggests to serve online orders only if the cost of assigning them to crowd-shippers is no more than the approximate marginal cost of serving them at a later decision epoch. Due to integrality restrictions on decision variables, the assignment problem in the optimality equation (18) is an integer program, which can be efficiently solved with state-of-the-art solvers within less than a second even for large-scale problems (as shown in our computational section) for online decision making. `ADP` only requires knowing the value function approximations; in the next section we explain a procedure to estimate them.

5.4. Forward-pass ADP Algorithm

We adopt a forward-pass ADP algorithm for estimating the value functions of online orders in our problem. This algorithmic framework was initially developed for large-scale fleet management problems (e.g., study of Simao et al. (2009) and Al-Kanj, Nascimento, and Powell (2020)). To the best of our knowledge, our work is the first adaptation of such an algorithm in a crowd-shipping problem. Similar to the aforementioned studies, we adopt a pure exploitation forward pass, since the exploration effort has a negligible value for problems with high-dimensional state space (Powell

2011, p. 464). In what follows, we walk through our derivations, approximations, and algorithms, mostly adopted from the aforementioned studies and tailored for our own problem setting. We index the algorithmic procedure's main iterations by n, and add that index as a superscript to all of the MDP model and ADP algorithm components.

Algorithm 1 summarizes the steps of the forward-pass ADP algorithm. We first explain the main

Algorithm 1 Forward-pass ADP algorithm

Step 0: Initialize post-decision values $(\bar{v}_{to}^{\texttt{Post},0})_{t\in\mathcal{T},o\in\mathcal{O}}$, state S_1^0, and step-size α^1

Step 1: For $n=1,\ldots,N$:

  Step 2: Choose a sample path $(\hat{W}_t^n)_{t\in\mathcal{T}}$

  Step 3: For $t=1,\ldots,T$:

    Step 3.a: Solve the ADP optimality equation in (19), obtain optimal decisions $\boldsymbol{a}^{t,n}$, and compute marginal values of Constraints (6b), $\vartheta_t^n=(\vartheta_{to}^n)_{\forall o\in\mathcal{O}:\, S_{to}^{\texttt{Order},n}>0}$ defined in (21)

    Step 3.b: Compute $(\vartheta_{t-1,o}^{\texttt{Post},n})_{o\in\mathcal{O}}$ using (23), and update post-decision value function estimates of the previous epoch, $(\bar{v}_{t-1,o}^{\texttt{Post},n-1})_{o\in\mathcal{O}}$, via (24)

    Step 3.c: Based on optimal decisions $\boldsymbol{a}^{t,n}$ and random information realization $\hat{W}_t^n$, move from the current state S_t^n to the next state S_{t+1}^n using (9)-(11)

Step 4: Return the final approximation values $\bar{v}^{\texttt{Post},N}=(\bar{v}_{to}^{\texttt{Post},N})_{t\in\mathcal{T},o\in\mathcal{O}}$ for real-time decision making

logic of the algorithm, and then discuss its more important steps in detail. The algorithm starts by initializing the value functions of post-decision states, system state, and step-size for learning value functions. In iteration n, we draw a realization of random variables for all epochs of the operation horizon, $(\hat{W}_t^n)_{t\in\mathcal{T}}$, which represents the random arrival information of crowd-shippers, $(\hat{W}_t^{\texttt{Crowd},n})_{t\in\mathcal{T}}$, and online orders, $(\hat{W}_t^{\texttt{Order},n})_{t\in\mathcal{T}}$. In Step 3, we start from the first epoch and move forward in time based on optimal actions and observations from the random information sample path, and update the value functions of post-decision states. More specifically, in Step 3.a, we obtain optimal decisions solving the Bellman optimality equation based on post-decision value function approximations from the previous iteration, $\bar{V}_t^{\texttt{Post},n-1}(\cdot)$, and derive the marginal values of observed orders in epoch t, denoted as ϑ_t^n. In Step 3.b, we use the marginal values ϑ_t^n to update the value function of post-decision states at the *previous* epoch, $t-1$, since the value function of a post-decision state at epoch $t-1$ is an estimated cost of serving orders from epoch t until the end of horizon. In Step 3.c, we move to the next state in epoch $t+1$ using the optimal decisions from Step 3.a and the random information realization $\hat{W}_t^n$. In Step 4, we output the final estimates for value functions, $\bar{v}^{\texttt{Post},N}$, which can be used to implement `ADP` using (18).

In what follows, we explain the detailed derivation, corresponding equations, and some implementation tips of Steps 3.a and 3.b in Algorithm 1.

Step 3.a. At decision epoch t, by visiting state S_t^n, we obtain optimal decisions $\boldsymbol{a}^{t,n}$ by solving

$$F_t(S_t^n) := \min_{\boldsymbol{a}^t \in \mathcal{A}^t(S_t^n)} \left\{ C(\boldsymbol{a}^t) + \bar{V}_t^{\texttt{Post},n-1}(S_t^{\texttt{Order-Post},n}) \right\} \tag{19}$$

based on the estimated value function of post-decision state from iteration $n-1$, namely

$$\bar{V}_t^{\texttt{Post},n-1}(S_t^{\texttt{Order-Post},n}) = \sum_{o \in \mathcal{O}} \bar{v}_{to}^{\texttt{Post},n-1} S_{to}^{\texttt{Order-Post},n}. \tag{20}$$

We note that for real-time decision making via (19), the number of order and crowd-shipper attributes might be quite large as a result of using a fine discretization of the operation horizon. As such, it might be impractical to define the decision variables $\boldsymbol{a}^t$ and model the decision constraints (6) describing $\mathcal{A}^t$ for all possible attributes in the problem. Fortunately, in each epoch, we only have a limited number of orders and crowd-shippers, thus we can model only the components for the observed order attributes in the system., i.e., $S_{to}^{\texttt{Order},n}, S_{tc}^{\texttt{Crowd},n} > 0$. Henceforth, we suppose that this convention has been employed while solving (19).

The *marginal values* for visited order attributes, $\vartheta_t^n = (\vartheta_{to}^n)_{\forall o \in \mathcal{O}: S_{to}^{\texttt{Order},n} > 0}$, can be defined as partial derivatives of F_t with respect to the right-hand side of the flow conversation Constraints (6b):

$$\vartheta_{to}^n := \frac{\partial F_t(S_t^n)}{\partial S_{to}^{\texttt{Order},n}} \qquad \forall o \in \mathcal{O} : S_{to}^{\texttt{Order},n} > 0. \tag{21}$$

ϑ_{to}^n captures the cost of serving an additional order with attribute o in epoch t or at a later epoch. Since the ADP optimality equation (19) is an integer program, the challenge is to find an efficient method for deriving ϑ. For this purpose, we implement and compare the use of numerical derivatives and the duals of the linear programming (LP) relaxation of the integer program. We calculate the numerical derivatives using

$$\vartheta_{to}^n = \frac{F_t(S_t^n)^{o+} - F_t(S_t^n)^{o-}}{2} \qquad \forall o \in \mathcal{O} : S_{to}^{\texttt{Order},n} > 0 \tag{22}$$

in which we average the left and right numerical derivative with respect to the order attribute with $S_{to}^{\texttt{Order},n} > 0$. We define the $F_t(S_t^n)^{o+}$ and $F_t(S_t^n)^{o-}$ as the optimal expected cost of the states that have one more, and one less order with attribute o in the original state S_t, respectively. Due to integrality of the Constraint (6b) right-hand sides, the numerical derivative is a valid method but it can be computationally very expensive. The method requires solving the ADP optimality equation, (19), two more times for each order attribute with $S_{to}^{\texttt{Order},n} > 0$. In contrast, deriving marginal values of integer linear programs via the duals of the LP relaxation is a computationally efficient method, although it would usually yield unreliable marginal values due to lack of strong duality. Fortunately, we show in Section 6.3 that the quality of `ADP` based on LP duals is very good due the small optimality gap for our integer programs.

As a side note regarding the computation of numeral derivatives, while solving the ADP optimality equation (19) to compute $F_t(S_t^n)^{o+}$ and $F_t(S_t^n)^{o-}$, we can benefit from the solution of the original optimality equation, the one yielding $F_t(S_t^n)$. For example, having an additional order with attribute o results in a cost ranging from fixed cost to the cost of order postponement. Hence, the optimal value $F_t(S_t^n)^{o+}$ has the lower bound of $F_t(S_t^n) + f^{\texttt{Fix}}$ and the upper bound of $F_t(S_t^n) + \bar{v}_{to}^{n-1}$ (which can be provided by means of a warm-start solution). The introduced upper and lower bounds can be imposed in the branch-and-bound method, which can significantly reduce the computational burden in calculating the numerical derivatives.

Step 3.b. The observed marginal values at decision epoch t can be used for updating the value functions of the post-decision states at $t-1$. The reason to update estimates from the previous epoch is that a postponed order with attribute o from epoch $t-1$ appears in the post-decision state of epoch $t-1$, $S_{t-1,o}^{\texttt{Order-Post},n}$, as such directly affects the next state at epoch t, i.e., $S_{to}^{\texttt{Order},n} = S_{t-1,o}^{\texttt{Order-Post},n} + \hat{W}_{t-1,o}^{\texttt{Order},n}$. Therefore, the marginal values obtained at epoch t, ϑ_t^n, corresponding to the value of having one more order at that epoch, sheds light on the cost of postponing an order at the previous epoch.

In that regard, we first define the post-decision analog of the marginal values. While executing Step 3.b for t, we compute such marginals for epoch $t-1$, and denote them by $\vartheta_{t-1}^{\texttt{Post},n}$. That is, such marginals are computed *at* t *for* $t-1$. More specifically, $\vartheta_{t-1,o}^{\texttt{Post},n}$ is defined as the marginal value of having an additional order with attribute o in post-decision state $S_{t-1,o}^{\texttt{Order-Post},n}$ with respect to F_t. For computing this marginal values, Simao et al. (2009) suggested the following formula:

$$\vartheta_{t-1,o}^{\texttt{Post},n} := \sum_{o' \in \mathcal{O}} \frac{\partial F_t(S_t^n)}{\partial S_{t,o'}^{\texttt{Order},n}} \frac{\partial S_{t,o'}^{\texttt{Order},n}}{\partial S_{t-1,o}^{\texttt{Order-Post},n}} \Bigg|_{W_{t-1}^{\texttt{Order}} = \hat{W}_{t-1}^{\texttt{Order},n}} \tag{23}$$

where $\frac{\partial F_t(S_t^n)}{\partial S_{t,o'}^{\texttt{Order},n}} = \vartheta_{t,o'}^n$ as computed in Step 3.a, and $\frac{\partial S_{t,o'}^{\texttt{Order},n}}{\partial S_{t-1,o}^{\texttt{Order-Post},n}}$ is the relative change in the number of orders with attribute o' in $S_{t,o'}^{\texttt{Order},n}$ with respect to the number of orders with attribute o in $S_{t-1,o}^{\texttt{Order-Post},n}$. Note that due to the random information $W_{t-1}^{\texttt{Order}}$, we require to account for all realized random transitions in $\hat{W}_{t-1}^{\texttt{Order},n}$. If an order with attribute o evolves to an order with attribute o' due to random information $\hat{W}_{t-1}^{\texttt{Order},n}$ (e.g., change in order delivery location), the value of $\frac{\partial S_{t,o'}^{\texttt{Order},n}}{\partial S_{t-1,o}^{\texttt{Order-Post},n}}$=1, and accordingly $\vartheta_{t-1,o}^{\texttt{Post},n} = \vartheta_{t,o'}^n$. This means that we only need to keep track of order transitions from one attribute to another instead of summing over all attributes in (23) (Simao et al. 2009). In our numerical experiments in Section 6, we only account for the arrival of new online orders to the system as the random information (i.e., no change in the attributes of the existing online orders due to random information arrival), accordingly, the Equation (23) can be reduced to $\vartheta_{t-1,o}^{\texttt{Post},n} = \vartheta_{to}^n$.

For any visited order attribute, $\forall o \in \mathcal{O} : S_{to}^{\texttt{Order},n} > 0$, by obtaining the observed marginal value $\vartheta_{t-1,o}^{\texttt{Post},n}$ from (23), we update the post-decision state value function $\bar{v}_{t-1,o}^{\texttt{Post},n-1}$ via

$$\bar{v}_{t-1,o}^{\texttt{Post},n} = (1-\alpha^n)\bar{v}_{t-1,o}^{\texttt{Post},n-1} + \alpha^n \vartheta_{t-1,o}^{\texttt{Post},n} \tag{24}$$

where α^n is the desired step-size (e.g., BAKF step-size, see Section 5.5.1 for more details).

5.5. Enhancements

Obtaining robust estimates for the value functions of order attributes requires visiting states with each attribute at least for few times. In our problem, due to the large state space, estimating the post-decision value functions in a reasonable number of iterations is not viable with the basic version of Algorithm 1. In this section, we discuss some algorithmic enhancements to improve value function estimation and in turn the convergence of the ADP algorithm.

5.5.1. Hierarchical aggregation and BAKF step-size In problems with a large state space, one of the effective strategies is to aggregate the state space. In Monte Carlo simulation, aggregation increases the number of observations for value function approximations resulting in estimates with smaller estimation error at the expense of structural error due to the aggregation. Therefore, a right level of aggregation needs to balance the trade off between bias of aggregation and error of value function estimation. In that regard, George, Powell, and Kualkarni (2008) proposed a method, namely Hierarchical Aggregation (HA), which provides an improved estimation of disaggregate value functions by weighted combination of the value functions in different levels of aggregation. In our problem, it is likely that orders with spatial and temporal correlation have a similar value function. The HA algorithm enables us to pass the observed value functions of an order attribute for updating value functions of the "neighbouring" orders.

Let $\mathcal{G} = \{0, 1, \ldots, G\}$ be the index set of aggregation levels, where 0 refers to the disaggregated level, and $\mathcal{O}^g$ be the attribute space for aggregation level $g \in \mathcal{G}$. For example, the level of aggregation g can define a less granular representation of customer locations, i.e., each location in the aggregation level g consists of few disaggregated locations. We define $M^g : \mathcal{O} \to \mathcal{O}^g$ for $g \in \mathcal{G}$ as the aggregation function that maps the disaggregate attributes to the attribute with aggregation level g. That is, it provides the order attribute o^g in the aggregation level g that includes the disaggregated attribute o, i.e., $o^g = M^g(o)$. We also define the value function of online orders in post-decision states for each level of aggregation $g \in \mathcal{G}$ as $\bar{v}_{t-1}^{\texttt{Post},g} = (\bar{v}_{t-1,o^g}^{\texttt{Post},g})_{o^g \in \mathcal{O}^g}$. Then, we use the weighted sum of estimates in different aggregation levels to obtain an improved estimate for post-decision value functions of attribute o:

$$\bar{v}_{t-1o}^{\texttt{Post}} := \sum_{g \in \mathcal{G}} w_{t-1,o}^g \, \bar{v}_{t-1,M^g(o)}^{\texttt{Post},g} \tag{25}$$

where $\{w^g_{t-1,o}\}_{g\in\mathcal{G}}$ are weights in different aggregation levels for post-decision state with attribute o at epoch $t-1$. George, Powell, and Kualkarni (2008) showed that weights inversely proportional to the statistical estimates of variances and biases of each aggregated level are near-optimal.

In addition, we implement bias-adjusted Kalman filter (BAKF) step-size in (24) for efficient learning and convergence of value function estimations with limited number of observations. George and Powell (2006) proposed this step-size for non-stationary data and showed that it results in a faster convergence compared to other step-size rules for updating value functions. The BAKF step-size aims to balance the variance of observations and the transient bias. Therefore, it results in smaller step-sizes for large variations and low biases of value function observations, and conversely, larger step-sizes for value function observations with small variations and high biases.

In order to incorporate HA in the forward-pass ADP algorithm, we only require to implement the HA algorithm in Step 3.b of Algorithm 1 to obtain the improved estimates of value functions. The HA algorithm obtains the observed value functions of post-decision states in the disaggregated level as an input and requires to compute the observed value functions for each aggregated level based on the aggregation mapping function. Let

$$\mathcal{H}^{g,n}_{t-1,o^g} = \{o \in \mathcal{O} : M^g(o) = o^g, \exists\, \vartheta^{\texttt{Post},n}_{t-1,o}\} \qquad \forall o^g \in \mathcal{O}^g, g \in \mathcal{G} \tag{26}$$

be the set of observed disaggregated post-decision state attributes which are mapped to the aggregated attribute o^g at epoch $t-1$ in iteration n. We define the observed aggregated post-decision value functions as $\vartheta^{\texttt{Post},g,n}_{t-1,o^g}$ for $g \in \mathcal{G}, o^g \in \mathcal{O}^g : |H^{g,n}_{t-1,o^g}| \neq \emptyset$, where $\vartheta^{\texttt{Post},g,n}_{t-1,o^g}$ is equal to the average of disaggregated observation as follows:

$$\vartheta^{\texttt{Post},g,n}_{t-1,o^g} = \frac{\sum_{o\in H^{g,n}_{t-1,o^g}} \vartheta^{\texttt{Post},n}_{t-1,o}}{|H^{g,n}_{t-1,o^g}|} \qquad \forall g \in \mathcal{G}, \forall o^g \in \mathcal{O}^g : |H^{g,n}_{t-1,o^g}| \neq \emptyset \tag{27}$$

For example, if an aggregated attribute includes four disaggregated attributes and only two of them are observed, then we take the average of the two observed value functions to represent the value function for the aggregated attribute. The detailed algorithmic procedure for HA with BAKF step-size is given in Algorithm 2 in Appendix B.

5.5.2. Monotonicity of value functions We exploit the monotonicity property for post-decision value functions of online orders with respect to order delivery deadlines, which helps with accelerating the value function learning process. Jiang and Powell (2015) showed that exploiting monotonicity properties can increase the rate of convergence in ADP algorithms. As mentioned earlier, $\bar{v}^{\texttt{Post}}_{to}$ represents the expected cost of serving an order, with attribute o in post-decision state at epoch t, immediately after decision in epoch t until end of the horizon. Therefore, it is intuitive that for online orders with the same delivery location, the orders with higher delivery

deadlines have a smaller value since they have more opportunities to be matched with a crowd-shipper requiring a lower compensation in subsequent decision epochs. To be more specific, at epoch t and location ℓ, for two online orders $o=(o_{\texttt{loc}}, o_{\texttt{due}}), o'=(o'_{\texttt{loc}}, o'_{\texttt{due}}) \in \mathcal{O}_\ell^{\texttt{SameLoc}}$, we would like to have $o_{\texttt{due}} < o'_{\texttt{due}} \implies \bar{v}_{to'}^{\texttt{Post}} \leq \bar{v}_{to}^{\texttt{Post}}$. Such a monotonicity property has a higher chance to hold true under the following assumptions:

(i) For $o, o' \in \mathcal{O}$ with $o_{\texttt{due}} \leq o'_{\texttt{due}}$ and $o_{\texttt{loc}} = o'_{\texttt{loc}}$, the transition of post-decision states to online orders with attributes o and o' via random information realization $\hat{W}$ of W_t, respectively denoted by $o^{\hat{W}}$ and $o'^{\hat{W}}$, satisfy $o^{\hat{W}}_{\texttt{due}} \leq o'^{\hat{W}}_{\texttt{due}}$.

(ii) The cost of serving orders is independent of orders delivery deadline.

In our problem, the online orders random information consists of only arrivals of new orders (i.e, no change in the delivery location of existing orders, nor their delivery deadline), as such the assumption (i) holds. The assumption (ii) is also satisfied since the compensation paid to crowd-shippers depends only on the number of served orders and the distance deviation. Although it is possible to prove the above-mentioned monotonicity property for a system with the capacity of one order, the proof may not go through for the general case. Nevertheless, such an intuitive property might be desirable to have in a policy. Moreover, our results in Section 6 confirm that monotone ADP policies can be obtained significantly more efficiently, and they can be of high quality.

To obtain such monotone policies, in the ADP algorithm, we impose the monotonicity of the value functions with respect to delivery deadline after every update step as a result of a new observation. Jiang and Powell (2015) showed that using the latest observation for imposing the monotonicity property is asymptotically optimal. Our approach is simple; we use the leveling algorithm (Powell 2011, p. 502) to enforce monotonicity based on the latest observation as defined below.

Definition 1. In iteration n of Algorithm 1, for each location $\ell \in \mathcal{L}$ we have a new observation $\vartheta_{t-1,o}^{\texttt{Post},n}$ which updates the value function $v_{t-1,o}^{\texttt{Post},n-1}$ to $v_{t-1,o}^{\texttt{Post},n}$ via Equation (24). In order to preserve the monotonicity of the value functions for $o \in \mathcal{O}_\ell^{\texttt{SameLoc}}$, we apply the following:

$$v_{t-1,o'}^{\texttt{Post},n} = \begin{cases} \min\{v_{t-1,o'}^{\texttt{Post},n-1}, v_{t-1,o}^{\texttt{Post},n}\} & \text{if } o'_{\texttt{due}} < o_{\texttt{due}} \\ v_{t-1,o}^{\texttt{Post},n} & \text{if } o'_{\texttt{due}} = o_{\texttt{due}} \\ \max\{v_{t-1,o'}^{\texttt{Post},n-1}, v_{t-1,o}^{\texttt{Post},n}\} & \text{if } o'_{\texttt{due}} > o_{\texttt{due}} \end{cases} \qquad o' \in \mathcal{O}_\ell^{\texttt{SameLoc}} \tag{28}$$

In short, the rules in (28) increase or decrease the value estimates of the attributes that are violating the monotonicity property according to the latest observation. For implementing the monotonicity with the HA algorithm, we impose the monotonicity property via Definition 1 to the value function estimates on every aggregation level, as well as to the improved value estimates obtained from Algorithm 2.

6. Computational Experiments

This section presents the results of our extensive computational experiments and our thorough policy analysis to provide insights on (i) efficiency and efficacy of the proposed ADP algorithm, and (ii) benefits of the proposed policies based on various performance measures with sensitivity analysis on the model parameters. For conciseness of expression, we continue to use `ADP` and `Myopic` to denote the ADP and Myopic policies, respectively.

Section 6.1 describes our test instances that are based on the City of Toronto. Section 6.2 provides implementation details of the ADP algorithm and the policy evaluation procedure for `ADP` and `Myopic`. In Section 6.3, we show the convergence of the ADP algorithm, and the quality of `ADP` based on different combinations of the algorithmic enhancements. Then, we compare `ADP` and `Myopic` in terms of various performance measures including the followings:

- Operational cost and cost breakdown for each cost component (Section 6.4)
- Average assigned orders and distance deviation per crowd-shipper (Section 6.5)
- Served orders and assigned crowd-shippers in the operation horizon (Section 6.6)
- Average postponements of the served online orders (Section 6.7)

6.1. Instance Generation

We generate our test instances based on the City of Toronto. Our data generation relies on the 2016 version of the Transportation Tomorrow Survey (TTS), a comprehensive travel survey that is conducted every five years in the Toronto Area (DMG 2018). The study area is the City of Toronto downtown core which has 117 locations (i.e., traffic zones) with non-zero residential population. The store location has a shopping mall with well-known big-box and grocery stores generating shopping trips from this location. Figure 12 in Appendix C shows the study area, the store location, and the online order arrival heat map for total online orders of $N_o = 1000$. We obtain the travel distances in our study area from the City of Toronto road network. For simplicity in our implementation, we fix the road network speed to $Speed = 20$ (km/h) and accordingly obtain travel times in the study area. Nevertheless, our model is capable of considering dynamic travel times.

We consider an operation horizon from 9 a.m. to 10 p.m. discretized into 52 equal-length time periods, each with length of $\delta = 15$ minutes. We assume that online orders arrive until 8 p.m. and the last two hours are the closing period in which we only have crowd-shipper arrivals. Figure 1 shows the total arrival profile of online orders and crowd-shippers in each time period for $N_o = 1000$ orders and $N_c = 1000$ crowd-shippers. We assume that there are two surges in the number of orders before noon and close to the end of regular work hours due to the fact that many people might make an order at those times (e.g., they can have their order when they arrive back home). For crowd-shipper arrivals, we consider a surge in arrivals after regular working hours which is consistent with the

trends from real data from Los Angeles and Orange County based on the study of Allahviranloo and Baghestani (2019). For our computational experiments, we consider three levels of the total order arrivals, $N_o \in \{500, 1000, 1500\}$, and five ratios of crowd-shipper arrivals to online order arrival, $N_c/N_o \in \{0.6, 0.8, 1.0, 1.2, 1.4\}$. We consider a Poisson process for online order arrivals and crowd-shipper arrivals for which we obtain their corresponding temporal profile from Figure 1. In each

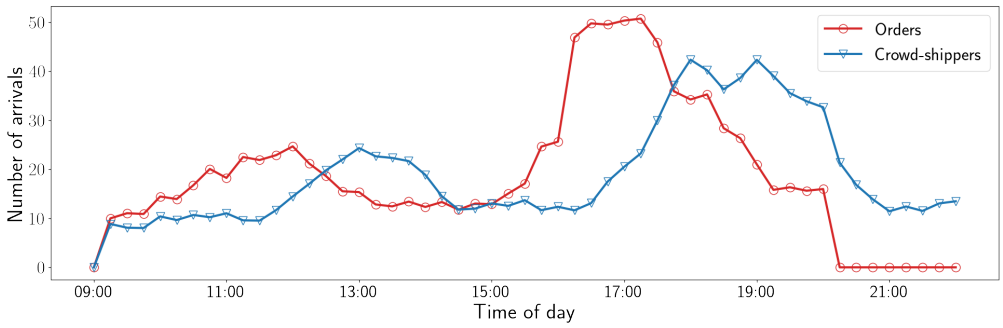


Figure 1 **Arrival profile of crowd-shippers and online orders for $N_c = 1000, N_o = 1000$.**

time period the average number of online order arrivals for each location is generated proportional to the residential population of the corresponding location. Similarly, the crowd-shipper trips (i.e., in-store customers) are generated proportional to the TTS non-discretionary trips (i.e., shopping trips) between the store location and all locations. For simplicity in the arrivals generation, we assume the spatial proportion of online order and crowd-shipper arrivals are homogeneous for all time periods. We also assume that crowd-shippers have a capacity of $Q \in \{1, 2, 3, 4\}$ with equal probability in all test instances. For our sensitivity analysis, we consider three levels for crowd-shipper flexibility rate, $\zeta \in \{1.1, 1.3, 1.5\}$, which are within range of values in the study of Archetti, Savelsbergh, and Speranza (2016).

The cost of same day delivery for Amazon Canada ranges from 6.99 CAD to 11.99 CAD (Amazon.ca 2021), and express delivery by Walmart U.S. is 10 USD on top of existing delivery charges (Walmart Inc 2020). Accordingly, we fix the cost of not serving to $f^{\texttt{NotServ}} = 10$, which is within range of the aforementioned values. Regarding the compensation scheme for crowd-shippers, we perform a sensitivity analysis on the fixed compensation per order and deviation compensation rate per unit of distance deviation by crowd-shippers, considering $f^{\texttt{Fix}} \in \{2, 3, 4\}$ and $f^{\texttt{Dev}} \in \{2, 3, 4\}$, respectively. These values are within the range of values in the study of Dahle et al. (2019) who performed a sensitivity analysis on different compensation schemes.

We analyze three levels of delivery deadline for online orders, namely $D \in \{4, 8, 12\}$, with unit of number of epochs (e.g., $D = 8$ is equal to two hours in our study). For all computational experiments, we consider service time of $\mu = 15$ minutes at intermediate delivery locations, which accounts for package service time at customer doors and intrazonal travel times (traffic zones represent online order locations).

6.2. Implementation Details

For implementation, we use Python 3, and we run all the experiments on Compute Canada servers, namely Beluga, Cedar, and Graham (Computecanada 2021). For solving Bellman optimality equations we use Gurobi optimizer (version 9.0.1). In the ADP algorithm, the value function approximations are usually initialized as zero. However, in our model, the minimum expected cost of serving online orders is equal to the fixed cost, thus we initialize approximated values as the fixed cost, $f^{\texttt{Fix}}$. For HA, we assume three levels of spatial aggregation. Figure 13 in Appendix C shows the aggregation levels for our study. The most disaggregated level has the originally defined 117 locations, the first level of aggregation has 35 locations, and the second level has 17 locations. For evaluations of `ADP` and `Myopic`, we average the results from $|\Omega| = 1000$ simulated sample paths of the operation horizon, characterized by random realization of online order and crowd-shipper arrivals. Table 5 in Appendix D shows all the parameters for the computational experiments.

6.3. Algorithm Performance

In this section, we present our findings regarding the effect of HA, the monotonicity property, and value function update mechanism on solution quality and convergence of the ADP algorithm.

Figure 2 shows the total operation cost over the first 200 iterations of the ADP algorithm when different combinations of the algorithmic enhancements are considered. The total operational cost for an iteration is obtained through simulating the corresponding policy (described by the value function approximations at that iteration). The results show that the ADP algorithm without enhancements (Basic) has the slowest convergence rate, while incorporating both HA and monotonicity (HA-MT) results in the fastest convergence, especially yielding significant improvement at early iterations. Also, the version with only HA enhancement (HA) has a similar early improvement as the HA-MT variant, however, it performs worse than all other variants after 200 iterations. Incorporating only the monotonicity property (MT) slightly improves the convergence compared to the Basic variant. As an example, for reaching the total operational cost just below 5% of `Myopic`, the ADP policies from Basic, MT, HA, and HA-MT take 70, 55, 42, and 30 iterations, respectively.

After 1000 iterations, the total operational cost based on Basic, MT, HA, and HA-MT ADP variants converges to 3585.2, 3581.7, 3611.9, and 3582.1, respectively. These results show three important observation which requires more in-depth explanations for understanding our ADP algorithm value function update mechanism and their implications for practical purposes:

(i) *Basic ADP and MT ADP policies after 1000 iterations have a comparable performance.* In the Basic ADP, by visiting states with different orders attribute sufficiently, the monotonicity property is highly likely to be satisfied. In the initial iterations of Algorithm 1, the online order value functions for each location start to first rise for the order attributes with delivery

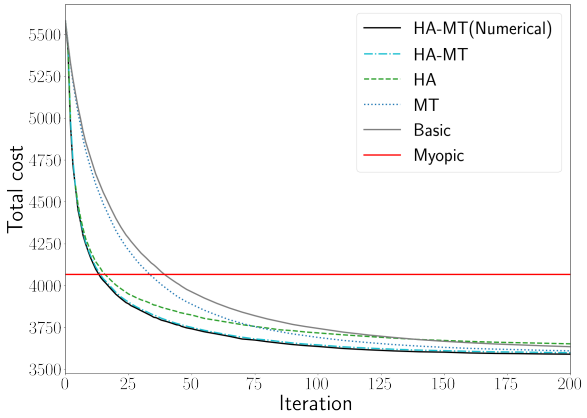


Figure 2 **Performance of different ADP algorithms.**

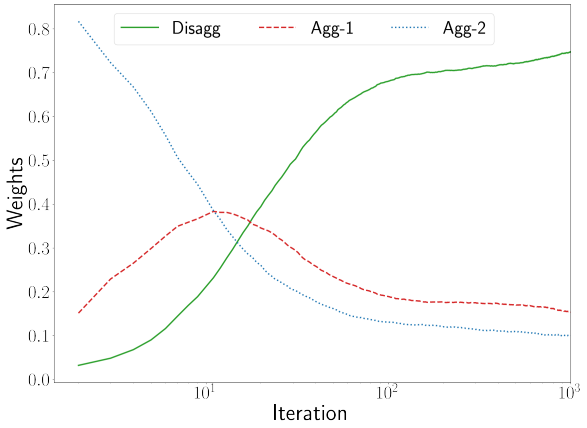


Figure 3 **Hierarchical aggregation weights for HA-MT.**

deadline of zero (due to the penalty of not serving) and then propagate to the orders with later delivery deadlines in the subsequent iterations. Thus, for a specific location, the value function of orders with earlier delivery deadline is usually higher than the orders with later delivery deadlines through the value iteration algorithm. This observation also explains the reason behind the marginal improvement of MT ADP compared to the Basic ADP in Figure 2.

(ii) *The solution quality of HA ADP policy is slightly worse than the other ADP algorithm variants.* Without imposing the monotonicity property, it is quite likely to diverge from the monotonicity property in the weighted value function approximation (i.e., Equation (25)). Mes, Powell, and Frazier (2011) proved that, in the context of stochastic search, the estimation provided by HA is asymptotically unbiased for a learning policy that ensures an infinite observation of points. However, there has not been such a proof in the context of approximating value functions (Al-Kanj, Nascimento, and Powell 2020). Based on our observations, we hypothesize that imposing the monotonicity properties of value functions while using HA may result in a less biased (or unbiased) ADP policy; we leave investigating this observation for future research.

(iii) *HA-MT ADP algorithm has comparable performance to Basic ADP after sufficient number of iterations.* In our problem, the value functions of online order attributes are only required for the locations that have online orders. In the initial iterations of Algorithm 1, the online order value functions for each location start to first rise for the order attributes with delivery deadline of zero (due to the penalty of not serving) and then propagate to the orders with later delivery deadlines in the subsequent iterations. Therefore, the Basic ADP can visit all online order attributes and provide a value function estimation after a sufficient number of iterations, which results in a comparable policy quality compared to HA-MT ADP. On the contrary, incorporating HA in fleet management problem results in a significant improvement in the ADP solution quality compared to the Basic variant (see Sections 3.1 and 3.3 in (Simao et al. 2009) and (Al-Kanj, Nascimento, and Powell 2020) for more details). Nevertheless,

incorporating the monotonicity property and HA can be essential for practical purposes in our problem. In assignment of orders to crowd-shippers, the decision epochs may need to be reduced to few seconds (in our test instances, the decision epoch duration is 15 minutes). This would substantially increase the number of online order attributes requiring a much higher number of ADP iterations. In this case, algorithmic enhancements would be essential for getting a good `ADP` due to the need of high computation time for offline learning.

Lastly, Figure 2 shows that the difference in the quality of `ADP` obtained by using the numerical derivative (HA-MT Numerical) and LP duals (HA-MT), in updating value functions is negligible. This stems from the fact that the optimality gap of integer program representing the ADP Bellman optimality Equation (19) is very small. In our computational experiments, this gap is on average about 2.46%. In terms of computational time, the ADP algorithm with numerical derivatives takes more than twice compared to the LP dual version. Thus, using LP duals can yield a significant computational gain with a minimal effect on the `ADP` quality.

Figure 3 shows the average weights for different levels of aggregations in HA through the ADP algorithm iterations averaged over all the attributes. Over the iterations, the weights at the disaggregated level start to increase due to having more observations in that level due to decrease in the variance of value function estimates. After 1000 iterations, the average weight of the disaggregated level is around 0.72. At early iterations, order attributes with smaller delivery deadline are visited for learning the value functions, but as the algorithm progresses, these attributes will not be visited further even after a large number of iterations (e.g., in specific location and time period, if the values of the orders are high resulting in high cost of postponements, it is very likely to serve the orders earlier and not to visit the order attributes). The limited observations for those attributes result in a high variance of estimation in the disaggregated level, which puts the most weights to the aggregated level based on Equation (38). Furthermore, the range of value functions is quite limited (from fixed cost to cost of not serving). It is likely that the bias between the aggregated and disaggregated levels would be small in some cases. Thus, the weights might be mostly distributed based on the variance of the estimates for different aggregation levels, which also contributes to not reaching a large weight (e.g., close to 1) for the disaggregated level in Figure 3.

In terms of computational effort for online decision making, the solution time (with Intel Gold 6148 Skylake @ 2.4 GHz CPU, 4 GB RAM (single thread)) for the Bellman optimality equation is sufficiently small, on average 0.31 seconds. We note that the optimality equation can be solved by inspection when the number of crowd-shippers arrival in a period is small. This is usually the case when the decision epoch duration is short for real-time decision making. For example, if the decision epoch duration is less than inter-arrival time of crowd-shippers, we expect only few arrivals in each time-period. Therefore, we only need to choose from combination of assignment

of orders to crowd-shippers (considering the constraints of the model) which minimizes the cost of postponement plus the cost of assigning order to crowd-shippers. Such a solution method can greatly help for both speeding up the offline learning and online decision making.

The rest of the result section (Sections 6.4 - 6.7) is dedicated to the sensitivity analysis for our proposed policies. For all those analyses, we use ADP of HA-MT obtained after 1000 iterations.

6.4. Operational Cost Analyses

In this section, we first analyze ADP's total operational cost and cost-saving compared to Myopic by performing various sensitivity analyses. We also provide ADP cost breakdown, and the savings over Myopic for each cost component, namely fixed cost, deviation cost, and cost of not serving. Our results show that ADP significantly outperforms Myopic, yielding cost-savings up to 25.2%. Next, with detailed policy analysis, we explain how these savings are obtained in different settings.

Table 1 presents the sensitivity analysis of the total operational cost of ADP and its percentage cost-saving compared to Myopic with respect to total order arrivals, N_o, delivery deadline, D, and crowd-shippers to orders ratio, N_c/N_o. We observe few trends from these results:

Table 1 Total operational cost and the cost-saving of ADP.

		$N_c/N_o = 0.6$		$N_c/N_o = 0.8$		$N_c/N_o = 1$		$N_c/N_o = 1.2$		$N_c/N_o = 1.4$	
N_o	$\mathcal{D}$	ADP cost	Saving %	ADP cost	Saving %	ADP cost	Saving %	ADP cost	Saving %	ADP cost	Saving %
500	4	3081.1	3.47	2748.2	5.34	2528.5	6.79	2362.7	7.94	2237.5	8.84
	8	2564.8	6.15	2247.1	8.45	2063.9	10.01	1941.2	11.30	1866.7	12.20
	12	2351.3	6.40	2059.2	8.14	1910.3	9.81	1814.8	11.73	1756.4	13.21
1000	4	5290.9	6.38	4646.0	8.73	4239.7	10.03	3981.5	10.68	3805.7	11.00
	8	4362.6	9.78	3850.0	10.96	3590.5	11.70	3443.5	12.36	3354.9	12.99
	12	4082.7	8.68	3642.2	9.31	3430.4	10.92	3319.6	12.34	3252.0	13.47
1500	4	7268.9	8.50	6334.2	10.83	5800.1	11.44	5500.5	11.29	5294.5	11.16
	8	6049.4	11.58	5418.7	11.34	5107.2	11.55	4945.9	11.87	4833.4	12.29
	12	5799.4	9.41	5233.0	9.32	4972.3	10.70	4819.5	11.81	4754.3	12.39

- As expected, increasing the number of crowd-shippers, N_c/N_o ratio, results in a decrease in total operational cost due to having more options for matching orders to crowd-shippers. Similarly, an increase in delivery deadline, D, decreases total operational cost since orders have a lower chance of remaining unserved and a higher chance of being matched with a crowd-shipper requiring lower compensation. We note that, these trends are also observed in Myopic, however, whose cost are excluded in Table 1.
- An increase in D leads to a higher percentage cost-saving of ADP. With larger D, ADP is able to match orders with crowd-shippers who require smaller deviation, or to assign more orders per crowd-shipper. Furthermore, ADP has a better delivery prioritization based on order locations

and delivery deadlines to avoid missing deliveries to some locations. This prioritization is guided by the learned value function approximations. More specifically, orders with lower likelihood of being matched with a crowd-shipper have higher value function estimates, thus likely to be prioritized in online assignment to maximize the total number of served orders.

- Similarly, an increase in N_c/N_o results in a higher percentage of cost-saving by `ADP`. Having more crowd-shippers provides the opportunity of making a better assignment between orders and crowd-shippers. `ADP` captures this opportunity by lowering value function estimates of online orders to avoid inefficient assignments.
- An increase in N_o also results in a higher percentage of cost-saving by `ADP`. A higher number of online orders results in a higher order density in delivery operations providing an opportunity for more efficient assignment of crowd-shippers to online orders. `ADP` exploits this opportunity by postponing online orders with the hope of assigning orders to crowd-shippers with lower deviation and achieving a higher number of orders assigned per crowd-shipper.

Figure 4 provides the breakdown of the total operation cost, for each cost component of the objective function in (7) for three levels of crowd-shipper range, ζ. We observe the following trends:

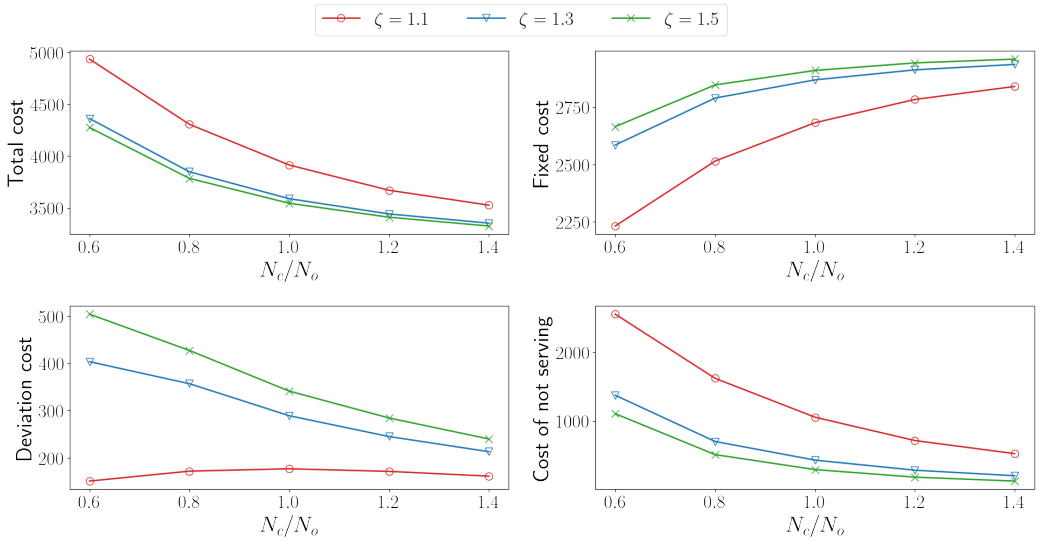


Figure 4 **Effect of crowd-shipper deviation range and crowd-shipper-to-order ratio on cost components for `ADP`.**

- Similar to the total cost, as mentioned above, the cost of not serving decreases with increase in N_c/N_o. Also, the total cost and the cost of not serving are higher for lower levels of ζ.
- The fixed cost increases in ζ and N_c/N_o since it is possible to serve more orders.
- The deviation cost decreases with an increase in N_c/N_o for larger crowd-shipper deviation range values, while for $\zeta = 1.1$, it initially increases and then decreases. For $\zeta = 1.1$ and low N_c/N_o, the crowd-shipping operation is under stress due to lack of crowd-shippers and `ADP`

does not have much flexibility in assigning orders to crowd-shippers. In this case, the initial increase in N_c/N_o results in serving more orders and accordingly higher total paid deviation compensation to crowd-shippers. By further increasing N_c/N_o, ADP has the opportunity to exploit the higher availability of crowd-shippers by assigning orders to crowd-shippers who require smaller distance deviations to reduce the deviation cost.

Table 2 presents the sensitivity analysis on ADP percentage of cost-saving with respect to crowd-shipper deviation range, fixed compensation, deviation compensation, and crowd-shippers to online orders ratio, for $D = 8$ and $N_o = 1000$. We have the following main observations:

Table 2 Effect of crowd-shippers deviation range and compensation scheme on ADP cost saving (%).

		$\zeta = 1.1$					$\zeta = 1.3$					$\zeta = 1.5$				
$N_o/N_c \longrightarrow$		0.6	0.8	1.0	1.2	1.4	0.6	0.8	1.0	1.2	1.4	0.6	0.8	1.0	1.2	1.4
f^{Fix}	f^{Dev}															
2	2	9.2	10.2	10.0	9.0	7.9	14.6	14.9	14.4	14.1	14.3	15.9	17.2	17.6	18.3	19.2
	3	8.7	9.9	9.8	9.2	8.3	13.5	15.3	16.3	17.0	18.0	14.8	17.9	19.9	21.5	22.9
	4	8.5	9.6	9.8	9.4	8.9	12.9	15.8	17.7	19.3	20.6	14.3	18.3	21.3	23.5	25.2
3	2	6.9	7.4	7.0	6.1	5.3	10.5	10.5	10.1	10.0	10.2	11.3	12.1	12.5	13.0	13.6
	3	6.6	7.1	6.9	6.3	5.7	9.8	11.0	11.7	12.4	13.0	10.8	12.9	14.3	15.5	16.5
	4	6.4	7.0	7.0	6.6	6.2	9.5	11.5	12.9	14.0	14.9	10.5	13.3	15.3	16.8	18.0
4	2	5.2	5.3	4.9	4.3	3.7	7.6	7.7	7.5	7.5	7.8	8.2	8.9	9.3	9.8	10.3
	3	4.9	5.2	5.0	4.5	4.1	7.2	8.2	8.8	9.4	9.9	7.9	9.6	10.8	11.7	12.4
	4	4.8	5.1	5.1	4.8	4.6	7.0	8.5	9.6	10.4	11.1	7.7	9.8	11.3	12.4	13.2

- An increase in ζ results in a higher percentage saving by ADP. With a larger crowd-shippers range, Myopic is able to assign online orders to crowd-shippers who require a large distance deviation, which translates into a higher compensation. However, ADP considers the future opportunity for assignment of orders to crowd-shippers who require less deviation with the help of estimated value functions for the orders.
- Interestingly, an increase in N_c/N_o results in mixed trends on ADP percentage cost-saving for different crowd-shipper deviation ranges. As an example, in Table 2, for $\zeta = 1.1, f^{\text{Fix}} = 3$ and $f^{\text{Dev}} = 3$, ADP cost-saving initially increases from 6.6% to 7.1% and ultimately decreases to 5.7% with an increase in N_c/N_o to 1.4. For $\zeta = 1.5, f^{\text{Fix}} = 3$ and $f^{\text{Dev}} = 3$, the cost-saving increases from 10.8 % to 16.5 % with increase in N_c/N_o. To explain this phenomenon, we first provide the breakdown of the percentage saving of ADP on each cost component in Figure 5 for three levels of $\zeta \in \{1.1, 1.3, 1.5\}$ with $N_o = 1000$, $f^{\text{Fix}} = 3$, and $f^{\text{Dev}} = 3$. We see that ADP results in a cost-saving in two ways:
  (i) Decreasing total crowd-shippers deviation compensation: The low value of ζ restricts crowd-shipper assignment with large distance deviation and also provides fewer opportunities for the order assignment. For $\zeta = 1.1$, ADP does not have much leverage on reducing

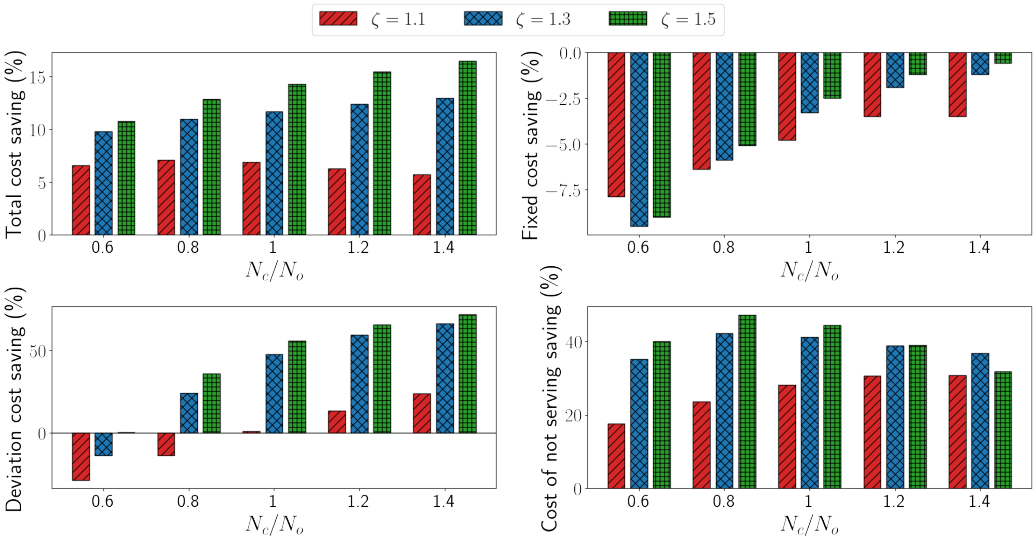


Figure 5 **Percentage cost-saving of `ADP` on each cost component for different levels of ζ and N_c/N_o.**

crowd-shippers deviation compensation and mostly leverage on decreasing the cost of not serving by increasing the number of served orders. As shown on Figure 5, the distance deviation percentage of cost saving is negative for $N_c/N_o \in \{0.6, 0.8\}$. `ADP` is heavily constrained by the limited crowd-shipper deviation range for serving orders, therefore, it increases the number of served orders by efficient prioritization in assigning crowd-shippers to orders at the expense of increase in deviation compensation of crowd-shippers.

(ii) Reducing cost of not serving: By (further) increasing N_c/N_o, `ADP` loses its edge in increasing the number of served orders; the high availability of crowd-shippers closes the gap with `Myopic`. Thus, the cost-saving mostly comes from the distance deviation which is modest for the low value of ζ resulting in reduction of `ADP` cost-saving. On the contrary, for all instances with $\zeta = 1.5$, the total cost-saving increases as N_c/N_o gets larger.

Table 2 also presents the results of a sensitivity analysis on crowd-shipper compensation parameters, i.e., fixed compensation per package $f^{\texttt{Fix}}$ and deviation compensation rate per kilometer $f^{\texttt{Dev}}$. We observe that an increase in $f^{\texttt{Fix}}$ results in a decrease in total cost-saving of `ADP`. Higher values of $f^{\texttt{Fix}}$ increase the total cost substantially in both `ADP` and `Myopic`; therefore, the cost-saving due to reduction in deviation compensation and cost of not serving becomes less significant. Surprisingly, the increase in $f^{\texttt{Dev}}$ leads to a mixed trend in `ADP` cost-saving. For a lower level of ζ and N_o/N_c, the increase in $f^{\texttt{Dev}}$ decreases the cost-saving. In this case, `ADP` mostly aims to serve orders as much as possible to avoid the large cost of not serving. This increase in served orders simply translates to a higher crowd-shippers deviation; therefore, increase in $f^{\texttt{Dev}}$ negatively affects `ADP` cost-saving. In contrast, for a higher level of ζ and N_o/N_c, `ADP` can yield a higher cost-saving by decreasing the deviation cost, and accordingly, increase in $f^{\texttt{Dev}}$ magnifies `ADP` cost-saving.

6.5. Assigned Order and Crowd-shipper Deviation Analyses

This section analyzes the effect of ADP on individual crowd-shippers, in terms of assigned orders and distance deviation. Figure 6 shows the average assigned orders per crowd-shipper through the operation horizon for both ADP and Myopic. As a reminder, in our test instances, crowd-shippers can have a capacity of 1 to 4, with equal likelihood. The figure indicates that ADP achieves a higher

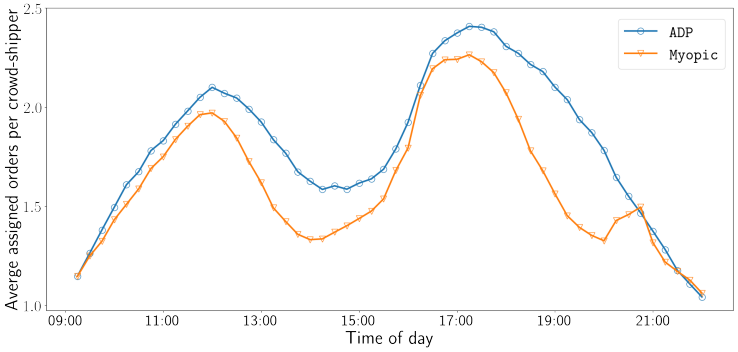
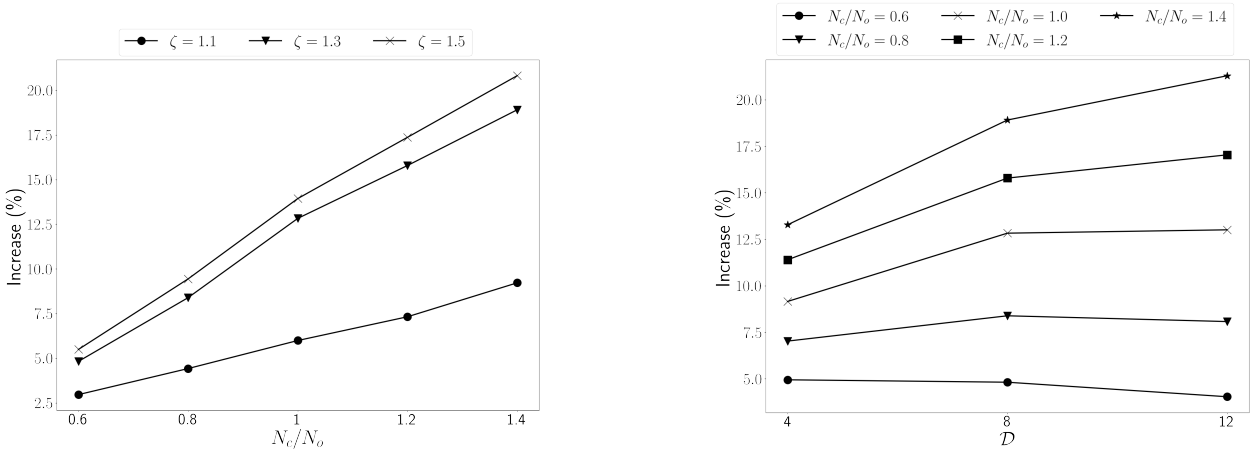


Figure 6 Average assigned orders per crowd-shipper through the operation horizon for ADP and Myopic.

average assigned orders per crowd-shipper in almost all time periods. Also, due to increase in the number of available orders in the two surges of temporal order arrival curve in Figure 1, both ADP and Myopic have an increase in the average assigned orders per crowd-shipper. We also observe that the difference between results of the two policies is much higher in the periods of the two surges of crowd-shippers arrival (see Figure 1). By anticipating the surge in availability of crowd-shippers, ADP postpones some orders to these time periods, which in turn increases the number of available orders for crowd-shippers and the average assigned orders per crowd-shippers.

Figure 7 presents the percentage increase in average assigned orders per crowd-shipper by ADP compared to Myopic with a sensitivity analysis on crowd-shipper deviation range, ζ, maximum delivery deadline, D, and the total crowd-shippers to total orders ratio, N_c/N_o. Figure 7a indicates that by having more flexible crowd-shippers or more available crowd-shippers, ADP yields a higher percentage increase in average number of assigned orders per crowd-shipper compared to Myopic. In contrast, Figure 7b shows a mixed trend on the effect of D. We observe that, for $N_c/N_o = 0.6$, an increase in D leads to a slight decrease in relative gain of ADP in increasing the average assigned orders per crowd-shippers. However, this trend starts to reverse as N_c/N_o increases.

One of the important aspects in participation of in-store customers in crowd-shipping is the amount of distance deviation they incur for delivering customer packages (Archetti, Savelsbergh, and Speranza 2016). Rationally, lower distance deviation usually leads to higher willingness to participate. From the transportation network viewpoint, lower distance deviation reduces the externalities such as traffic congestion and greenhouse gas emission on the transportation network. Our



(a) Analysis on ζ and N_c/N_o.

(b) Analysis on D and N_c/N_o.

Figure 7 Percentage increase in average assigned orders per crowd-shipper by `ADP`.

analysis shows that `ADP` has an exceptional performance in reducing crowd-shippers distance deviation making the policy more sustainable for our last-mile delivery operation. Figure 8 presents an analysis of `ADP` average distance deviation per crowd-shipper and its percentage saving compared to `Myopic`, with respect to ζ and N_c/N_o. As expected, an increase in the crowd-shipper deviation

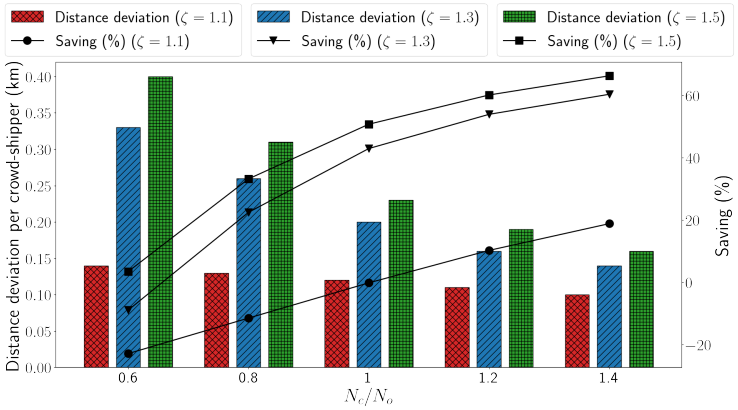


Figure 8 Average distance deviation per crowd-shipper of `ADP` and the percentage saving compared to `Myopic` for different levels of ζ and N_c/N_o.

range results in a higher average distance deviation per crowd-shipper, also higher availability of crowd-shippers lowers average distance deviation per crowd-shipper. We observe that `ADP` percentage saving in crowd-shipper deviation ranges from -22.9 % (for $N_c/N_o = 0.6$ and $\zeta = 1.1$) to 66.2 % (for $N_c/N_o = 1.4$ and $\zeta = 1.5$). For the low value of crowd-shipper deviation range (e.g., $\zeta = 1.1$) and the number of available crowd-shippers (e.g., $N_c/N_o = 0.6$), the operation is under stress for serving orders. Compared to `Myopic`, `ADP` increases the distance deviation of crowd-shippers to serve the orders that require higher distance deviation by crowd-shippers. For higher values

of ζ and N_c/N_o, ADP decreases the average distance deviation per crowd-shipper to save on the crowd-shippers' compensation. As future research, it would be interesting to incorporate the effect of distance deviation on the acceptance and rejection of the assigned orders.

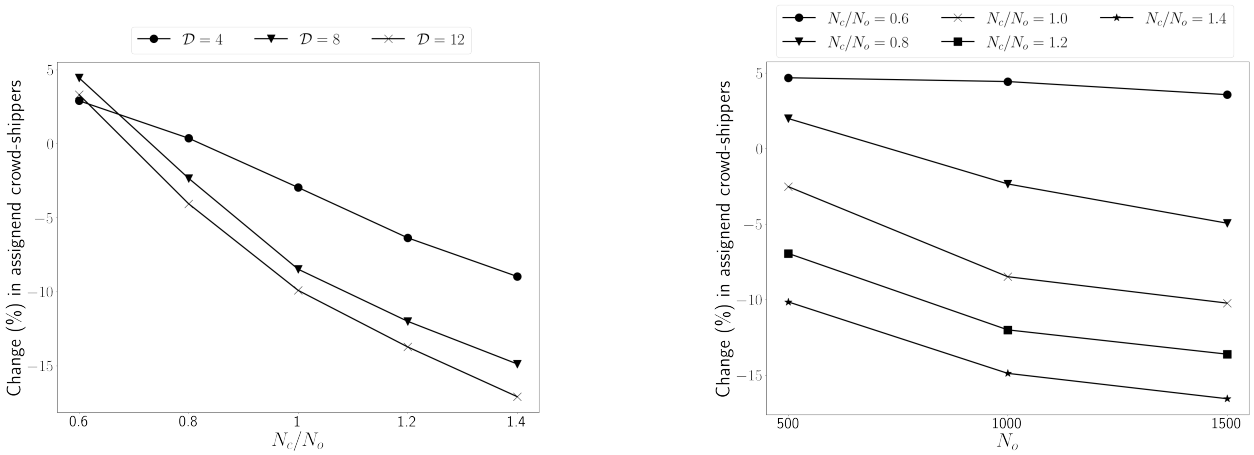
6.6. Served Order and Assigned Crowd-shipper Analyses

Since our model solely depends on in-store customers, the model analysis on the percentage of served orders, and ADP advantage in increasing number of served orders are cardinal from managerial perspective. Table 3 presents the percentage of served orders with ADP and its relative

Table 3 Percentage of orders served by ADP and percentage improvement compared to Myopic.

		D=4		D=8		D=12	
N_o	N_c/N_o	Served (%)	Improvement (%) (vs. Myopic)	Served (%)	Improvement (%) (vs. Myopic)	Served (%)	Improvement (%) (vs. Myopic)
500	0.6	60.6	6.0	75.7	7.7	81.8	6.9
	0.8	71.2	6.4	85.1	6.7	90.2	5.1
	1	78.2	5.6	90.1	4.6	94.1	2.9
	1.2	82.9	4.7	92.7	2.9	96.0	1.6
	1.4	86.3	3.7	94.4	1.9	96.7	1.0
1000	0.6	72.7	8.0	86.2	9.5	89.9	7.5
	0.8	82.8	7.4	93.0	5.9	95.6	3.7
	1	88.1	5.9	95.7	3.3	97.5	1.8
	1.2	91.6	4.3	97.1	1.9	98.4	1.0
	1.4	93.5	3.1	97.9	1.2	99.1	0.6
1500	0.6	78.8	9.2	90.4	9.8	93.1	7.3
	0.8	88.2	8.0	95.3	4.9	97.1	3.0
	1	92.3	5.7	97.4	2.5	98.6	1.3
	1.2	94.7	3.7	98.3	1.4	99.1	0.7
	1.4	96.0	2.4	98.8	0.9	99.4	0.4

improvement in number of served orders (compared to Myopic) with the sensitivity analysis on N_o, N_c/N_o and D. The percentage of served orders ranges from 60.6 % to 99.4 % with average value of 90.5 %. We observe that increasing the number of orders or delivery deadlines results in significant gains in percentage of served orders by ADP. Furthermore, Table 3 indicates that ADP increases the number of served orders up to 9.8% compared to Myopic. In this regard, ADP performance is notable when the system is under stress, i.e., when N_c/N_o and/or D are small. ADP can efficiently prioritize serving the orders to increase total number of served orders and avoid cost of not serving. Another important aspect from a managerial perspective is guaranteeing a certain level of service. We think that the simulation results can be used to identify orders that are likely being missed due to their location, requested delivery deadline, or time of day. Therefore, the system can automatically refuse to serve those online orders with crowd-shipping in order to avoid unpleasant experience for e-shoppers. We leave the investigation of this aspect for future research.



(a) Analysis on D and N_c/N_o.

(b) Analysis on N_o and N_c/N_o.

Figure 9 Percentage change in number of assigned crowd-shippers for ADP.

Figure 9 presents an analysis of the percentage change in the number of assigned (i.e., used) crowd-shippers throughout the operation horizon by ADP compared to Myopic. The results show that when the availability of crowd-shippers is low or the delivery deadline is short, ADP maximizes the use of crowd-shippers by increasing the number of assigned crowd-shippers compared to Myopic. In contrast, for higher availability of crowd-shippers, ADP decreases the number of assigned crowd-shippers by opportunity to more selectively choose crowd-shippers that have possibility of being matched with more orders due to their higher capacity or destination location. Moreover, with increase in N_o, ADP results in decrease in the number of assigned crowd-shippers compared to Myopic due to an increase in averaged assigned orders per crowd-shipper. The decrease in the number of assigned crowd-shippers raises concern regarding the willingness of crowd-shippers to participate in the delivery operation. However, many of the potential crowd-shippers are not beneficial to the operation due to their destination location or capacity. Thus, losing those crowd-shippers in the long term might not affect the quality of operation significantly. Also, assigning orders to crowd-shippers that are more beneficial to the operation may potentially foster the future participation of those crowd-shippers in the delivery operation. As a future research direction, the presented optimization model in this study can be a basis to incorporate the effect of crowd-shippers assignment likelihood on the number of available crowd-shippers in the delivery operation.

6.7. Order Postponement Analyses

One last important aspect in this operation is to know the time it takes to serve orders. Figure 10 shows the average postponement (in terms of time periods) for served orders in ADP and Myopic. As expected, Myopic takes fewer time periods to serve the orders. Furthermore, by increasing N_c/N_o the average postponement for serving orders starts to decrease. However, this decrease is steeper

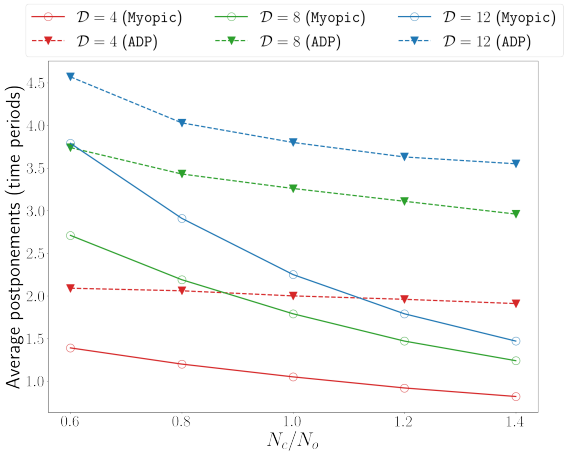


Figure 10 **Comparison of ADP and Myopic in average number of postponements for served orders.**

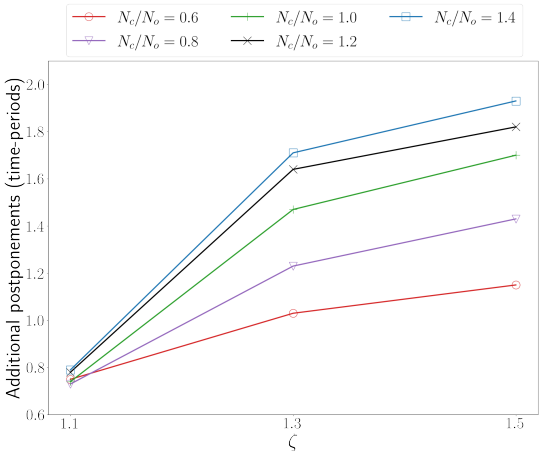


Figure 11 **Impact of crowd-shippers range on ADP additional number of postponement of served orders.**

for Myopic resulting in a larger difference between ADP and Myopic. ADP exploits the opportunity of higher number of crowd-shippers to additionally postpone the orders in order to match them with cost-efficient crowd-shippers. Nevertheless, the average additional postponement is low compared to the length of delivery deadlines. Figure 11 provides an analysis of the number of additional postponement by ADP with respect to ζ and N_c/N_o. Similarly, a higher value of N_c/N_o or higher ζ results in an increase in the average number of postponements of ADP. To conclude, for at most two time periods additional postponements, ADP can obtain significant cost-savings compared to Myopic. This observation is vital for the operation manager who might be concerned about the number of additional postponements by ADP.

7. Conclusion

In this paper, we studied a variant of dynamic crowd-shipping problem with in-store customers, and formulated it as an MDP. In addition to an optimization-based myopic policy, we proposed an ADP policy based on value function approximation of online orders to capture the downstream uncertainty in arrival of online orders and crowd-shippers. This method helps real-time decision making while considering multiple attributes for crowd-shippers and customers without any restrictions for the stochastic process. We also incorporated several enhancements in the ADP algorithm.

Our test instances are based on the City of Toronto where we simulate the proposed crowd-shipping operation for 14 hours. We showed the efficiency and efficacy of the proposed ADP algorithm and incorporated several enhancements. We performed a comprehensive sensitivity analysis on model parameters, and showed that the ADP policy for online decision-making is superior to the Myopic policy under different parameter configurations. Specifically, depending on the instance parameters, the ADP policy can result in up to 25.2% saving in operational cost, 9.8% increase

in number of served orders, 66.2% increase in distance deviation per crowd-shipper, and 21.1% increase in average number of orders assigned per crowd-shipper. We also showed that the ADP policy intelligently makes decision depending on the characteristics of the system.

This research is a prologue to several new research directions in real-time decision making in crowd-shipping. First, the considered setting can be extended to multiple stores for fulfilment of the orders, as well as multiple intermediate location visits by crowd-shippers with a proper routing model. Second, other types of uncertainties can be incorporated. The presented model in this study can be a basis for implementing a dynamic pricing strategy by considering compensation to crowd-shippers and delivery fee for e-shoppers. Lastly, the model and the ADP algorithm can be extended to a dynamic pickup and delivery crowd-shipping operation.

Acknowledgments

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC). We also acknowledge Compute Ontario (www.computeontario.ca), Calcul Québec (www.calculquebec.ca), WestGrid (www.westgrid.ca), and Compute Canada (www.computecanada.ca) for computational resources.

References

Al-Kanj L, Nascimento J, Powell WB, 2020 *Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. Eur. J. Oper. Res.* 284(3):1088–1106.

Allahviranloo M, Baghestani A, 2019 *A dynamic crowdshipping model and daily travel behavior. Transp. Res. E Logist. Transp. Rev.* 128:175–190.

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

Amazonca, 2021 *Same-day and one-day delivery rates.* https://www.amazon.ca/gp/help/customer/display.html?nodeId=GTFUTRQF459U3H3Y, accessed 2021-06-08.

Archetti C, Savelsbergh M, Speranza MG, 2016 *The vehicle routing problem with occasional drivers. Eur. J. Oper. Res.* 254(2):472–480.

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

Boysen N, Emde S, Schwerdfeger S, 2021 *Crowdshipping by employees of distribution centers: Optimization approaches for matching supply and demand. Eur. J. Oper. Res.* 296(2):539–556.

Cleophas C, Cottrill C, Ehmke JF, Tierney K, 2019 *Collaborative urban transportation: Recent advances in theory and practice. Eur. J. Oper. Res.* 273(3):801–816.

Computecanada, 2021 *Available resources for RAC 2022.* https://www.computecanada.ca/research-portal/accessing-resources/available-resources/.

Dahle L, Andersson H, Christiansen M, 2017 *The vehicle routing problem with dynamic occasional drivers. Int. Conf. Comput. Logist.*, 49–63 (Springer).

Dahle L, Andersson H, Christiansen M, Speranza MG, 2019 *The pickup and delivery problem with time windows and occasional drivers. Comput. and Oper. Res.* 109:122–133.

Dai H, Liu P, 2020 *Workforce planning for O2O delivery systems with crowdsourced drivers. Ann. Oper. Res.* 291(1):219–245.

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

DMG, 2018 *2016 TTS: Design and conduct of the survey.* http://dmg.utoronto.ca/transportation-tomorrow-survey/tts-reports.

Gdowska K, Viana A, Pedroso JP, 2018 *Stochastic last-mile delivery with crowdshipping. Transp. Res. Proc.* 30:90–100.

George A, Powell WB, Kualkarni SR, 2008 *Value function approximation using multiple aggregation for multiattribute resource management. J Mach Learn Res.* 9(10).

George AP, Powell WB, 2006 *Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. Mach. Learn.* 65(1):167–198.

Jiang DR, Powell WB, 2015 *An approximate dynamic programming algorithm for monotone value functions. Oper. Res.* 63(6):1489–1511.

Kafle N, Zou B, Lin J, 2017 *Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. Transportation Res. Part B: Methodological* 99:62–82.

Mes MR, Powell WB, Frazier PI, 2011 *Hierarchical knowledge gradient for sequential sampling. J Mach Learn Res.* 12(10).

Mousavi K, Bodur M, Roorda MJ, 2020 *Stochastic last-mile delivery with crowd-shipping and mobile depots* http://www.optimization-online.org/DB_FILE/2020/02/7619.pdf.

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

Raviv T, Tenzer EZ, 2018 *Crowd-shipping of small parcels in a physical internet.* https://www.researchgate.net/publication/326319843_Crowd-shipping_of_small_parcels_in_a_physical_internet.

Simao HP, Day J, George AP, Gifford T, Nienow J, Powell WB, 2009 *An approximate dynamic programming algorithm for large-scale fleet management: A case application. Transportation Sci.* 43(2):178–197.

Skålnes J, Dahle L, Andersson H, Christiansen M, Hvattum LM, 2020 *The multistage stochastic vehicle routing problem with dynamic occasional drivers. Int. Conf. Comput. Logist.*, 261–276 (Springer).

Spivey MZ, Powell WB, 2004 *The dynamic assignment problem. Transportation Sci.* 38(4):399–419.

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

Walmart Inc, 2020 *Walmart introduces express delivery.* https://corporate.walmart.com/newsroom/2020/04/30/walmart-introduces-express-delivery, accessed 2021-06-08.

Wolsey LA, Nemhauser GL, 1999 *Integer and combinatorial optimization*, volume 55 (John Wiley & Sons).

World Economic Forum, 2021 *Covid-19 pandemic accelerated shift to e-commerce by 5 years, new report says.* https://www.weforum.org/agenda/2020/08/covid19-pandemic-social-shift-ecommerce-report/, accessed 2021-06-12.

Yıldız B, 2021 *Express package routing problem with occasional couriers. Transportation Research Part C: Emerging Technologies* 123:102994.

Zebra, 2019 *Reinventing the supply chain:the future of fulfillment vision study.* https://www.zebra.com/content/dam/zebra_new_ia/en-us/solutions-verticals/vertical-solutions/retail/vision-study/fulfillment-vision-study-report-en-us.pdf.

Appendix A: MDP Model Notation Summary

Table 4 Summary of notation introduced in Section 4

Sets and indices:	
$\mathcal{T}$	Decision epochs in the planning horizon, $t \in \mathcal{T} := \{1, 2, \ldots, T\}$
$\mathcal{L}$	Possible locations in the operational area, $\ell \in \mathcal{L} := \{0, 1, \ldots, L\}$; the store is located at $\ell = 0$
$\mathcal{Q}$	Possible capacities for crowd-shippers, $q \in \mathcal{Q} := \{1, 2, \ldots, Q\}$
$\mathcal{D}$	Possible order dispatch deadlines, numbers of epochs left to dispatch order from the depot directly to its location for delivery on time, $d \in \mathcal{D} := \{0, 1, \ldots, D\}$
$\mathcal{C}$	Possible attribute vectors for crowd-shippers, $c = (c_{\texttt{loc}}, c_{\texttt{cap}}) \in \mathcal{C} := \mathcal{L} \times \mathcal{Q}$
$\mathcal{O}$	Possible attribute vectors for online orders, $o = (o_{\texttt{loc}}, o_{\texttt{due}}) \in \mathcal{O} := \mathcal{L} \times \mathcal{D}$
Π	Set of all possible policies, $\pi \in \Pi$
$\mathcal{L}_c^{\texttt{Elig}}$	Locations that are eligible for crowd-shippers with attribute vector c based on distance deviation, defined in (2)
$\mathcal{O}_c^{\texttt{Elig}}$	Order attribute vectors with eligible location for crowd-shippers with attribute vector c, defined in (3)
$\mathcal{O}_c^{\texttt{SameLoc}}$	Order attribute vectors with the same location as the one in crowd-shipper attribute c, defined in (4)
$\mathcal{O}_\ell^{\texttt{SameLoc}}$	Order attribute vectors with location ℓ, defined in (4)
$\mathcal{L}_{co}^{\texttt{EligMid}}$	Eligible locations for crowd-shippers with attribute vector c to serve orders as a middle stop before serving orders with attribute vector o on time, defined in (5)
$\mathcal{A}^t$	Set of feasible actions at decision epoch t, $\boldsymbol{a}^t \in \mathcal{A}^t$
Parameters:	
δ	Duration of time intervals, i.e., time between consecutive decision epochs
$f_o^{\texttt{NotServ}}$	Cost of not serving an order with attribute vector o by its deadline
$f_o^{\texttt{Fix}}$	Fixed compensation cost paid to crowd-shippers for serving an order with attribute vector o
$f^{\texttt{Dev}}$	Crowd-shipper compensation rate for each unit of distance deviation from their original route
$\Delta_{c\ell}$	Distance deviation of crowd-shipper with attribute vector c visiting location ℓ, defined in (**??**)
ζ	Crowd-shippers flexibility for distance deviation, maximum allowed multiplicative deviation ratio
μ	Service time at an order location
$\texttt{dist}(\ell, \ell')$	Distance between locations ℓ and ℓ'
$\texttt{time}(\ell, \ell')$	Travel time between locations ℓ and ℓ'
State variables at decision epoch t:	
$S_{tc}^{\texttt{Crowd}}$	Number of crowd-shippers with attribute vector c
$S_{to}^{\texttt{Order}}$	Number of orders with attribute vector o
Random variables associated with decision epoch t:	
$W_{tc}^{\texttt{Crowd}}$	Number of crowd-shipper arrivals with attribute vector c
$W_{to}^{\texttt{Order}}$	Number of order arrivals with attribute vector o
Decision variables at decision epoch t:	
$x_{c\ell}^t$	Number of crowd-shippers with attribute vector c assigned to orders with delivery location ℓ
y_{co}^t	Number of orders with attribute o served by crowd-shippers with attribute c
$z_{c\ell o}^t$	Number of orders with attribute o served by crowd-shippers with the same destination (i.e., $c_{\texttt{loc}} = o_{\texttt{loc}}$), while the crowd-shipper had a stop to serve orders at another location ℓ
p_o^t	Number of orders with attribute vector o that are postponed to the next decision epoch
e_o^t	An auxiliary binary variable that takes 1 if an order with attribute o is postponed to the next epoch, 0 otherwise

Appendix B: Hierarchical Aggregation Algorithm

Algorithm 2 provides the algorithmic procedure for HA with BAKF step-size. In order to incorporate the

Algorithm 2 Hierarchical aggregation algorithm with BAKF step-size

Step 0: Initialize the parameters $\bar{\beta}^{g,0}_{t-1,o^g}=0$, $\bar{\bar{\beta}}^{g,0}_{t-1,o^g}=0$, and $\lambda^{g,0}_{t-1,o^g}=0\ \forall g\in\mathcal{G}$, $\forall o^g\in\mathcal{O}^g$, and $\eta^0=1$

Step 1: Obtain the observed post-decision value functions $\vartheta^{\text{Post},n}_{t-1}$ from epoch t in Step 3.b of Algorithm 1

Step 2: For each attribute $o^g\in\{g\in\mathcal{G},\, o^g\in\mathcal{O}^g : |H^{g,n}_{t-1,o^g}|\neq\emptyset\}$ obtain $\vartheta^{\text{Post},(g,n)}_{t-1,o^g}=\dfrac{\sum_{o\in H^{g,n}_{t-1,o^g}}\vartheta^{\text{Post},n}_{t-1,o}}{|H^{g,n}_{t-1,o^g}|}$, and do:

Step 2.a: Update the bias of transient data series, $\bar{\beta}$, and total squared variation, $\bar{\bar{\beta}}$, as

$$\bar{\beta}^{g,n}_{t-1,o^g}=(1-\eta^{n-1})\bar{\beta}^{g,n-1}_{t-1,o^g}+\eta^{n-1}(\vartheta^{\text{Post},(g,n)}_{t-1,o^g}-\bar{v}^{\text{Post},(g,n-1)}_{t-1,o^g}) \tag{29}$$

$$\bar{\bar{\beta}}^{g,n}_{t-1,o^g}=(1-\eta^{n-1})\bar{\bar{\beta}}^{g,n-1}_{t-1,o^g}+\eta^{n-1}(\vartheta^{\text{Post},(g,n)}_{t-1,o^g}-\bar{v}^{\text{Post},(g,n-1)}_{t-1,o^g})^2 \tag{30}$$

Update η, using the McClain's step-size formula with $\bar{\eta}=0.1$:

$$\eta_n=\frac{\eta^{n-1}}{1+\eta^{n-1}-\bar{\eta}} \tag{31}$$

Step 2.b: Update variance of observations, $(s^2_{t-1,o^g})^{g,n}$, using:

$$(s^2_{t-1,o^g})^{g,n}=\frac{\bar{\bar{\beta}}^{g,n}_{t-1,o^g}-(\bar{\beta}^{g,n}_{t-1,o^g})^2}{1+\lambda^{(g,n-1)}_{t-1,o^g}} \tag{32}$$

Step 2.c: Update the $\alpha^{g,n-1}_{t-1,o^g}$ using the BAKF step-size:

$$\alpha^{g,n}_{t-1,o^g}=\begin{cases}\dfrac{1}{n+1} & n=1,2\\ 1-\dfrac{(s^2_{t-1,o^g})^{g,n}}{(1+\lambda^{(g,n-1)}_{t-1,o^g})(s^2_{t-1,o^g})^{g,n}+(\bar{\beta}^{g,n}_{t-1,o^g})^2} & n>2\end{cases} \tag{33}$$

Step 2.d: Update post-decision value function estimations:

$$\bar{v}^{\text{Post},(g,n)}_{t-1,o^g}=(1-\alpha^{g,n}_{t-1,o^g})\bar{v}^{\text{Post},(g,n-1)}_{t-1,o^g}+\alpha^{g,n}_{t-1,o^g}\vartheta^{\text{Post},(g,n)}_{t-1,o^g} \tag{34}$$

Step 2.e: Update $\lambda^{g,n}_{t-1,o^g}$ using:

$$\lambda^{g,n}_{t-1,o^g}=\begin{cases}(\alpha^{g,n-1}_{t-1,o^g})^2 & n=1\\ (1-\alpha^{g,n-1}_{t-1,o^g})^2\lambda^{g,n-1}_{t-1,o^g}+(\alpha^{g,n-1}_{t-1,o^g})^2 & n>1\end{cases} \tag{35}$$

Step 2.f: Update the variance of value function estimates, $(\bar{\sigma}^2_{t-1,o^g})^{g,n}$, using:

$$(\bar{\sigma}^2_{t-1,o^g})^{g,n}=Var[\bar{v}^{g,n}_{t-1,o^g}]=\lambda^{g,n}_{t-1,o^g}(s^2_{t-1,o^g})^{g,n} \tag{36}$$

Step 3: For each $o\in\mathcal{O}, g\in\mathcal{G}$, compute the bias of aggregation, $\bar{\mu}$, as

$$\bar{\mu}^{g,n}_{t-1,o}=\bar{v}^{\text{Post},(g,n)}_{t-1,M^g(o)}-\bar{v}^{\text{Post},(0,n)}_{t-1,o} \tag{37}$$

Step 4: For each $o\in\mathcal{O}$, $g\in\mathcal{G}$, compute weights based on

$$w^g_{t-1,o}=\frac{(\bar{\sigma}^2_{t-1,M^g(o)})^{g,n}+(\bar{\mu}^{g,n}_{t-1,o})^2}{\sum_{g'\in\mathcal{G}}(\bar{\sigma}^2_{t,M^{g'}(o)})^{g',n}+(\bar{\mu}^{g',n}_{t-1,o})^2} \tag{38}$$

Step 5: For each $o\in\mathcal{O}$, compute the improved post-decision value function estimates, $\bar{v}^{\text{Post},n}_{t-1o}=\sum_{g\in\mathcal{G}}w^g_{t-1,o}.\bar{v}^{\text{Post},(g,n)}_{t-1,M^g(o)}$
Return the values $\bar{v}^{\text{Post},n}_{t-1o}$ to use in the next iteration of Algorithm 1

HA algorithm in the forward-pass ADP algorithm, we require to implement Algorithm 2 in Step 3.b of

Algorithm 1. As an input, Algorithm 2 (in its Step 1) obtains the observed value functions of post-decision states in the disaggregated level, $\vartheta^{\texttt{Post},n}_{t-1,o}$ from Step 3.b of Algorithm 1 at epoch t. In Step 2 of Algorithm 2, we require to compute the observed aggregated post-decision value functions for all aggregation level. For observed order attributes in all levels of aggregations, we update the estimate of post-decision value functions, and variance of the value function estimates in Steps 2.d and 2.f of Algorithm 2, respectively. For updating post-decision value function estimates, we use the BAKF step-size based on Equation (33) in Step 2.c. Due to instability of our initial estimates of bias of transient, and variance of observations, we adopt harmonic step-size in the first two iterations. In Step 3, for each disaggregated order attributes, we compute the value functions aggregation bias for each aggregation level. Finally, we compute aggregation weights using (38), and weighted estimation of post-decision value functions using (25), in Steps 4 and 5, respectively.

Appendix C: Instance Characteristics

Figure 12 shows the study area, store location, and the online order arrival spatial heat map for total online orders of $N_o = 1000$, whereas Figure 13 provides the hierarchical aggregation of the locations.

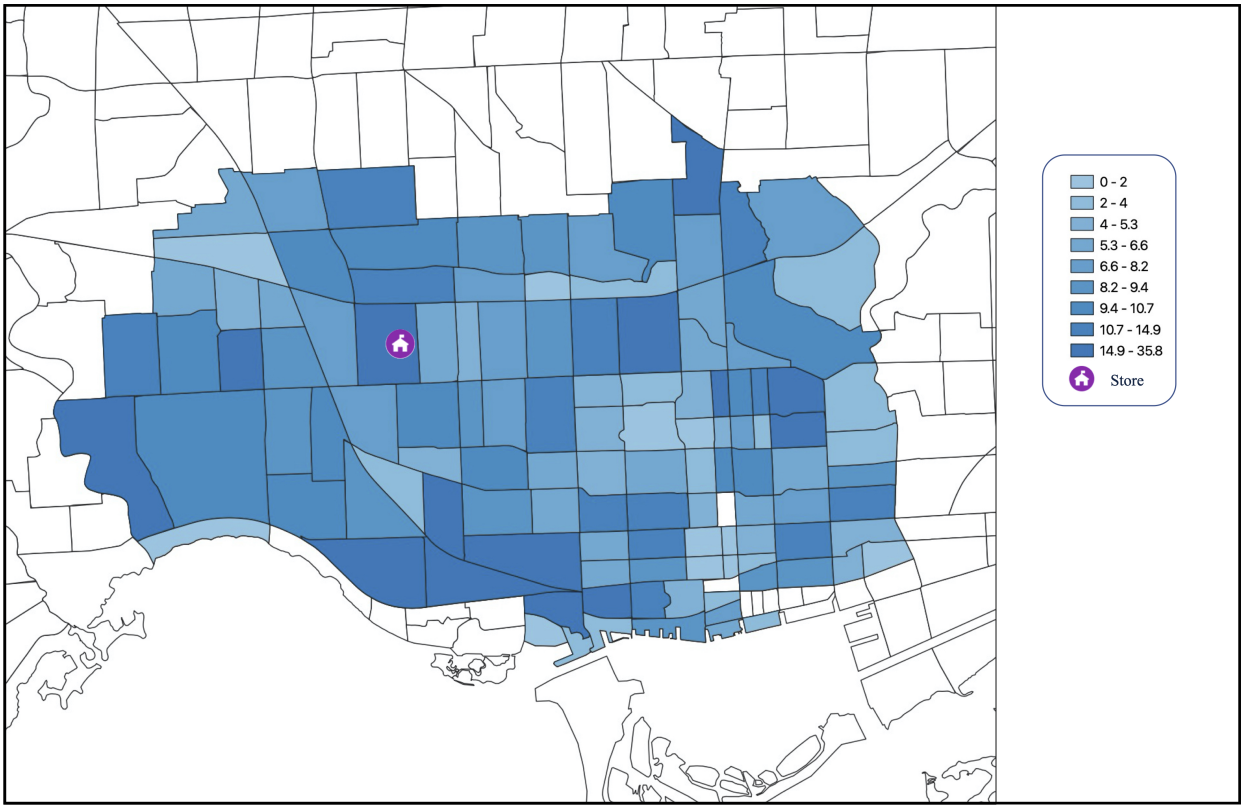


Figure 12 **The study area and total online order arrivals per location for $N_o = 1000$.**

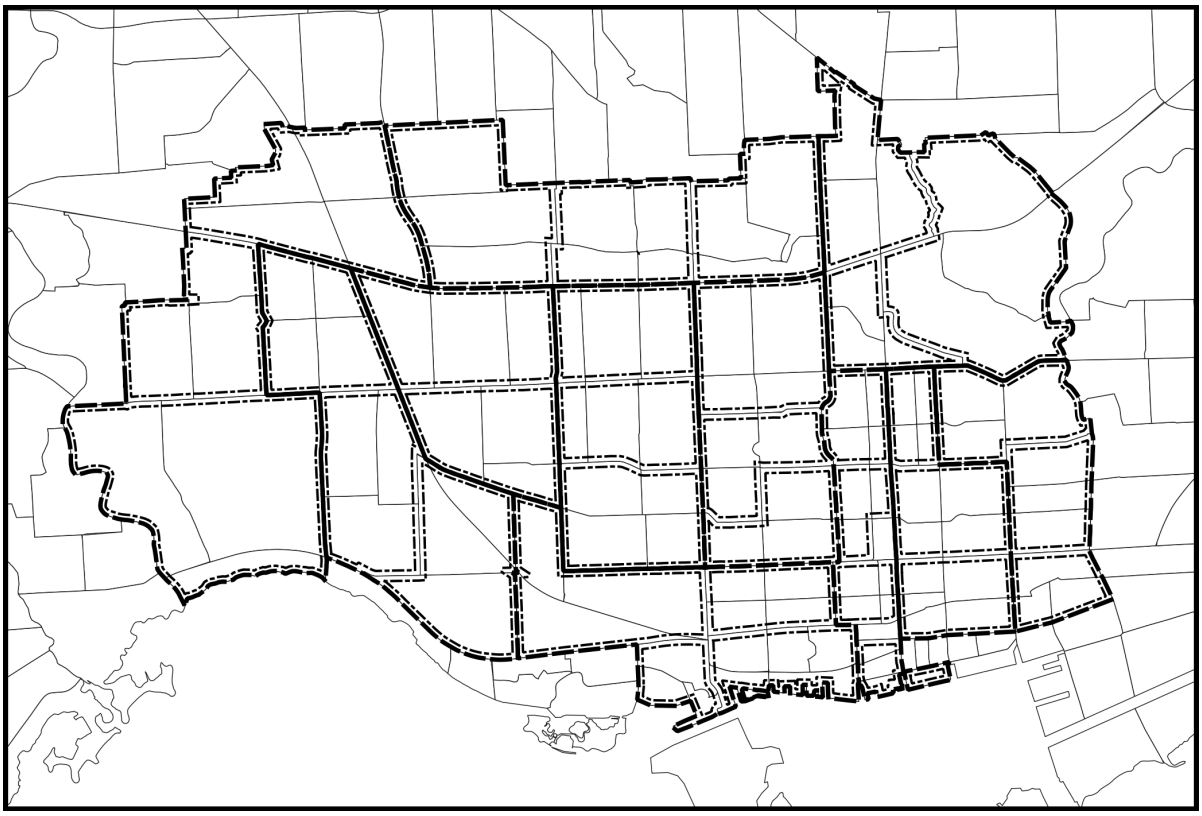


Figure 13 **The spatial boundaries for each level of aggregation; continuous lines, dash-dotted lines, and dashed lines correspond to disaggregated level, aggregation level 1, and aggregation level 2, respectively.)**

Table 5 shows the summary of all parameters for the computational experiments in Section 6.

Table 5 **List of parameters and their values for computational experiments.**

	Description	Values
Instance parameters:		
L	Number of locations	117
N_o	Total order arrivals	$\in \{500, 1000, 1500\}$
N_c/N_o	Ratio of total crowd-shipper arrivals to total order arrivals	$\in \{0.6, 0.8, 1, 1.2, 1.4\}$
D	Delivery deadline (time periods)	$\in \{4, 8, 12\}$
$f^{\texttt{Fix}}$	Fixed compensation cost per package	$\in \{2, 3, 4\}$
$f^{\texttt{Dev}}$	Deviation compensation rate per unit of distance deviation	$\in \{2, 3, 4\}$
$f^{\texttt{NotServ}}$	Cost of not serving orders	10
ζ	Crowd-shipper deviation range	$\in \{1.1, 1.3, 1.5\}$
Q	Crowd-shippers capacity.	$\in \{1, 2, 3, 4\}$
T	Number of decision epochs	52
δ	Length of each decision epoch (min)	15
μ	Delivery service time at each location (min)	15
$Speed$	Network speed (km/h)	20
ADP algorithm and simulation parameters:		
$\|\mathcal{G}\|$	Number of aggregation levels	3
N	Number of ADP algorithm iterations	1000
$\|\Omega\|$	Total number of simulations for policy evaluation	1000

Appendix D: Detailed Numerical Results

Table 6 **Analysis on percentage of cost-saving by ADP with crowd-shippers deviation range of $\zeta = 1.3$.**

			$D=4$					$D=8$					$D=12$				
$N_c/N_o \longrightarrow$			0.6	0.8	1	1.2	1.4	0.6	0.8	1	1.2	1.4	0.6	0.8	1	1.2	1.4
N_o	f^{Fix}	f^{Dev}															
500	2	2	4.92	7.50	9.38	10.66	11.45	9.03	12.00	13.14	13.62	13.75	9.65	11.22	11.70	12.86	13.94
		3	4.43	6.95	8.92	10.51	11.70	8.17	11.38	13.60	15.19	16.40	8.80	11.08	13.28	15.76	17.77
		4	4.20	6.68	8.80	10.62	12.09	7.74	11.29	14.18	16.54	18.35	8.34	11.50	14.78	18.07	20.62
	3	2	3.81	5.68	6.98	7.89	8.41	6.70	8.63	9.42	9.82	10.06	6.93	7.91	8.38	9.31	10.23
		3	3.47	5.34	6.79	7.94	8.84	6.15	8.45	10.01	11.30	12.20	6.40	8.14	9.81	11.73	13.21
		4	3.34	5.22	6.82	8.13	9.21	5.93	8.50	10.60	12.34	13.70	6.21	8.64	10.99	13.48	15.31
	4	2	2.90	4.28	5.24	5.90	6.31	4.93	6.35	7.00	7.44	7.71	5.02	5.74	6.30	7.16	7.92
		3	2.70	4.11	5.23	6.14	6.80	4.61	6.34	7.65	8.68	9.48	4.74	6.14	7.58	9.09	10.31
		4	2.62	4.01	5.19	6.11	6.88	4.49	6.36	7.96	9.27	10.29	4.66	6.50	8.37	10.22	11.54
1000	2	2	8.93	12.24	13.70	13.95	13.72	14.64	14.89	14.36	14.08	14.31	13.32	11.85	12.27	13.19	14.24
		3	8.38	11.72	13.64	14.60	15.05	13.51	15.33	16.26	17.04	18.00	12.31	12.97	15.17	17.10	18.68
		4	8.01	11.44	13.71	15.16	16.02	12.93	15.75	17.70	19.28	20.60	11.75	14.15	17.41	20.00	21.90
	3	2	6.78	9.01	9.90	10.01	9.79	10.52	10.48	10.10	10.03	10.19	9.27	8.27	8.66	9.44	10.15
		3	6.38	8.73	10.03	10.68	11.00	9.78	10.96	11.70	12.36	12.99	8.68	9.31	10.92	12.34	13.47
		4	6.17	8.59	10.17	11.15	11.74	9.46	11.45	12.87	14.01	14.91	8.51	10.33	12.63	14.47	15.77
	4	2	5.10	6.68	7.31	7.39	7.29	7.58	7.65	7.51	7.55	7.76	6.55	6.00	6.54	7.22	7.80
		3	4.84	6.55	7.53	8.03	8.30	7.15	8.17	8.83	9.38	9.88	6.24	7.01	8.36	9.49	10.33
		4	4.69	6.42	7.56	8.25	8.69	6.96	8.47	9.58	10.42	11.11	6.22	7.01	9.44	10.86	11.78
1500	2	2	12.00	15.30	15.54	14.53	13.60	17.06	15.01	13.81	13.41	13.43	14.25	11.47	11.85	12.60	13.05
		3	11.32	14.85	15.89	15.77	15.62	16.17	15.98	16.24	16.64	17.29	13.50	13.13	15.06	16.62	17.40
		4	10.90	14.58	16.15	16.67	17.04	15.60	16.91	18.15	19.18	20.12	13.19	14.68	17.46	19.49	20.64
	3	2	8.94	11.04	11.01	10.21	9.56	12.07	10.44	9.67	9.36	9.44	9.77	7.95	8.33	8.89	9.21
		3	8.50	10.83	11.44	11.29	11.16	11.58	11.34	11.55	11.87	12.29	9.41	9.32	10.70	11.81	12.39
		4	8.23	10.72	11.74	12.00	12.19	11.26	12.11	12.97	13.66	14.27	9.32	10.46	12.48	13.83	14.67
	4	2	6.65	8.06	8.02	7.52	7.10	8.67	7.57	7.15	7.05	7.15	6.91	5.81	6.28	6.78	7.03
		3	6.35	7.99	8.45	8.40	8.36	8.35	8.39	8.67	8.95	9.29	6.74	6.98	8.11	8.96	9.42
		4	6.15	7.89	8.63	8.84	8.99	8.19	8.94	9.61	10.17	10.63	6.76	7.83	9.32	10.38	10.95

Table 7 **Impact of ADP on postponement of orders for $N_o = 1000$.**

	$D=4$			$D=8$									$D=12$		
$\zeta \longrightarrow$	1.3			1.1			1.3			1.5			1.3		
N_c/N_o	Myopic	ADP	Difference	Myopic	ADP	Difference	Myopic	ADP	Difference	Myopic	ADP	Difference	Myopic	ADP	Difference
0.6	2.6	1.9	0.7	5.0	4.2	0.8	5.3	4.3	1.0	5.4	4.3	1.2	8.2	7.4	0.8
0.8	2.8	1.9	0.9	5.4	4.6	0.7	5.8	4.6	1.2	6.0	4.5	1.4	9.1	8.0	1.1
1.0	3.0	2.0	1.0	5.7	4.9	0.7	6.2	4.7	1.5	6.4	4.7	1.7	9.7	8.2	1.5
1.2	3.1	2.0	1.0	5.9	5.1	0.8	6.5	4.9	1.6	6.7	4.9	1.8	10.2	8.4	1.8
1.4	3.2	2.1	1.1	6.1	5.4	0.8	6.8	5.0	1.7	6.9	5.0	1.9	10.5	8.4	2.1

Table 8 Summary of cost breakdown and the saving ADP on each cost components for $N_o = 1000$ and $f^{\texttt{Fix}} = 3$.

ζ	N_c/N_o	$f^{\texttt{Dev}}$	Total cost	Saving (%)	Fixed cost	Saving (%)	Deviation cost	Saving (%)	Cost of not serving	Saving (%)
1.1	0.6	2	4882.0	6.9	2234.9	-8.1	106.0	-35.7	2541.0	18.0
		3	4935.4	6.6	2231.3	-7.9	151.0	-28.8	2553.1	17.6
		4	4985.3	6.4	2227.0	-7.7	190.9	-22.3	2567.4	17.2
	0.8	2	4249.5	7.4	2518.0	-6.5	121.8	-20.7	1609.6	24.1
		3	4307.2	7.1	2514.8	-6.4	172.0	-13.6	1620.4	23.6
		4	4360.6	7.0	2511.1	-6.2	216.8	-7.4	1632.8	23.1
	1	2	3858.1	7.0	2685.0	-5.0	126.4	-6.3	1046.7	28.8
		3	3914.9	6.9	2682.3	-4.8	177.0	0.9	1055.6	28.1
		4	3967.8	7.0	2679.0	-4.7	222.2	6.6	1066.7	27.4
	1.2	2	3617.8	6.1	2786.1	-3.6	122.8	6.9	708.9	31.3
		3	3671.7	6.3	2783.9	-3.5	171.5	13.4	716.3	30.6
		4	3722.4	6.6	2781.3	-3.4	216.2	18.1	725.0	29.7
	1.4	2	3478.3	5.3	2843.1	-2.6	117.3	16.9	517.9	31.8
		3	3528.0	5.7	2840.7	-3.5	161.4	23.8	526.0	30.7
		4	3575.8	6.2	2838.6	-2.5	204.4	27.6	532.8	29.8
1.3	0.6	2	4223.4	10.5	2605.6	-10.3	312.5	-30.8	1305.2	38.4
		3	4362.6	9.8	2585.1	-9.5	403.7	-13.8	1373.8	35.2
		4	4480.0	9.5	2564.2	-8.7	472.6	-2.7	1443.2	32.2
	0.8	2	3733.7	10.5	2804.1	-6.3	273.4	15.0	656.2	45.8
		3	3850.0	11.0	2790.1	-5.9	357.1	24.2	702.8	42.2
		4	3951.5	11.5	2774.6	-5.4	422.3	29.8	754.6	38.6
	1	2	3498.1	10.1	2878.7	-3.5	218.5	42.8	400.9	45.0
		3	3590.5	11.7	2869.5	-3.3	289.3	47.7	431.7	41.2
		4	3675.8	12.9	2858.8	-3.0	349.8	49.9	467.2	37.4
	1.2	2	3366.6	10.0	2919.1	-2.1	181.9	56.6	265.6	42.7
		3	3443.5	12.4	2913.2	-1.9	245.2	59.4	285.1	38.8
		4	3515.3	14.0	2905.6	-1.8	299.3	60.4	310.4	35.0
	1.4	2	3288.4	10.2	2941.7	-1.4	157.4	64.1	189.3	40.9
		3	3354.9	13.0	2937.0	-1.2	213.2	66.2	204.7	36.7
		4	3418.5	14.9	2930.9	-1.1	262.5	66.6	225.1	32.2
1.5	0.6	2	4106.7	11.3	2701.8	-10.2	420.2	-16.9	984.7	45.9
		3	4275.8	10.8	2665.3	-9.0	504.3	0.2	1106.2	40.0
		4	4415.1	10.5	2630.0	-8.1	561.1	9.1	1224.1	34.9
	0.8	2	3644.8	12.1	2877.3	-5.9	355.2	27.0	412.3	56.3
		3	3786.5	12.9	2847.5	-5.1	427.5	35.8	511.5	47.2
		4	3908.2	13.3	2818.5	-4.5	481.6	39.6	608.0	40.0
	1	2	3434.4	12.5	2931.0	-2.9	276.7	51.7	226.7	54.9
		3	3546.3	14.3	2910.7	-2.5	341.4	55.7	294.2	44.4
		4	3645.6	15.3	2890.0	-2.2	392.3	56.7	363.2	36.5
	1.2	2	3318.1	13.0	2958.3	-1.5	224.9	63.7	134.9	51.7
		3	3412.0	15.5	2943.6	-1.2	284.5	65.6	183.9	38.9
		4	3493.3	16.8	2928.2	-1.1	329.7	65.8	235.4	30.2
	1.4	2	3248.6	13.6	2971.5	-0.8	187.1	70.9	90.0	46.8
		3	3327.5	16.5	2960.3	-0.6	240.1	71.9	127.2	31.8
		4	3398.0	18.0	2948.2	-0.5	282.3	71.6	167.5	22.6

Table 9 Number of served orders and assigned crowd-shippers by ADP and the savings compared to Myopic.

N_o	D	N_c/N_o	# of served orders	% of Served orders	% change in # of served orders	# of served orders at the crowd-shippers destination	% change in # of served orders at the crowd-shippers destination	# of served orders at the intermediate stop of crowd-shippers	% change in # of served orders at the intermediate stop of crowd-shippers	# of assigned crowd-shippers	% change in # of assigned crowd-shippers compared to myopic
500	4	0.6	303.1	60.6	6.0	75.9	21.4	227.2	1.7	188.3	3.69
		0.8	356.2	71.2	6.4	91.7	24.6	264.5	1.2	228.5	2.78
		1	391.0	78.2	5.6	103.4	26.9	287.6	-0.4	256.0	0.69
		1.2	414.3	82.9	4.7	111.3	27.1	303.0	-1.7	275.1	-1.68
		1.4	431.7	86.3	3.7	117.5	27.2	314.2	-3.0	289.3	-4.03
	8	0.6	378.6	75.7	7.7	113.7	24.3	264.9	1.9	204.3	4.69
		0.8	425.7	85.1	6.7	130.7	28.1	295.0	-0.6	240.2	1.99
		1	450.6	90.1	4.6	139.1	29.1	311.6	-3.6	261.0	-2.52
		1.2	463.6	92.7	2.9	145.1	30.8	318.5	-6.2	272.3	-6.94
		1.4	472.1	94.4	1.9	149.3	30.2	322.7	-7.5	280.3	-10.15
	12	0.6	408.9	81.8	6.9	134.2	25.1	274.7	-0.2	208.2	3.96
		0.8	451.1	90.2	5.1	148.1	27.2	303.0	-3.1	242.0	0.70
		1	470.4	94.1	2.9	153.5	27.4	316.9	-5.8	259.4	-4.77
		1.2	480.1	96.0	1.6	157.0	28.1	323.1	-7.7	268.1	-9.52
		1.4	483.5	96.7	1.0	159.7	31.0	323.8	-9.3	274.8	-12.73
1000	4	0.6	727.1	72.71	8.0	222.6	24.7	504.6	2.0	385.8	2.90
		0.8	828.2	82.82	7.4	258.4	26.1	569.8	0.7	453.2	0.37
		1	881.3	88.13	5.9	278.2	25.7	603.1	-1.2	493.5	-2.96
		1.2	916.1	91.61	4.3	291.0	24.3	625.1	-3.0	519.4	-6.36
		1.4	935.1	93.51	3.1	300.6	23.4	634.6	-4.3	534.9	-8.97
	8	0.6	861.7	86.17	9.5	311.6	27.8	550.1	1.2	410.8	4.44
		0.8	930.0	93.00	5.9	333.4	25.9	596.6	-2.8	458.8	-2.34
		1	956.5	95.65	3.3	344.4	25.4	612.1	-6.1	480.9	-8.48
		1.2	971.1	97.11	1.9	349.6	25.7	621.5	-7.9	499.1	-12.00
		1.4	979.0	97.90	1.2	350.8	25.6	628.2	-8.7	509.3	-14.88
	12	0.6	899.1	89.91	7.5	346.0	26.7	553.2	-1.8	415.3	3.31
		0.8	955.5	95.55	3.7	355.9	23.2	599.6	-5.2	456.6	-4.06
		1	975.4	97.54	1.8	359.4	23.6	616.1	-7.7	477.5	-9.91
		1.2	983.8	98.38	1.0	364.0	25.1	619.7	-9.3	491.6	-13.73
		1.4	990.9	99.09	0.6	366.2	26.8	624.7	-10.3	498.1	-17.09
1500	4	0.6	1182.0	78.8	9.2	403.0	25.9	778.9	2.2	578.3	2.67
		0.8	1322.7	88.2	8.0	452.9	25.3	869.9	0.8	664.7	-1.02
		1	1384.6	92.3	5.7	478.5	23.9	906.0	-2.0	708.7	-5.48
		1.2	1420.8	94.7	3.7	493.0	21.7	927.8	-3.8	736.4	-8.96
		1.4	1440.5	96.0	2.4	507.0	21.0	933.6	-5.5	754.4	-11.47
	8	0.6	1356.0	90.4	9.8	537.4	29.1	818.6	-0.1	611.1	3.57
		0.8	1429.1	95.3	4.9	556.6	24.4	872.5	-4.7	661.1	-4.94
		1	1460.6	97.4	2.5	565.3	23.4	895.3	-7.3	691.6	-10.23
		1.2	1474.7	98.3	1.4	565.9	22.5	908.9	-8.4	711.4	-13.61
		1.4	1481.5	98.8	0.9	565.1	22.4	916.4	-9.0	719.6	-16.56
	12	0.6	1395.9	93.1	7.3	580.6	26.8	815.3	-3.2	616.1	2.14
		0.8	1456.8	97.1	3.0	578.1	21.0	878.8	-6.3	660.0	-6.35
		1	1478.8	98.6	1.3	579.5	21.5	899.3	-8.4	687.6	-11.42
		1.2	1486.1	99.1	0.7	580.8	22.6	905.3	-9.7	699.4	-15.22
		1.4	1490.7	99.4	0.4	575.0	22.8	915.7	-9.9	705.5	-18.32

Table 10 **Crowd-shippers distance deviation and average assigned orders per crowd-shippers by `ADP` and the percentage change compared to `Myopic` for $N_o = 1000$ and $f^{\texttt{Fix}} = 3$.**

ζ	N_c/N_o	$f^{\texttt{Dev}}$	Distance deviation (km)	Change (%)	Assigned orders	Change (%)
1.1	0.6	2	0.14	29.08	2.00	2.83
		3	0.14	22.93	2.00	2.96
		4	0.13	17.25	2.01	3.33
	0.8	2	0.14	17.75	1.93	3.90
		3	0.13	11.53	1.94	4.42
		4	0.13	6.02	1.95	4.85
	1	2	0.13	6.57	1.88	5.24
		3	0.12	0.23	1.89	5.99
		4	0.12	-5.03	1.90	6.50
	1.2	2	0.12	-4.44	1.84	6.37
		3	0.11	-10.19	1.86	7.32
		4	0.11	-14.51	1.87	7.91
	1.4	2	0.11	-12.76	1.81	7.68
		3	0.10	-18.78	1.83	9.22
		4	0.10	-22.43	1.84	9.72
1.3	0.6	2	0.38	23.83	2.09	4.50
		3	0.33	8.96	2.10	4.82
		4	0.29	-0.68	2.10	5.14
	0.8	2	0.29	-13.84	2.01	7.76
		3	0.26	-22.41	2.03	8.39
		4	0.23	-27.62	2.04	8.74
	1	2	0.22	-37.97	1.98	12.20
		3	0.20	-42.89	1.99	12.83
		4	0.18	-45.03	2.00	12.93
	1.2	2	0.18	-50.94	1.93	15.30
		3	0.16	-53.87	1.95	15.79
		4	0.15	-54.87	1.95	15.91
	1.4	2	0.15	-58.13	1.91	18.36
		3	0.14	-60.31	1.92	18.91
		4	0.13	-60.81	1.93	18.73
1.5	0.6	2	0.49	11.28	2.12	4.90
		3	0.40	-3.43	2.13	5.49
		4	0.34	-11.25	2.13	5.59
	0.8	2	0.38	-24.77	2.04	9.07
		3	0.31	-33.15	2.05	9.44
		4	0.26	-36.79	2.05	9.44
	1	2	0.28	-46.60	1.99	13.90
		3	0.23	-50.70	2.00	13.94
		4	0.20	-51.84	2.01	13.73
	1.2	2	0.22	-57.95	1.95	17.71
		3	0.19	-60.08	1.96	17.37
		4	0.17	-60.45	1.97	16.97
	1.4	2	0.18	-65.10	1.93	21.08
		3	0.16	-66.24	1.94	20.82
		4	0.14	-66.05	1.94	20.11