Beta
Research School for Operations
Management and Logistics

The delivery dispatching problem with time windows for urban consolidation centers

W.J.A. van Heeswijk, M.R.K. Mes, J.M.J Schutten

Beta Working Paper series 493

BETA publicatie	WP 493 (working paper)
ISBN	
ISSN	
NUR	
Eindhoven	December 2015

The delivery dispatching problem with time windows for urban consolidation centers

Wouter van Heeswijk[*], Martijn Mes, Marco Schutten

December 21, 2015

Abstract

This paper addresses the dispatch decision problem faced by an urban consolidation center. The center receives orders according to a stochastic arrival process, and dispatches them for the last-mile distribution in batches. The operator of the center aims to find the cost-minimizing consolidation policy, depending on the orders at hand, pre-announced orders, and stochastic arrivals. We present this problem as a variant of the Delivery Dispatching Problem that includes dispatch windows, and model it as a Markov decision problem. For toy-sized instances, we solve this model to optimality. Through numerical experiments on these instances, we show that we approximate the optimal values with an error of less than 2%. Larger instances suffer from intractably large state-, outcome-, and action spaces. We propose an Approximate Dynamic Programming (ADP) algorithm that can handle such instances, using value function approximation to estimate the downstream costs. To cope with large action spaces – with sizes up to 2^{120} in our experiments – we formulate an integer linear program to be used within our ADP algorithm. To evaluate the performance of our ADP policies, we test against various benchmark policies, including a lookahead policy based on scenario sampling. We test the performance of ADP on a variety of networks. When the dispatching problem provides sufficient flexibility in dispatch times, ADP outperforms our myopic benchmark policies by more than 15%, and lookahead policies by over 10%.

1 Introduction

The demand for goods in urban areas is continuously increasing, and is expected to further increase in the future (Transmodal 2012). Another trend in urban freight transport – particularly in retail – is the fragmentation of freight flows within urban areas. The abundance of small and independent carriers, shippers, and receivers contribute to this fragmentation. Transport capacity is therefore often utilized only partially. These trends result in inefficient transportation movements within urban areas. This inefficiency gives rise to various external costs, such as congestion, air pollution, and noise hindrance. In response, governments aim to mitigate such effects with measures such as restricted access areas and road pricing for heavy vehicles. Due to these developments, last-mile distribution within urban areas becomes increasingly complex and expensive. A common supply-side solution to improve the efficiency of urban logistics is the use of urban consolidation centers (Quak 2008, Transmodal 2012). Trucks arriving from the long haul can unload at a

[*]Corresponding author. Address: Department of Industrial Engineering and Business Information Systems, Faculty of Behavioural, Management & Social Sciences , University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands; phone (+31 53 489)4091; fax: (+31 53 489)2159; e-mail: w.j.a.vanheeswijk@utwente.nl.

consolidation center – usually located at the edge of the urban area – instead of making an inefficient delivery tour through the city. Such transshipments allow both for bundling goods – thereby minimizing the number of movements within the city – and dispatching environment-friendly vehicles on the last mile.

The operator in charge of the consolidation center decides on both the timing and the composition of order batches to dispatch. Cost minimization requires that vehicles are used as efficiently as possible. Therefore, the operator has an incentive to wait for future orders to arrive, which may result in better consolidation opportunities and more efficient tours. However, the order arrival process is subject to replanning and disruptions within the supply chain. The degree of communication between the actors within the supply chain dictates how accurate the information on future arrivals is. Furthermore, dispatch decisions are generally bound by hard and/or soft time windows, as well as vehicle availability and storage capacity. The operator faces complete or partial uncertainty in the arrival process, and tries to find a consolidation policy that minimizes the costs under this uncertainty.

We consider a setting in which arriving orders are dynamically revealed to the operator. Orders either arrive at the consolidation center without advance notice, or are pre-announced to arrive at a future point in time. Once orders arrive at the consolidation center, they can be dispatched to the customers in the city. Arriving orders may have a large variety in destinations, dispatch windows, load sizes, etc. In the dispatch decisions, the operator aims to minimize costs by consolidating orders based on these properties, striving for high utilization of vehicle capacity and minimal travel distance. Based on both the available knowledge regarding current orders and the anticipation of new orders, the operator is able to make informed consolidation decisions.

To formalize the problem, we extend the Delivery Dispatching Problem (DDP) by including time windows. We will refer to this extension as the DDP with time windows, or DDP-TW. The absence of time windows is a major shortcoming in the traditional DDP; time windows are an integral component of logistics in general, and perhaps even more so in an urban logistics context. Shipment consolidation policies stemming from the traditional DDP indicate when the full accumulated set of orders should be dispatched. However, when orders are subject to time windows, one would dispatch orders with distinct time windows at different times, holding on to some orders for a later dispatch time. Hence, the decision maker wants to also know which subset of orders to dispatch. Another key difference with existing work on the DDP is that we consider a finite planning horizon, which allows for time-dependent arrival processes. For these reasons, our extension to the DDP provides a better fit with real-life consolidation problems, allowing for applications to dispatching problems in general. The dynamic and stochastic nature of our DDP is modeled using a Markov decision model. This Markov decision model provides the optimal consolidation policy. In line with the DDP definition as provided by (Minkoff 1993), we explicitly separate the dispatch decision from the routing decision, the latter being outside the scope of this paper. We apply a cost function that estimates the transportation costs for dispatching a given set of orders. After determining which orders should be dispatched, a VRP algorithm could subsequently be applied to obtain detailed delivery tours. With this paper, we contribute to existing literature by formally defining the DDP with hard and soft dispatch windows, and designing an algorithm that can handle large instances for this problem class.

The remainder of the paper is structured as follows. Section 2 provides a literature review on the DDP and other related works. We specifically focus on transportation problems dealing with both dynamic and stochastic properties. Section 3 gives a formal definition of the DDP-TW. The corresponding decision problem is modeled as a Markov Decision Problem in Section 4. Section 5 elaborates on our solution method. Both the setup and results of our numerical experiments are given in Section 6, including managerial implications of the results. Finally, we provide the conclusions of our work in Section 7.

2 Literature Review

We present a literature overview that includes studies on the DDP and several related problem classes. We refer to recent literature studies for the various problem classes, and highlight a number of studies that address problems with comparable characteristics. We discuss various solution approaches to transportation problems with dynamic and stochastic properties, and evaluate their suitability to solve the DDP-TW. We conclude with the literature gap that we address with this paper.

Optimization problems that embed both stochastic and dynamic properties are notoriously hard to solve (Powell et al. 2012). Even though stochastic information is recognized as an integral aspect of optimization in transportation problems, the majority of transportation literature focuses on deterministic problems. Traditionally, mathematical programming and (meta)heuristics are used to handle high-dimensional transportation problems. However, these methods generally do not cope well with stochastic information being revealed over time (Powell et al. 2012). Successful incorporation of stochastic information in solution methods is still an ongoing development (Powell et al. 2012, SteadieSeifi et al. 2014). Suitable solution methods are usually based on either stochastic modeling or scenario sampling (Lium et al. 2009, Pillac et al. 2013). The latter is generally applied to extend heuristics and mathematical programs designed for deterministic problems towards stochastic problems. A key challenge with scenario sampling is to correctly represent the stochastic process; incorrect sampling may yield poor results (Lium et al. 2009). In stochastic modeling, the stochastic process is not restricted (i.e., all outcomes may occur), therefore often requiring more computational effort than sampling. However, the problem space is explored more thoroughly, which tends to yield more accurate estimates. As a result, stochastic modeling is more suitable for preprocessed decisions (returning a policy for online-decision making), whereas scenario sampling is more directly applicable to online decision-making.

In the Delivery Dispatching Problem, order arrivals follow a stochastic process and are dispatched in batches at a given decision moment (Minkoff 1993). The aim of solving the DDP is to find a consolidation policy that returns the optimal dispatch time for an accumulated set of orders. With every new arrival, the decision maker assesses (i) the time elapsed since the first order in inventory arrived (representing the maximum service time), and (ii) the volume of the accumulated orders. The consolidation policy is based on one or both of these measures. Dispatching the accumulated orders is subject to a cost function, in which route duration and -costs are pre-defined input (Minkoff 1993). For our DDP, we study recurrent consolidation policies, meaning that the dispatch decision depends on the state of the problem (Higginson and Bookbinder 1995, Powell 2011). The stochastic and dynamic elements embedded in such DDPs give rise to the use of Markov decision models (Minkoff 1993). Although a Markov decision model is a useful framework to describe decision problems, practical implementations generally suffer from intractably large state spaces and expected values that cannot be calculated exactly (Minkoff 1993, Powell 2011, Pillac et al. 2013). Çetinkaya (2005) provides an overview on DDP literature, although describing the problem class as 'inventory and shipment consolidation decisions'. Relatively little work has been done on the optimization of consolidation policies; most studies rather focus on evaluating existing consolidation policies (Çetinkaya 2005, Mutlu et al. 2010, Bookbinder et al. 2011). Studies that do address optimization tend to consider only volume and arrival time as order properties, and present results that are valid only for a limited set of distributions. Çetinkaya and Bookbinder (2003) derive results on optimal policies that are either quantity-based or time-based, but do not combine both measures into a hybrid policy. Bookbinder et al. (2011) describe a generic solution method based on a batch Markovian arrival process, which is able to cope with a multitude of distributions. However, all transition probabilities must be computed to describe the arrival process. This is computationally challenging when considering orders with multiple stochastic properties. Finally, Cai et al. (2014) present an algorithm that – under certain conditions – returns an optimal consolidation policy. The drawback of this algorithm is that every state must be visited, thus it requires the state space to be sufficiently small to fully enumerate.

We now look at various problem classes related to the DDP. The Vehicle Routing Problem (VRP) is concerned with routing rather than dispatch decisions. Despite this key distinction, it has some overlap with the DDP. A common characteristic is that both problems are order-based, i.e., order demand is specified by the customer. Assigning time windows to orders is common in VRPs. However, in a VRP the time windows usually only affect the routing decision, not the dispatch decision as in our DDP. Ritzinger et al. (2015) provide a literature overview of the dynamic and stochastic VRP, which generally considers re-optimization during the execution of routes. The VRP variant that is most related to our DDP combines dynamic order requests with stochastic customer information. A distinction is made between online decision making and preprocessed decision support. For preprocessed decision support, a number of stochastic modeling solutions exist (e.g., Simão et al. 2009, Ulmer et al. 2015), while solutions for online decision problems primarily rely on scenario sampling.

Also the Inventory Routing Problem (IRP) has ties to the DDP (Minkoff 1993). The IRP addresses repeated stock replenishment from a facility to a fixed set of customer locations, the product quantities to deliver, and the decision when to visit a location. The facility optimizes these decisions, given the – possibly stochastic – depletion rates at the customers. The dispatch decision is an integral part of the IRP. A key distinction between the DDP and the IRP is that the latter is generally not order-based. This implies that the facility is not subject to an inbound arrival process, and goods are typically not coupled to individual customers. Furthermore, only one or several types of goods have to be distributed along the customers. For the solution of stochastic IRPs, generally either mathematical programming is applied, or Markov decision models are heuristically solved. We mention a few IRP studies that handle both dynamic and stochastic properties. An example is the study of Coelho et al. (2012), who propose a heuristic solution with scenario sampling for this IRP class. Bertazzi et al. (2013) formulate the stochastic IRP as a Markov decision model, and solve the associated decision problem with a hybrid rollout algorithm. Coelho et al. (2014) provide a recent overview of IRP literature.

The last related problem class that we address is Service Network Design (SND). SND entails the selection and timing of transportation services. Known solution methods mostly use mathematical programming, (meta)heuristics, and graph theory. The majority of SND studies focus on deterministic instances. Steadie-Seifi et al. (2014) mention a number of SND studies that do incorporate stochastic demand (e.g., Lium et al. 2009, Meng et al. 2012). Solutions to the stochastic SND generally are multi-stage mathematical programs, in which scenarios are added to reflect uncertainty in future demand. Lium et al. (2009) state that generating a compact yet representative scenario tree is one of the key challenges in this solution method.

The contribution of this paper is threefold. First, we formulate a Markov decision model for the DDP-TW, as such formally defining our extension of the DDP. Second, we develop a solution method to solve large instances of this problem class. Based on the stochastic modeling frameworks of Topaloglu and Powell (2006) and Powell (2011) for dynamic resource-allocation problems, we develop an ADP algorithm to solve our DDP variant. The main contribution of this algorithm is the design of our value function approximation, while also reflecting on several common implementation issues that are encountered in the design phase. Third, we formulate an Integer Linear Program (ILP) to solve the decision problem within the ADP algorithm. With the ILP, we find approximating solutions when the embedded decision problem cannot be enumerated within reasonable time.

3 Problem Formulation

This section introduces the planning problem. We aim to provide a generic formulation, such that it can be applied to a variety of instances. For the notation of this problem, we expand upon our earlier work (Van Heeswijk et al. 2015). We assume that the characteristics of arriving orders are stochastic and have a known associated probability distribution. When an order is pre-announced or arrives at the center, its exact properties are revealed. Certain variables that are treated as stochastic in our description may be known in practice. In that case, the stochastic variable can simply be replaced by the actual information regarding future orders, creating a restricted instance of our problem. We optimize over a finite planning horizon $\mathcal{T} = \{0, 1, \dots, T\}$, during which orders arrive at the consolidation center. We make dispatch decisions at fixed decision moments $t \in \mathcal{T}$; the decision moments are separated by constant time intervals. The following properties are modeled as stochastic variables in our problem: the number of orders arriving per decision moment, order volumes, order destinations, the hard dispatch window, and the soft dispatch window. Every order must be shipped within its hard window, while a penalty is incurred when dispatching outside the soft window. Note that we use dispatch windows rather than the more common delivery windows. The latter would require to solve the routing problem, thereby significantly increasing the complexity of the model. For an urban logistics setting – in which travel times are relatively short – explicitly including delivery windows would overly complicate the representation of our problem.

Our decision is to choose which subset of the accumulated orders to dispatch at the current decision moment. To take into account the impact of this decision on the future, we evaluate the effects of postponing the dispatch of orders, e.g., distributing them from the center to their destination in the urban area at a later decision moment. It may be possible to combine postponed orders in a dispatch batch with future orders, thereby decreasing overall costs. As such, we optimize over a planning horizon rather than a single decision moment; we measure the impact of current decisions on downstream costs, i.e., the expected costs over the remainder of the horizon.

We consider an urban area with a fixed set of customer locations (i.e., order destinations). Last-mile distribution takes place via a consolidation center at the edge of the area. Our representation of the urban distribution network is as follows. Let $\mathcal{G} = \{\mathcal{V} \cup 0, \mathcal{A}\}$ be a directed graph, with $\mathcal{V} \cup 0$ being the set of vertices and $\mathcal{A}$ being the set of arcs. Vertex 0 represents the consolidation center in the network. The remaining vertices signify the set of order destinations $\mathcal{V} = \{1, 2, \dots, |\mathcal{V}|\}$.

For the transportation of orders, we restrict ourselves to homogeneous fleets, although our method is able to handle heterogeneous fleets as well. We distinguish between primary and secondary vehicles. The total number of primary vehicles $q^{pr,max} \in \mathbb{N}$ is finite, and either represents the fleet of the consolidation center or a rented fleet. To ensure feasible solutions at all times, we assume that the amount of secondary vehicles $q^{se,max} \in \mathbb{N}$ is always sufficiently large to dispatch all accumulated orders, and that secondary vehicles are more expensive than primary vehicles. Secondary vehicles may represent an actual backup option (e.g., renting additional vehicles in case of shortage) or a dummy fleet with infinitely high costs. Such a dummy fleet effectively serves as a hard bound on vehicle capacity, and eases the formulation of the model. We only dispatch secondary vehicles if all primary vehicles are in use.

Each vehicle dispatched at t has a finite and deterministic route duration $\tau_t \in \{1, \dots, \tau^{MaxRoute}\}$. For notational convenience, we assume that all vehicles dispatched at the same decision moment have the same route duration. This assumption can easily be relaxed to handle varying route durations. When dispatching at t, vehicles will be available again at $t + \tau_t$, with τ_t depending on the subset of locations $\mathcal{V}' \subseteq \mathcal{V}$ we visit and the number of vehicles we must dispatch to satisfy constraints on capacity and route duration. For decision-making purposes, we keep track of the dispatch availability of primary vehicles now and in the future. Because vehicles that are dispatched at t are available for dispatch again at a decision moment between $t+1$ and $t+\tau^{MaxRoute}$, we keep track of dispatch availability up to $t+\tau^{MaxRoute}$. q_{t,t^r} denotes the number of primary vehicles available for dispatch at $t+t^r$, with $t^r \in \{0,\dots,\tau^{MaxRoute}\}$. Note that $q_{t,0}$ primary vehicles can be dispatched at t. We store the number of primary vehicles available for dispatch – based on preceding dispatch decisions – in the vector $Q_t = (q_{t,0}, q_{t,1}, \dots, q_{t,\tau^{MaxRoute}})$.

An order is characterized by six properties: destination, load size, hard earliest dispatch time, hard latest dispatch time, soft earliest dispatch time, and soft latest dispatch time. We refer to every unique combination of these six properties as an order type. The order destination is represented by a vertex in $\mathcal{V}$. Let $\mathcal{L} = \{\frac{1}{k}, \frac{2}{k}, \dots, 1\}$ be the discretized set of feasible load sizes, with $k \in \mathbb{N}^+$. The size of an order is an element in $\mathcal{L}$, with the element 1 representing a full load for an urban vehicle. Next, we formulate the hard and soft dispatch windows. To ease the notation, we restrict ourselves to describing the dispatch windows for arriving orders only. Dispatch windows for accumulated orders must be updated over time, we introduce this procedure in Section 4. All time indices are relative to the decision moment t. Every order has a hard dispatch window, within which the order must be dispatched from the depot. The hard dispatch window is defined by an earliest dispatch time $t^{eh} \in \mathcal{T}^{eh}$ and a latest dispatch time $t^{lh} \in \mathcal{T}^{lh}$. Order types with $t^{eh} > 0$ describe pre-announced orders for which all properties are known and deterministic, but that have not yet arrived at the consolidation center. We do not allow for $t^{eh} < 0$, as this would indicate an earliest dispatch time in the past, while t^{eh} for a pre-announced order is bounded from above by $\tau^{MaxTimeAhead}$. Hence, the set of feasible hard earliest dispatch times is

$$\mathcal{T}^{eh} = \{0, \dots, \tau^{MaxTimeAhead}\} \ .$$

The latest dispatch time cannot be a moment in time before the earliest dispatch time, we therefore impose that $t^{lh} \geq t^{eh}$. Furthermore, we set a maximum length $\tau^{MaxWindow} \in \mathbb{N}$ on the hard dispatch window, such that $t^{lh} \leq t^{eh} + \tau^{MaxWindow}$. Hence, we have

$$\mathcal{T}^{lh} = \{t^{eh}, \dots, t^{eh} + \tau^{MaxWindow}\} \ .$$

In addition to the hard dispatch window, we consider a soft dispatch window that is defined by an earliest dispatch time $t^{es} \in \mathcal{T}^{es}$ and a latest dispatch time $t^{ls} \in \mathcal{T}^{ls}$. Dispatching outside the soft window – but within the hard window – is allowed, but in that case a penalty cost is incurred. We impose that for arriving orders, $t^{es} - t^{eh} \leq \tau^{MaxEarliness}$ and $t^{lh} - t^{ls} \leq \tau^{MaxLateness}$, with $\tau^{MaxEarliness}, \tau^{MaxLateness} \in \mathbb{N}$. Given t^{eh} and t^{lh}, the feasible earliest and latest soft dispatch times for arriving orders are

$$\begin{aligned} t^{es} &\in \{t^{eh}, \dots, \min\{t^{lh}, t^{eh} + \tau^{MaxEarliness}\}\} \ , \\ t^{ls} &\in \{\max\{t^{es}, t^{lh} - \tau^{MaxLateness}\}, \dots, t^{lh}\} \ . \end{aligned}$$

With our time notation, we can model a variety of dispatch windows in a finite state space. E.g., setting the hard window equal to the soft window eliminates the soft window, while setting the hard window wide effectively yields a problem with soft windows only. We proceed to define the orders in our problem. Let $I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \in \mathbb{N}$ be the number of a given order type at hand at t. We capture our deterministic knowledge regarding both pre-announced orders and accumulated orders at decision moment t as

$$I_t = (I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}})_{v \in \mathcal{V}, l \in \mathcal{L}, t^{eh} \in \mathcal{T}^{eh}, t^{lh} \in \mathcal{T}^{lh}, t^{es} \in \mathcal{T}^{es}, t^{ls} \in \mathcal{T}^{ls}} \ .$$

For the sake of readability, we omit from here on the set notation when referring to these elements, unless describing exceptions. We continue to define the state of the problem. The state at decision moment t is denoted as $S_t \in \mathcal{S}$; its definition combines the dispatch availability of primary vehicles and the deterministic order knowledge:

$$S_t = (Q_t, I_t) \ .$$

Next, we describe our action space. Let $l^{max} \in \mathbb{N}$ be the maximum number of orders that can be held in the consolidation center, i.e., the maximum inventory remaining after a decision. Rather than a maximum number of orders, we could also have set a constraint on the maximum volume. For every decision moment t, we decide how many orders of every order type in inventory to dispatch. Orders that are not dispatched remain in inventory, and are available at the next decision moment. Let the integer variable $x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}$ describe the number of a specific order type to be dispatched at t. A feasible action at decision moment t is given by

$$x_t(S_t) = (x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}})_{\forall v,l,t^{eh},t^{lh},t^{es},t^{ls}} \ , \tag{1}$$

where

$$\sum_{\forall v,l,t^{lh},t^{es},t^{ls}} (I_{t,v,l,0,t^{lh},t^{es},t^{ls}} - x_{t,v,l,0,t^{lh},t^{es},t^{ls}}) \quad \leq \quad l^{max} \ . \tag{2}$$

$$x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \quad \leq \quad I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \qquad \forall v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \ , \tag{3}$$

$$x_{t,v,l,t^{eh},0,t^{es},t^{ls}} \quad = \quad I_{t,v,l,t^{eh},0,t^{es},t^{ls}} \qquad \forall v, l, t^{eh}, t^{es}, t^{ls} \ , \tag{4}$$

$$x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \quad = \quad 0 \qquad t^{eh} > 0, \forall v, l, t^{es}, t^{lh}, t^{ls} \ , \tag{5}$$

$$x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \quad \in \quad \mathbb{N} \qquad \forall v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \ . \tag{6}$$

Constraint (2) ensures that at most the maximum inventory remains after dispatching. Note that only orders with $t^{eh} = 0$ are part of inventory. With Constraint (3), we ensure that we cannot dispatch more orders of a certain type than are available at the decision moment t. Constraint (4) stipulates that all orders with a latest dispatch time equal to the maximum delay are dispatched. Constraint (5) prevents that we dispatch pre-announced orders that are not yet in inventory. Finally, Constraint (6) ensures that only nonnegative integer amounts of orders can be dispatched. The set of feasible actions in a given state is described by $\mathcal{X}_t(S_t)$.

To calculate the direct costs $C(S_t, x_t)$, we must know how many vehicles we dispatch and what distance they travel. As our focus is not on routing decisions, we use the approximation formula of Daganzo (1984) to estimate routing costs. As these costs must be computed for every possible action for a large amount of states, an efficient cost function is required. Daganzo's formula is known to provide good estimates of VRP distances (Robusté Antón et al. 2004), given some constraints on the number of destinations visited per vehicle, the total number of destinations, and the shape of the service area. The formula is based on the notion that when increasing the number of locations in an area, the length of the tour visiting these locations asymptotically converges to a constant multiplied by the square root of the number of points and the service area (Figliozzi 2009). As the function does not necessarily assume a depot located in the center of the customer locations, the approximation is particularly applicable to our setting, where the consolidation center is typically located at the edge of the urban area. After some rewriting of Daganzo's formula to better fit our problem setting, we estimate the total travel distance – assuming Manhattan distances – with $d(x_t) = \frac{2\bar{d}|\mathcal{V}'|}{|\mathcal{V}'|/\min(|\mathcal{V}'|, q^{pr}+q^{se})} + 0.73\sqrt{|\mathcal{V}'|a}$, with $a \in \mathbb{R}^+$ being the size of the service area (e.g., the surface of a grid), $\mathcal{V}' \subseteq \mathcal{V}$ being the subset of customers visited, q^{pr} and q^{se} being the numbers of primary and secondary vehicles dispatched, and $\bar{d} = (\sum_{v \in \mathcal{V}'} d_{0,v})/|\mathcal{V}'|$ being the average depot-customer distance. The number of vehicles dispatched must be sufficient to both carry the total volume of dispatched orders and satisfy the maximum route duration. After determining the number of vehicles required, we charge a fixed cost per dispatched truck, variable route costs, handling costs per location visited, and penalty costs for orders dispatched outside their soft dispatch window. To conclude, we note that the formula does not consider distances between customers. Therefore, the formula is relatively indifferent to the positioning of customer locations. As we consider an urban setting – that typically has a high number of locations within a small area – and we only need an estimate of the travel costs for a dispatch decision, the formula suffices for the purpose of this paper. However, the formula can easily be substituted by other solution methods, as long as solutions can be calculated efficiently.

4 Markov Decision Model

We model the decision problem as a Markov decision model. Based on the order arrivals that can occur during the planning horizon, we can compute a consolidation policy $\pi_t \in \Pi_t$, with $\pi_t : S_t \mapsto x_t$. To model the uncertainties with respect to the properties of arriving orders, we introduce seven stochastic variables. These are (i) the number of orders arriving O_t, (ii) the order destination V, (iii) the order size L, (iv) the earliest hard dispatch time T^{eh}, (v) the length of the hard dispatch window T^{window}, (vi) the earliness E, and (vii) the lateness D. Based on the realizations of these stochastic variables (given by lower case representation of the variables), we can deterministically compute the values of three remaining variables required for our problem. We obtain the latest hard dispatch time with $T^{lh} = T^{eh} + T^{window}$. To compute the earliest soft dispatch time T^{es}, we introduce the capped earliness $E^{cap} = \min(E, T^{window})$, such that $T^{es} = T^{eh} + E^{cap}$. Similarly, we use the capped lateness $D^{cap} = \max(T^{lh} - T^{es}, T^{lh} - D)$ to obtain the latest soft dispatch time $T^{ls} = T^{lh} - D^{cap}$. The corresponding probability distributions are discrete and finite. To represent all probability distributions with a single variable, we define the exogenous information variable $\tilde{I}_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \in \mathbb{N}$, which indicates the number of arrivals of a specific order type at time $t \geq 1$. The generic variable W_t captures all exogenous information, i.e., all orders that arrive between $t-1$ and t (i.e., before making decision x_t):

$$W_t = [\tilde{I}_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}]_{\forall v,l,t^{eh},t^{lh},t^{es},t^{ls}} \qquad t \geq 1 \enspace .$$

We proceed to describe the transition from S_t to the next state S_{t+1}. The transition is affected by the action x_t and the new arrivals W_{t+1}. Orders that are not dispatched remain in inventory, hence must be included in S_{t+1}. The indices t^{eh}, t^{lh}, t^{es} and t^{ls} are adjusted over time such that they reflect the changing dispatch windows relative to the current time. We impose that $t^{eh}, t^{es} \geq 0$ to reduce the size of the state space; earliest dispatch times in the past would not affect our decisions. However, we do allow t^{ls} to be smaller than 0, as $t^{ls} < 0$ indicates that the soft dispatch window is violated. Observe that this conversion only applies to orders in inventory.

The transition from S_t to S_{t+1} depends on our dispatch decision $x_t \in \mathcal{X}_t$ and the realization of order arrivals, represented by the information variable W_{t+1}. Actions determine the change in vehicle availability from Q_t to Q_{t+1}, and update the remaining orders in inventory. For a given action, let $\bar{q}_{t,\tau_t}$ be the number of primary vehicles dispatched at t that is available for dispatch at $t + \tau_t$. By combining this information with the initial fleet availability Q_t, we can compute Q_{t+1}. This gives us the transition function S^M:

$$S_{t+1} = (Q_{t+1}, I_{t+1}) = S^M(S_t, x_t, W_{t+1}) \enspace , \tag{7}$$

where

$$I_{t+1,v,l,0,t^{lh},0,t^{ls}} = \sum_{\tilde{t}^{eh}=0}^{1} \sum_{\tilde{t}^{es}=0}^{1} (I_{t,v,l,\tilde{t}^{eh},t^{lh}+1,\tilde{t}^{es},t^{ls}+1} - x_{t,v,l,\tilde{t}^{eh},t^{lh}+1,\tilde{t}^{es},t^{ls}+1}) + \tilde{I}_{t+1,v,l,0,t^{lh},0,t^{ls}}, \quad \forall v, l, t^{lh}, t^{ls} \;, \tag{8}$$

$$I_{t+1,v,l,0,t^{lh},t^{es},t^{ls}} = \sum_{\tilde{t}^{eh}=0}^{1} (I_{t,v,l,\tilde{t}^{eh},t^{lh}+1,t^{es}+1,t^{ls}+1} - x_{t,v,l,\tilde{t}^{eh},t^{lh}+1,t^{es}+1,t^{ls}+1}) + \tilde{I}_{t+1,v,l,0,t^{lh},t^{es},t^{ls}}, \quad t^{es} > 0, \forall v, l, t^{lh}, t^{ls} \;, \tag{9}$$

$$I_{t+1,v,l,t^{eh},t^{lh},t^{es},t^{ls}} = I_{t,v,l,t^{eh}+1,t^{lh}+1,t^{es}+1,t^{ls}+1} - x_{t,v,l,t^{eh}+1,t^{lh}+1,t^{es}+1,t^{ls}+1} + \tilde{I}_{t+1,v,l,t^{eh},t^{lh},t^{es},t^{ls}}, \quad t^{eh}, t^{es} > 0, \forall v, l, t^{lh}, t^{ls} \;, \tag{10}$$

$$q_{t+1,t^r} = \begin{cases} q_{t,t^r+1} - \bar{q}_{t,\tau_t+1} & \text{if } 0 \le t^r < \tau_t \;, \\ q_{t,t^r+1} & \text{if } \tau_t \le t^r < \tau^{MaxRoute} - 1 \;, \\ q^{pr,max} & \text{if } \tau^{MaxRoute} - 1 \le t^r \le \tau^{MaxRoute} \;. \end{cases} \tag{11}$$

Constraints (8)-(10) state that the amount of an order type at $t+1$ is the amount of the order type that we have in S_t, minus the amount of the order type that is dispatched at t, plus the amount of the order type that arrives at $t+1$. Constraint (11) ensures that Q_{t+1} is consistently updated. We denote the realization of orders arrivals as $\omega_t \in \Omega_t$. With this, we have all the ingredients required to introduce the optimality equation

$$V_t(S_t) = \min_{x_t \in \mathcal{X}_t(S_t)} \left(C(S_t, x_t) + \sum_{\omega_{t+1} \in \Omega_{t+1}} \mathbb{P}(W_{t+1} = \omega_{t+1}) V_{t+1}(S_{t+1} | S_t, x_t, \omega_{t+1}) \right) \;. \tag{12}$$

We conclude our description of the Markov decision model with the expression for $\mathbb{P}(W_t)$. For $t \ge 1$, let o_t be the number of orders arriving at decision moment t, with $o_t \in \{0, 1, \ldots, o_t^{max}\}$. The probability of o_t orders arriving is given by $\mathbb{P}(O_t = o_t)$. The probability of an arriving order being of a certain order type (i.e., the unique combination of order properties) follows from the multivariate distribution $\mathbb{P}(V, L, T^{eh}, T^{lh}, T^{es}, T^{ls})$. The properties of these stochastic variables have been outlined at the beginning of this section. The probability function for new arrivals $\mathbb{P}(W_t = \omega_t)$, with $t \ge 1$, is given by

$$\begin{aligned} \mathbb{P}(W_t = \omega_t) &= \mathbb{P}(O_t = o_t) \frac{o_t!}{\prod\limits_{\tilde{I}_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \in \omega_t} \tilde{I}_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}!} \\ &\quad \cdot \prod_{\forall v,l,t^{eh},t^{lh},t^{es},t^{ls}} \mathbb{P}(V = v, L = l, T^{eh} = t^{eh}, T^{lh} = t^{lh}, T^{es} = t^{es}, T^{ls} = t^{ls})^{\tilde{I}_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}} \;. \end{aligned} \tag{13}$$

5 Solution Method

When increasing the instance size of our problem, it becomes intractably large in terms of state space, action space, and outcome space. Powell (2011) refers to this phenomenon as the 'three curses of dimensionality', and provides an Approximate Dynamic Programming (ADP) framework that addresses these problems. Based on this framework, we develop an ADP algorithm that we use to solve our DDP with dispatch windows. The goal of solving the problem is to obtain a consolidation policy π^{ADP}, which can be used for online decision making. We separately discuss the three curses of dimensionality that we address.

We start by addressing the size of the outcome space; for this purpose we introduce the concept of the post-decision state (Powell 2011). The post-decision state S_t^x is the state immediately after action x_t, but before the arrival of new information ω_{t+1}. The information embedded in the post-decision state allows to estimate the downstream costs. We can assign expected downstream cost $\mathbb{E}\{V_{t+1}(S_{t+1})|S_t^x\}$ to every post-decision state S_t^x, eliminating the need to evaluate all possible outcomes for every action. Instead, we now represent our decision problem as a deterministic minimization problem. Applying action x_t in state S_t results in a deterministic transition to the post-decision state S_t^x. We express this transition in the function

$$S_t^x = S^{M,x}(S_t) \ , \tag{14}$$

where

$$\begin{aligned} I^x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} &= I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} - x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \ , \\ & \quad \forall t, v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \ , \end{aligned} \tag{15}$$

$$q^x_{t,t^r} = \begin{cases} q_{t,t^r} - \bar{q}_{t,\tau_t} & \text{if } t^r < \tau_t \ , \\ q_{t,t^r} & \text{if } t^r \geq \tau_t \ , \end{cases} \qquad \forall t^r \in \{0,\ldots,\tau^{MaxRoute}\} \ . \tag{16}$$

Next, we address the problem of a large state space. Although adopting the concept of the post-decision state greatly reduces the computational burden, it still requires a post-decision state to be visited in order to learn about its associated downstream costs. As the state space of our problem increases exponentially with the number of order types, visiting all post-decision states would require an unfeasibly large number of iterations. We therefore replace the value function with a single value function approximation (VFA) for the downstream costs: $\bar{V}_t^{n-1}(S_t^x)$, with n being the iteration counter. At every decision moment, we take the best decision given our VFA. Incoming arrivals are generated according to Equation (14). By solving the following deterministic minimization problem for iteration n, the best action is found:

$$\tilde{x}_t^n = \underset{x_t \in \mathcal{X}_t(S_t)}{\arg\min} \left(C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x) \right) \ . \tag{17}$$

Equation (17) provides the action that minimizes the sum of direct costs and estimated downstream costs $\hat{v}_t^n$, given by

$$\hat{v}_t^n = \min_{x_t \in \mathcal{X}_t(S_t)} \left(C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x) \right) \ . \tag{18}$$

Estimated downstream costs may be flawed due to a lack of observations. This may cause the algorithm to stall, and continue to visit the same suboptimal post-decision states. To avoid this, we do not always select the best option obtained with Equation (17); at every decision moment we select a random action $x_t \in \mathcal{X}_t(S_t)$ with probability ϵ. This allows us to explore new states and provide new observations. However, too much exploration will decrease the quality of the estimates. After obtaining $\hat{v}_t^n$, we update our estimate $\bar{V}_{t-1}^{n-1}(S_{t-1}^x)$. For this, we use the following function:

$$\bar{V}_{t-1}^n(S_{t-1}^x) \leftarrow U^V(\bar{V}_{t-1}^{n-1}(S_{t-1}^x), S_{t-1}^x, \hat{v}_t^n) \ . \tag{19}$$

Algorithm 1 provides an outline of our ADP algorithm. At every iteration, we first generate a random arrival path $\omega^n = \{\omega_1, \ldots, \omega_T\}$. We solve Equation (17) for $t = 0$, obtain $W_{t+1}(\omega^n)$, and find S_{t+1} by solving Equation (7). This procedure is repeated until reaching $t = T$. To update the value functions, we make use of a backward pass procedure (Sutton and Barto 1998). This means that after completing the forward iteration, we move backwards to recursively update $\bar{V}_t^{n-1}(S_t^x)$, based on the observed downstream costs $\hat{v}_t^n = C_t(S_t, x_t) + \hat{v}_{t+1}^n$.

Algorithm 1 *ADP backward pass algorithm with post-decision states*

```
Step 0  Initialize
        Step 0a:                 Initialize $\bar{V}_t^0(S_t)$, $\forall t \in \mathcal{T}$, $\forall S_t \in \mathcal{S}$.
        Step 0b:                 Set probability to select random action $\epsilon \in [0, 1]$.
        Step 0c:                 Set iteration counter to $n := 1$ and set
                                 the maximum number of iterations to $N$.
        Step 0d:                 Select an initial state $S_0$.
Step 1  Generate a sample path $\omega^n$.
Step 2  For $t = 0, 1, \ldots, T$ do:
        Step 2a:                 Find the best action $\tilde{x}_t^n$ by solving Equation (17).
                                 With probability $\epsilon$, randomly select $x_t \in \mathcal{X}_t$.
        Step 2b:                 Obtain post-decision state $S_t^x$ via Equation (14).
        Step 2c:                 Obtain sample realization $W_{t+1}(\omega^n)$,
                                 calculate $S_{t+1}$ with Equation (7).
Step 3  For $t = T, \ldots, 1$ do:
        Step 3a:                 If $t < T$, compute $\hat{v}_t^n = C_t(S_t, x_t) + \hat{v}_{t+1}^n$.
        Step 3b:                 Update $\bar{V}_t^{n-1}(S_t^x)$ using Equation (19).
Step 4  Set $n := n + 1$.
        If $n \leq N$, then go to Step 1.
Step 5  Return $\bar{V}_t^N(S_t^x)$   $\forall t \in \mathcal{T}$.
```

To overcome the problem with the large state space, $\bar{V}_t^n(S_t^x)$ must be updated in a way that does not require keeping track of all visited post-decision states. To this end, we make use of VFA with basis functions (Powell 2011). Let $\mathcal{F}$ be a set of features based on the post-decision state, with $f \in \mathcal{F}$ representing an explanatory variable for the true value function. Let $\phi_f : S_t^x \mapsto \mathbb{R}$ be a basis function of feature f (e.g., a polynomial of f). Let θ_f^n be a weight corresponding to ϕ_f. We obtain the value function approximation

$$\bar{V}_t^n(S_t^x) = \sum_{f \in \mathcal{F}} \theta_f^n \phi_f(S_t^x) \qquad \forall t \in \mathcal{T} \ . \tag{20}$$

With every observation, the full set of weights is updated, thereby adjusting the estimated downstream costs for all post-decision states. By visiting a single post-decision state, we obtain an updated set of weights with which we learn to estimate the downstream costs for other states as well. As such, VFA enables us to cope with large state spaces. A key difficulty with VFA is to define a set of basis functions that accurately approximates the downstream costs. Good insight into the structure of the problem is required to select appropriate basis functions. Although correctly estimating the downstream costs is important, Chang et al. (2013) note that an accurate estimate of downstream costs is not a prerequisite for obtaining good policies. Obtaining a good ranking of actions already strongly contributes to obtaining high-qualities policies, as it helps in selecting good actions. A set of basis functions can be viewed as an aggregate state description, retaining sufficient detail to capture the cost structure, but requiring less variables to do so. Large sets of basis functions (i.e., with low aggregation) render the regression procedure both time-inefficient and unstable. Instability often follows from deviating observations which are ‘corrected’ by assigning a very large weight to a basis function of small magnitude. Subsequently, new observations may yield extreme estimates. Using a large set of basis functions is therefore not recommended. On the other hand, basis functions with too much aggregation insufficiently capture the cost structure. Therefore, both the statistical significance and the performance in policy learning should be assessed when selecting basis functions.

After every iteration n, we update the weights $\theta^n_f, \forall f \in \mathcal{F}$ using recursive least squares for nonstationary data. A detailed description of this method can be found in Powell (2011). This procedure requires storing only a small amount of data, and can be swiftly executed during the ADP learning phase. However, least squares regression is not without flaws. Updating the weights is sensitive to outliers in observations, particularly if we also have values for basis functions that are rarely observed. Unstable behavior of the regression is common mostly for large sets of explanatory variables. Burger and Repiskỳ (2012) discuss these problems, as well as various other pitfalls in least squares regression.

We proceed with a brief outline of the regression method. Let B^n be a matrix of size $|\mathcal{F}| \cdot |\mathcal{F}|$. We recursively update this matrix with $B^n = \frac{1}{\lambda^n}(B^{n-1} - \frac{1}{\gamma^n}(B^{n-1}\phi^n(\phi^n)^T B^n - 1))$, with discount factor λ^n and $\gamma^n = \lambda^n + (\phi^n)^T B^{n-1}\phi^n$. Next, we define a matrix $H^n = \frac{1}{\lambda^n}B^{n-1}$. The set of weights is now updated with

$$\theta^n_f = \theta^{n-1}_f - H_n\phi_f(\bar{V}^{n-1}_{t-1}(S^x_{t-1}) - \hat{v}^n_t) \qquad \forall f \in \mathcal{F} \ . \tag{21}$$

Up to this point, we showed (i) how to convert our stochastic decision problem to a deterministic minimization problem, (ii) how to replace the post-decision state with a set of basis functions to estimate the downstream costs, and (iii), the regression procedure to update the estimated costs based on simulated observations. However, we have not yet proposed a way to handle the large action space. Taking into account that we cannot dispatch pre-announced orders and that orders with $t^{lh} = 0$ must be dispatched, the number of possible actions for a state is given by

$$\prod_{\forall v,l,t^{lh}>0,t^{es},t^{ls}} (I_{t,v,l,0,t^{lh},t^{es},t^{ls}} + 1) \ . \tag{22}$$

Eminently, complete enumeration of all decisions is not feasible for large instances. Generally, either mathematical programming or metaheuristics are used to solve large decision problems in ADP (Powell 2009). Therefore, we describe an extension that illustrates how to handle large action spaces. We formulate the decision problem – which is solved within the ADP – as an Integer Linear Program (ILP). This ILP solves Equation (17), and is subject to the set of action constraints (2)–(6) and post-decision constraints (15)–(16). Post-decision variables are required to compute the basis functions. As the Daganzo formula is based on the average customer-depot distances $\sum_{v\in\mathcal{V}'} d_{0,v}$, the ILP would require $|2^{\mathcal{V}}|$ average distances as pre-generated input. To solve large decision problems within reasonable time, we therefore adopt a simplifying assumption, which is based on approximating the sum of distances-to-depot. Let $d \in \{0, 1, \ldots, d^{max}\}$ be an index corresponding to a pre-defined sum of distances-to-depot $\delta_d \in \mathbb{R}^+$, such that we have a set of distances $\{\delta_0, \delta_1, \ldots, \delta_{d^{max}}\}$ with $\delta_d < \delta_{d+1}, \forall d \in \{0, 1, \ldots, d^{max} - 1\}$. The average distance-to-depot can be approximated by $\delta_d/|\mathcal{V}'|$. We set $\delta_{d^{max}} = \sum_{v\in\mathcal{V}} d_{0,v}$. With the ILP, we can solve our large instances within reasonable time, and closely approximate the solutions from the Daganzo formula. To retain linearity, basis functions should be defined linearly with respect to the decision variables. We note that non-linear basis functions may be included in the ILP, at the expense of introducing additional artificial variables. In the appendix, the full ILP model is presented.

6 Experimental setup

In our numerical experiments, we distinguish between the learning phase (in which we update the estimated downstream costs and learn the policy) and the simulation phase (in which we test the performance of a fixed policy learned with ADP). Customer locations are defined on a 10 by 10 grid, and we use a time horizon of $T = 10$ for all experiments.

We divide our experiments in three classes – small (S), medium (M), large (L) – each class having its own purpose. The main characteristics of our instances are listed in Table 1. Preliminary tests on medium-sized instances indicate that 5,000 iterations suffice to learn the policy; further observations do not significantly increase the quality of the policy. In the learning phase, we select a random action with probability $\epsilon = 0.05$, and select the best action with probability 0.95. The setting for ϵ is based on preliminary tests; these showed notably poorer performance when never selecting random actions, while values $\epsilon > 0.10$ tended to decrease performance as well. To initialize the recursive least squares method, we set $B^0 = \psi I$, with $\psi = 0.000001$ and I being the identity matrix. For the discount factor λ^n, we use $\lambda^n = 1 - \bar{\lambda}/n$; $\lambda^n < 1$ indicates that we put more weight on recent observations. Based on preliminary experiments, we fix $\bar{\lambda} = 0.99$, meaning that we put relatively high weights on recent observations during the first few iterations. The reason for this is that weights are initialized at 1, such that the first estimates are far off from the actual values. Our algorithm was coded in Delphi XE6, and ran on a computer with 8GB RAM and a 2.90GHz Intel Core i7 processor. The ILP was solved with CPLEX 12.2.

With our experiments on small instances, we evaluate the behavior of ADP in the learning phase, comparing with the optimal values. As only toy-sized instances can be solved to optimality within reasonable time, these instances do not capture all the dynamics embedded in the problem. However, with the exact values we obtain, we can test the explanatory power of the basis functions, and unveil possible dependencies between basis functions. Hence, solving the toy-sized problem is a first design step in obtaining a consolidation policy with ADP. We define two small instances with distinct customer locations. Instance S1 has $d_{0,v} = 10, \forall v \in \mathcal{V}$, while instance S2 has $d_{0,1} = 5, d_{0,2} = 10, d_{0,3} = 15$. We use dynamic programming to obtain the exact values for these instances. Using these values as our benchmark, we compute the R^2 values (i.e., the coefficient of determination that describes the fit of the model) for multiple sets of basis functions, and test how closely we approximate the exact values in the learning phase.

Our experiments on the medium-sized instances provide insights into both the learning phase and the simulation phase. For our first six medium-sized instances (M1-M6) we take the first 10 customers of the 25-customer Solomon instance R101, scaled to our 10 by 10 grid. We consider a fleet of 8 primary vehicles; this number suffices to handle 85% of order arrival scenarios without using secondary vehicles. The first six instances differ in their degrees of dynamism (DoD) and the maximum length of the dispatch window. We test for DoD=1 (no pre-announced orders) in M1 and M4, DoD=0.5 (half the orders announced one time period in advance) in M2 and M5, and DoD=0 (all orders announced one time period in advance) in M3 and M6. Instances M1, M2, and M3 have a maximum dispatch window of 1; for M4, M5, and M6 we set the maximum dispatch window to 2. This setup allows us to evaluate the performance of ADP under various levels of both dynamism and stochasticity, as well as distinct degrees of flexibility in postponement. Furthermore, we test a number of variants of the aforementioned instances, with uneven numbers being variants of M2 and even numbers being variants of M5. In instances M7 and M8 we consider skewed demand; 2 of the 10 customers generate 50% of order demand. For instances M9 and M10, we use the last 10 customers of the 25-customer Solomon instance R101. In instances M11 and M12, we consider a fleet of 4 primary vehicles rather than 8, while in instances M13 and M14 we consider an unlimited fleet of primary vehicles. With these variants, we test how ADP copes with asymmetric networks, varying customer locations, and scarceness of vehicle capacity. In the learning phase, we test a variety of sets of basis functions, and highlight various issues encountered during implementing and testing. For the best sets of basis functions found, we test the performance of ADP against various benchmark policies in a simulation. The benchmark policies are defined in Section 6.1.

Table 1: Instance properties

	S1,2	M1	M2,7,9,11,13	M3	M4	M5,8,10,12,14	M6	L1	L2
# of customers	3	10	10	10	10	10	10	25	50
Max. # of order arrivals	2	15	15	15	15	15	15	40	40
Max. inventory	-	30	30	30	30	30	30	60	60
Min. time ahead	0	0	0	1	0	0	1	0	0
Max. time ahead	0	0	1	1	0	1	1	1	1
Max. length dispatch window	1	1	1	1	2	2	2	2	2
Soft windows	No	No	No	No	No	No	No	Yes	Yes
# order types	30	200	400	400	300	600	600	1700	3400
# of states	$93,000$	$\gg 10^{22}$	$\gg 10^{45}$	$\gg 10^{45}$	$\gg 10^{24}$	$\gg 10^{50}$	$\gg 10^{50}$	$\gg 10^{138}$	$\gg 10^{163}$

Finally, we perform simulation experiments on two large instances. The main contribution of these experiments is to demonstrate that realistic-sized instances can be handled with the ILP for the decision problem. For these instances, we consider more order arrivals and larger inventory capacity. We test two instances of varying sizes; the 25-customer Solomon instance R101 (L1) and the 50-customer Solomon instance R101 (L2). As the decision problems for these instances are too large to solve by full enumeration, we use the ILP model at the decision moments. The performance in the simulation phase is compared to those of the benchmark policies. Finally, we provide insights into the computation times required to solve the ILP.

We note that the experimental structure serves as a methodology to design appropriate basis functions. First, toy-sized instances are used to obtain exact benchmarks, test the explanatory value of basis functions, and uncover dependencies between basis functions that may lead to an unstable regression procedure. Subsequently, we use medium-sized instances that are large enough to capture the complexities of the problem, but still sufficiently small to test a variety of settings with limited computational effort. For the realistic-sized instances – where computation time becomes a problem – we can focus on performance and computational speed, having already obtained the key modeling insights from the small- and medium-sized instances.

Taking into account that pre-announced orders may accumulate and arrive at the same decision moment, we obtain a weak lower bound for the size of the state space by computing

$$\frac{|I_t|!}{(o_t^{max} \cdot (\tau_{MaxTimeAhead} + 1))! \cdot (|I_t| - o_t^{max} \cdot (\tau_{MaxTimeAhead} + 1))!} \ . \tag{23}$$

6.1 Benchmark policies

For our problem, exact benchmarks can only be computed for toy-sized instances. To evaluate the performance of ADP for larger instances, we therefore compare with three consolidation policies without lookahead, as well as a policy based on scenario sampling.

Under the first myopic policy π^{direct}, we always ship orders as soon as possible, as long as primary vehicle capacity is available. With the second myopic policy π^{post}, we postpone as many orders as possible, until an order reaches its latest dispatch time. For both policies, we fill up a vehicle that is to be dispatched with orders that are sorted and assigned as described in Algorithm 2. The third myopic policy is π^{min}, which enumerates the full set of actions, and minimizes the direct costs without estimating downstream costs. As no extra costs are charged to transport additional orders to locations that are already visited, we dispatch as much volume as possible if costs are equal. Thus, the three myopic policies make use of consolidation opportunities by utilizing the remaining capacity of dispatched vehicles, but do not take into account the stochastic arrival process.

We proceed to explain our policy based on scenario sampling, π^{sample}. The idea behind this policy it that we generate a number of sample arrival paths and solve heuristically the decision problems for these paths, thereby obtaining an estimate for the downstream costs. To apply π^{sample} at a given decision moment, we first generate a set Ω^m, which contains m random arrival paths (scenarios) of length τ^{sample}, i.e., $\omega^m = \{\omega^m_{t+1}, \dots, \omega^m_{t+\tau^{sample}}\}$. We then enumerate all actions x_t – meaning that the action space should be sufficiently small – to obtain the direct costs $C(S_t, x_t), \forall x_t \in \mathcal{X}_t$. Next, we select a path $\omega^m \in \Omega^m$, and obtain $S_{t+1} = S^M(S_t, x_t, W_{t+1}(\omega^m_{t+1}))$. After arriving in S_{t+1}, we use the heuristic-based policy π^{direct} to obtain decisions for the remainder of the horizon, and denote the resulting costs as $\tilde{V}_t(S_t, \omega^m)$. These costs can be calculated recursively for $t' = t+1, \dots, t+\tau^{sample}$ using

$$\tilde{V}_{t'}(S'_t, \omega^m) = C(S_{t'}, x_{t'}) + \tilde{V}_{t'+1}(S_{t'+1}, \omega^m) \tag{24}$$

with $S_{t'+1} = S^M(S_{t'}, x_{t'}, W_{t'+1}(\omega^m_{t'+1}))$ and $x_{t'} \hookleftarrow S_{t'} : \pi^{direct}$. We repeat this procedure for every post-decision state and for all scenarios $\omega^m \in \Omega^m$. Thus, the estimated post-decision values are obtained as follows:

$$\hat{V}_t(S^x_t) = \frac{\sum\limits_{\omega^m \in \Omega^m} \tilde{V}_{t+1}(S_{t+1}, \omega^m)}{|\Omega^m|} \ . \tag{25}$$

We obtain the best decision by solving

$$\arg\min_{x_t \in \mathcal{X}_t} = C(S_t, x_t) + \hat{V}_t(S^x_t) \ . \tag{26}$$

We do not compare the performance of ADP in L1 and L2 with scenario sampling; this would require to calculate downstream costs for every post-decision state in advance. As the ILP is designed to handle outcome spaces that are too large to enumerate, we also cannot pre-generate the downstream costs for all post-decision states. This problem could be overcome by assigning downstream costs to aggregated post-decision states. However, the design and selection of such aggregated states – while keeping running time at an acceptable level – are research problems that stretch beyond the scope of this paper.

Algorithm 2 *Heuristic rules for π^{direct} and π^{post}*

Step 0	Sort orders.	
	Step 0a:	Sort available orders based on lowest t^{lh}.
	Step 0b:	Sort available orders with same t^{lh} based on smallest size.
Step 1	While orders with $t^{lh} = 0$ are unassigned do:	Assign order with $t^{lh} = 0$ to vehicle.
Step 2	While remaining inventory exceeds l^{max} do:	Assign first capacity-feasible order on list to vehicle.
Step 3	If π^{direct}:While capacity from primary vehicles is sufficient do:	Assign first capacity-feasible order on list to vehicle.
	If π^{post}: While capacity from already dispatched vehicles is sufficient do:	Assign first capacity-feasible order on list to vehicle.

6.2 Generating a set of initial states

To learn a policy with ADP based on a given initial state, we can start every iteration with the same initial state, and obtain a policy tailored to that specific state. While this is useful for testing purposes, in practical settings we prefer not to repeat the entire learning phase for every state we encounter. Instead, we want to run the learning phase only once, obtaining a policy that is applicable to all states over a longer period of time. However, randomly generating an initial state at every iteration could result in policies based on states that we rarely encounter in practice. To obtain a set of realistic initial states $\mathcal{S}' \subset \mathcal{S}$, we perform a predefined number of simulation runs with π^{direct} – starting every run with a randomly generated initial state – and store the state encountered in after $T/2$ time periods in $\mathcal{S}'$. Next, at the start of every iteration we select $S_0 \in \mathcal{S}'$, with the probability of selecting a given state being proportional to the frequency it was encountered when creating $\mathcal{S}'$. As order arrivals are random, we still visit states $S_t \notin \mathcal{S}'$. However, by learning based on commonly encountered states, the results provide a better fit in practical settings.

7 Numerical results

In this section, we present the numerical results of our experiments. First, we show the results for small instances, then for medium-sized instances, and finally for large instances.

7.1 Experiments on small instances

We start by applying dynamic programming to the small instances, such that we obtain the values for all states as an exact benchmark. We define basis functions $\phi_f(S_t), \forall S_t \in \mathcal{S}$, and carefully construct sets of basis functions $\phi \supset \phi_f(S_t), \forall S_t \in \mathcal{S}$. Every set of basis functions hould include a constant $\phi_0 = 1$. The cost structure of our problem mainly depends on (i) the locations visited, (ii) the volume dispatched, (iii) the number of primary vehicles available, and (iv) the flexibility to postpone orders. Our basis functions reflect one or more of these properties. It is important that basis functions within a set are independent of each other; weights cannot be learned properly if deviating observations are explained by multiple basis functions. We divide our basis functions into three categories that are assumed to be largely independent: vehicle-based, location-based, and volume-based. Non-segregated basis functions (e.g., the ratio of vehicle capacity and volume per dispatch time) combine several of these properties and therefore may have a higher explanatory value, but are difficult to combine with other basis functions due to dependency. Also, the associated weights may be more difficult to learn. From the regression statistics, the significance of each basis function can be deducted, as well as dependencies between basis functions.

To assess the explanatory value of various sets of basis functions, we first compute their R^2 values, and then test how these sets perform in the learning phase. We start by computing R^2 for a number of separate basis functions to assess their explanatory value. Subsequently, we test a number of sets in which multiple basis functions are combined. A set ϕ yielding a high R^2 value does not guarantee good performance in the learning phase, and vice versa. The reason for this is twofold. First, the R^2 value is computed based on perfect information, while in ADP we learn weights based on a limited number of observations. Second, as mentioned before, the performance of ADP partially depends on the ranking of values of states, not only the degree of approximating the downstream costs. Therefore, we test how closely we approximate the optimal values in the learning phase. Using the procedure described in Section 6.2, we obtain the state set $\mathcal{S}'$. For all $S_0 \in \mathcal{S}'$, we perform 5,000 iterations to learn the basis function weights. A set of basis functions is considered as a candidate set if it satisfies the following criteria: (i) basis functions are independent of each other, (ii)

Table 2: R^2 values and value gaps for various sets of basis functions

Set	Basis functions	# basis functions	S1:R^2	Avg. gap	S2: R^2	Avg. gap
ϕ^1	Constant	1	-	6.76%	-	6.14%
ϕ^2	# available destinations at $t^{eh} = 0$	2	0.16	2.80%	0.14	3.57%
ϕ^3	# vehicles	$1 + \tau^{MaxRoute}$	0.34	6.77%	0.31	6.14%
ϕ^4	Total vol. per t^{lh}	$1{+}\tau^{LatestDispatch}$	0.35	2.11%	0.32	2.78%
ϕ^5	Total vol. per v	$1{+}\lvert\mathcal{V}\rvert$	0.27	2.97%	0.31	3.34%
ϕ^6	# orders per type	$1{+}\lvert\mathcal{I}\rvert$	0.35	5.27%	0.40	5.36%
ϕ^7	# Vehicles/# available destinations	2	0.41	5.63%	0.37	6.21%
ϕ^8	# Vehicles/Total vol. per t^{lh}	$1{+}\min(\tau^{MaxRoute}, \tau^{LatestDispatch})$	0.23	6.48%	0.20	6.70%
ϕ^9	# destinations per t^{lh}	$1{+}\tau^{LatestDispatch}$	0.14	2.92%	0.12	3.50%
ϕ^{10}	Vol. per location per t^{lh}	$1{+}(\tau^{LatestDispatch}) \cdot \lvert\mathcal{V}\rvert$	0.35	2.97%	0.39	3.34%
ϕ^{11}	# available destinations+ Total vol. per t^{lh}	$2{+}\tau^{LatestDispatch}$	0.48	2.05%	0.43	2.76%
ϕ^{12}	# vehicles+Total vol. per t^{lh}	$1{+}\tau^{MaxRoute}{+}\tau^{LatestDispatch}$	0.69	2.31%	0.62	2.92%
ϕ^{13}	# available destinations+ # vehicles per t^r+Total vol. per t^{lh}	$2 + \tau^{MaxRoute} + \tau^{LatestDispatch}$	0.82	2.03%	0.74	2.75%

every basis function in the set contributes significantly to the R^2 value, (iii) regression can be performed within reasonable time, (iv) the number of basis functions is scalable to larger instances, (v) robust numerical outcomes for all $S_0 \in S'$ (no deviations over 20% from the exact values), and (vi) the average value gap is smaller than 3.0% for both small instances.

The results for 13 sets of basis functions are shown in Table 2. We observe some discrepancies between the R^2 values and the performance in the learning phase. In general, we see that sets including the volume per t^{lh} yields estimates of good quality, which indicates that this property well explains the ranking of values of states. Furthermore, we observe that combining more than three types of basis function often results in unstable behavior. Similar instabilities are found when combining multiple basis functions of the same type, e.g., two volume-based basis functions. Nonsegregated, nonlinear functions do not yield favorable results in terms of accuracy, and are notably difficult to combine with other basis functions. ϕ^{13} yields the best results of all tested sets.

We illustrate the ADP performance using set ϕ^{13} on S1, comparing ADP estimates to exact values for the states in $\mathcal{S}'$. Figure 1 shows the distribution of absolute deviations from the optimal value. For 99.43% of the states ADP yields estimates that are higher than the optimal value, with an average absolute deviation of 1.9%. The tendency to overestimate costs follows from using suboptimal policies to estimate the downstream costs (Powell 2011). 0.24% of the states deviate from the optimal value by more than 5%, these deviations cause the tail on the right. In Figure 2, we show the ADP estimates for all $S_0 \in \mathcal{S}'$, plotted against the optimal values. This plot gives an insight in the ranking of ADP estimates compared to the true ranking.

7.2 Experiments on medium-sized instances

In our medium-sized instances, the number of order types is much larger than for the small instances. Therefore, we test how our sets of basis functions perform under this increased complexity. From Section 7.1, we take the sets of basis functions that yielded a value gap smaller than 5% on both S1 and S2. We re-evaluate the performance of these sets on M2 and M5, as these can be considered as hybrids of the other medium-sized instances. As we are ultimately interested in the performance of the policies rather than behavior in the learning phase, we no longer consider fixed initial states. Instead, we randomly draw an initial state from $\mathcal{S}'$ at every iteration. In total we perform 5,000 iterations to learn π^{ADP}. Subsequently, we

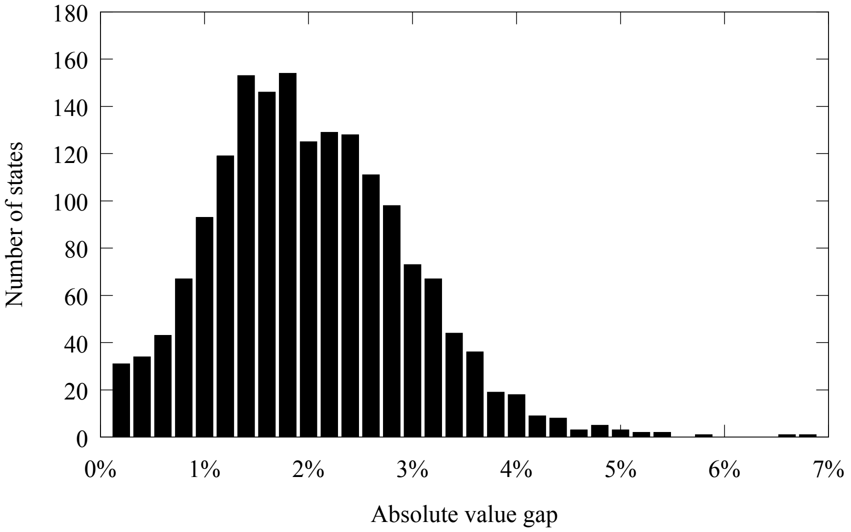

Figure 1: Histogram of absolute differences between exact values and ADP estimates for all $S_0 \in \mathcal{S}'$

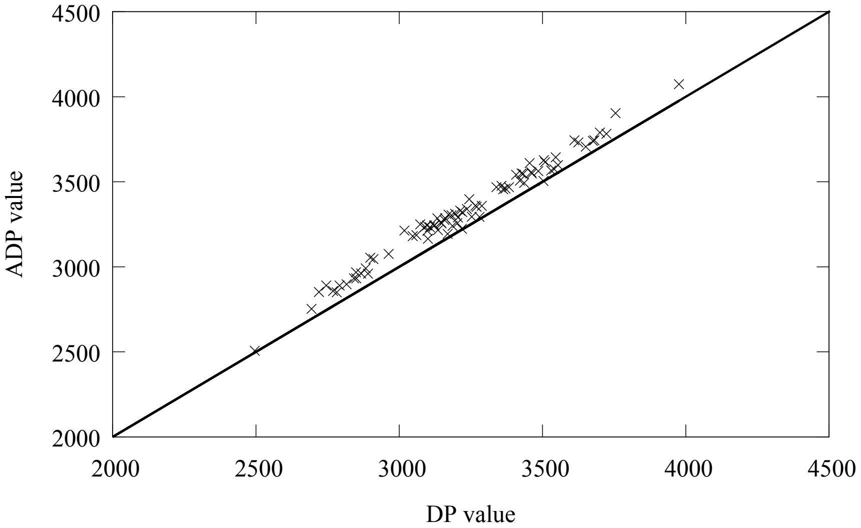

Figure 2: ADP estimates plotted against exact values for all $S_0 \in \mathcal{S}'$

Table 3: Performance of various ϕ on M2 and M5, relative to ϕ^1.

Set	Basis functions	Avg. gap (M2)	Avg. gap (M5)
ϕ^1	Constant	0%	0%
ϕ^2	# available destinations at $t^{eh} = 0$	-0.6%	0.2%
ϕ^4	Total vol. per t^{lh}	4.2%	10.1%
ϕ^5	Total vol. per v	5.0%	9.9%
ϕ^9	# destinations per t^{lh}	3.4%	7.5%
ϕ^{10}	Vol. per location per t^{lh}	3.7%	7.8%
ϕ^{11}	# available destinations+Total vol. per t^{lh}	2.4%	8.9%
ϕ^{12}	# vehicles+Total vol. per t^{lh}	7.7%	13.3%
ϕ^{13}	# available destinations+ # vehicles per t^r+Total vol. per t^{lh}	7.9%	14.0%

perform 1,000 iterations to measure the performance of the learned policy, with initial states again drawn from $\mathcal{S}'$. The average costs in the simulation phase represent the performance of the policy. We measure the performance of all policies relative to the performance under policy ϕ^1. The results are shown in Table 3. The best results are obtained with ϕ^{13} for both M2 and M5. Observe that the outperformance in M5 is notably stronger than with M2; ADP performs better when there is more flexibility in the dispatch decision.

We proceed to test the performance of the learned ADP policies for all medium-sized instances. We learn policies by using the set of basis functions the ϕ^{13}. Again, we randomly draw an initial state from $\mathcal{S}'$ at every iteration. After learning π^{ADP} for all instances, we perform 1,000 simulation runs to evaluate their performance. The results of these experiments in the simulation phase are shown in Table 4. ADP clearly and consistently outperforms the myopic benchmark policies π^{direct}, π^{post}, and π^{min} for all instances. Furthermore, we ran the sampling policy π^{sample} four times, testing with (i) 2 sample paths of length 2 ($\pi^{sample}_{2,2}$) (ii) 2 sample paths of length 5 ($\pi^{sample}_{2,5}$), (iii) 5 sample paths of length 2 ($\pi^{sample}_{5,2}$), and (iv) 5 sample paths of length 5 ($\pi^{sample}_{5,5}$). Scenario sampling consistently provides better results than the myopic policies. Note that sampling policies with shorter lookahead horizons perform better than those with longer horizons, possibly indicating that the former better captures the direct impact of a decision. For M3 and M13, scenario sampling even outperforms the ADP policy. For all other instances, ADP consistently outperforms scenario sampling. For all benchmarks, we note that ADP performs better when dispatch windows are larger (instances M4-M6). This indicates that ADP is more beneficial when there is sufficient flexibility in decision making. The performance of scenario sampling appears to increase when a larger percentage of orders is pre-announced. This may be because we then solve scenarios that are partially deterministic, as such increasing the accuracy of downstream cost prediction. We see that for M7 and M8 (uneven demand distribution) ADP performs worse than M2 and M5 (as its basis functions do not take into account this demand distribution), but still outperforms all benchmark policies. Changing the customer locations (M9, M10) has no notable effect. Reducing the fleet size does not significantly alter the performance for M11, but for M12 (having more flexibility in allocating orders to the scarce vehicle capacity) the myopic policies are outperformed by a larger margin. Increasing the fleet size (M13, M14) to infinite has strong effects. In particular, π^{post} performs notably worse, while the performances of π^{ADP} and π^{sample} are close. As spreading dispatch times over the horizon has less impact when capacity is abundant, using ADP has less added value. On a final note, the results of both ADP and scenario sampling compared to the myopic policies indicate that lookahead based on the stochastic arrival process significantly improves performance.

Table 4: Simulation performance benchmark policies, relative to ADP with ϕ^{13}

Instance	π^{direct}	π^{post}	π^{min}	$\pi^{sample}_{2,2}$	$\pi^{sample}_{2,5}$	$\pi^{sample}_{5,2}$	$\pi^{sample}_{5,5}$
M1	8.92%	9.46%	10.13%	5.32%	6.33%	3.92%	5.08%
M2	7.86%	9.25%	9.18%	4.49%	5.91%	3.76%	4.63%
M3	2.29%	3.31%	3.31%	-2.63%	-1.48%	-2.34%	-1.26%
M4	14.95%	15.55%	16.68%	11.47%	15.52%	9.65%	13.37%
M5	13.98%	14.70%	15.65%	12.22%	16.34%	10.75%	13.88%
M6	13.30%	13.94%	14.90%	9.73%	13.12%	9.09%	12.42%
M7	6.66%	8.80%	8.63%	3.55%	4.24%	2.18%	2.93%
M8	11.34%	15.32%	16.21%	11.34%	14.14%	8.81%	12.94%
M9	7.66%	9.29%	9.26%	4.64%	6.21%	4.10%	4.59%
M10	13.01%	14.32%	15.29%	11.70%	15.54%	10.31%	13.52%
M11	8.31%	9.55%	9.43%	3.62%	4.40%	2.44%	4.10%
M12	16.74%	20.17%	20.04%	10.39%	14.56%	12.37%	13.79%
M13	6.00%	15.08%	7.35%	0.17%	0.71%	-0.24%	0.67%
M14	12.04%	28.59%	14.24%	2.19%	2.75%	1.42%	1.80%
Average	10.22%	13.38%	12.16%	6.30%	8.45%	5.45%	7.32%

Table 5: Simulation performance of ADP policies against benchmark policies, relative to ADP with ϕ^{14}

Instance	π^{direct}	π^{post}	π^{min}	Avg. solving time ILP	% solved within 60s	Diff. with cost function
L1	18.58%	10.88%	12.90%	3.52s	100.00%	1.26%
L2	8.02%	10.41%	14.30%	7.67s	98.30%	1.65%
Average:	13.30%	10.65%	13.60%	5.60s	99.15%	1.45%

7.3 Experiments on large instances

For the large instances, we deal with action spaces with sizes up to 2^{120}. Enumerating all actions within reasonable time is not possible for these instances. In our experiments, we solve the decision problem at each decision moment with the ILP. Based on preliminary testing on medium-sized instances, we set the number of distances $d^{max} = 30$, yielding 30 distance-to-depot sums to estimate the actual travel distance. On average, this setting yields an average absolute cost difference between the ILP and our cost function of less than 2%. Furthermore, we now learn π^{ADP} with 1,000 iterations rather than 5,000, to keep computational time at a reasonable level. To account for the soft windows, we add a basis function that counts the number of orders with $t^{ls} = 0$ to our existing basis functions, and refer to the new set of basis functions as ϕ^{14}. The results of our experiments on L1 and L2 are shown in Table 5. As stated before, we compare the performance with the benchmark policies without lookahead. Even though using only 1,000 iterations to learn the values of $\gg 10^{138}$ states, the benchmark policies are clearly outperformed. In addition, we provide statistics on the time it takes to solve the ILP with CPLEX; the average solving time per decision problem is 5.60s. Although significant computational effort is required in the offline learning phase (as we need to solve the decision problem for N iterations times $T + 1$ time periods), the ILP needs only to be solved once per decision moment in the online application. Finally, we also show the average difference between the approximated cost function used in the ILP and our original cost function; this difference is calculated by using the ILP solution as input to the original cost function.

8 Conclusions

In this paper, we addressed the dispatching problem faced by operators of urban consolidation centers. These centers have a key role in many urban logistics initiatives, as they allow bundling fragmented freight flows to facilitate efficient distribution within urban areas. The operator of the center receives less-than-truckload orders that need to be distributed as efficiently as possible, but does not have perfect foresight into the arrival process. The operator periodically dispatches batches of accumulated orders, taking into account both current and future consolidation opportunities. To provide a formal definition of this problem, we extended the Delivery Dispatching Problem by including both hard and soft dispatch windows. We formulated this problem as a Markov decision model. Like many stochastic models, this Markov decision model becomes computationally intractable for larger instances of the problem. To be able to solve larger instances, we described an ADP algorithm based on value function approximation, with which we are able to overcome the computational problems associated with large state spaces and outcome spaces. The consolidation policy is learned offline, and can be periodically updated, e.g., if the stochastic arrival process changes. The learned policy can subsequently be applied on online decision problems. To address computational problems with large action spaces, we formulated an ILP model that can be used to solve the decision problem embedded in the ADP algorithm.

In our numerical experiments, we mainly focused on implementation problems and design choices. We proposed a three-step methodology to find suitable VFAs for realistic-sized problems. First, we showed how to use toy-sized instances to evaluate basis functions, using the exact values obtained by dynamic programming as a benchmark. We paid special attention to achieving stability of the regression procedure, and how an accurate ranking of actions influences the policies. Next, we evaluated the performance of ADP policies on various medium-sized instances, testing the effects of various network settings and flexibility in dispatch windows. To evaluate the quality of the ADP policies, we compared their performance to two policies based on heuristic rules, to a direct cost minimization policy, and to a policy that utilizes scenario sampling to estimate the downstream costs. We identified a suitable set of basis functions, for which the resulting ADP policy consistently outperformed the benchmark policies. Finally, we solved two large instances – having state spaces with sizes $\gg 10^{138}$ and action spaces of sizes up to 2^{120} – using the ILP to solve the decision problems within ADP, illustrating the suitability of our algorithm to handle realistic-sized instances. Again, the resulting ADP policy outperformed the benchmark policies. On average, we could solve the ILP in 5.60 seconds for our large instances, thereby sufficing for online decision making. However, due to the large number of iterations needed in ADP, the required computational effort to learn the consolidation policy remains significant. Therefore, heuristic reduction of the action space may be beneficial before implementing our approach in practice.

APPENDIX:ILP for large action spaces

As stated in Section 5, we use an ILP to solve decision problems that cannot be solved in reasonable time by enumerating. We present the ILP model in this appendix, starting by introducing the notation required for the objective function. The parameters $c^{pr,fix}$ and $c^{se,fix}$ indicate the fixed costs per primary and secondary vehicle dispatched. The binary variables $z^{pr}_{q^{pr}}$ and $z^{se}_{q^{se}}$ indicate whether q^{pr} primary vehicles and q^{se} secondary vehicles are dispatched. $q^{pr,max}$ and $q^{se,max}$ are the maximum numbers of primary and secondary vehicles that can be dispatched. The variable travel costs depend on the number of locations visited n, the index d for the approximate sum of distances-to-depot δ_d, and the dispatched number of primary vehicles q^{pr} and secondary vehicles q^{se}. These costs are given by parameter $c^{var}_{n,d,q^{pr},q^{se}}$; the binary variable $z_{n,d,q^{pr},q^{se}}$

indicates the unique combination of number of locations, distance and vehicles dispatched. When customer $v \in \mathcal{V}$ is visited, handling costs c_v^{hdl} are incurred. A visit to a location is indicated by the binary variable z_v^{vis}. Finally, parameters $c_{v,t^{es}}^{pen}$ and $c_{v,t^{ls}}^{pen}$ indicate the penalty costs for earliness and lateness, respectively. The objective function for our decision problem is

$$\begin{aligned}
\min_{x_t \in \mathcal{X}_t(S_t)} \quad & \Bigg(\sum_{q^{pr}=0}^{q^{pr,max}} z_{q^{pr}}^{pr} \cdot q^{pr} \cdot c^{pr,fix} + \sum_{q^{se}=0}^{q^{se,max}} z_{q^{se}}^{se} \cdot q^{se} \cdot c^{se,fix} + \\
& \sum_{n=0}^{|\mathcal{V}|} \sum_{d=0}^{d^{max}} \sum_{q^{pr}=0}^{q^{pr,max}} \sum_{q^{se}=0}^{q^{se,max}} z_{n,d,q^{pr},q^{se}} \cdot c_{n,d,q^{pr},q^{se}}^{var} + \\
& \sum_{v \in \mathcal{V}} z_v^{vis} \cdot c_v^{hdl} + \\
& \sum_{\forall t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \cdot c_{v,t^{es}}^{pen} + \\
& \sum_{\forall t,v,l,t^{eh},t^{lh},t^{es},t^{ls}<0} x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \cdot c_{v,t^{ls}}^{pen} + \\
& \sum_{f \in \mathcal{F}} \theta_f \cdot \phi_f(S_t^x) \Bigg) \quad .
\end{aligned}$$

The objective function is subject to a number of constraints. First, we introduce several inventory-based constraints. Constraint (27) indicates that after an action, at most l^{max} orders remains. Constraint (28) ensures that no more orders of a certain type can be shipped than the number available of that type. Constraint (29) stipulates that all orders with $t^{lh} = 0$ must be shipped, while Constraint (30) prevents orders with $t^{eh} > 0$ from being dispatched. Finally, Constraint (31) ensures that the post-decision inventory is calculated correctly.

$$\begin{aligned}
\sum_{\substack{v \in \mathcal{V}, l \in \mathcal{L}, t^{lh} \in \mathcal{T}^{lh}, \\ t^{es} \in \mathcal{T}^{es}, t^{ls} \in \mathcal{T}^{ls}}} (I_{t,v,l,0,t^{lh},t^{es},t^{ls}} - x_{t,v,l,0,t^{lh},t^{es},t^{ls}}) &\leq l^{max} \quad , && (27) \\
x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} &\leq I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \qquad \forall t, v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \quad , && (28) \\
x_{t,v,l,0,0,0,t^{ls}} &= I_{t,v,l,0,0,0,t^{ls}} \qquad \forall v, l, t^{ls} \quad , && (29) \\
x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} &= 0 \qquad , t^{eh} > 0, \forall t, v, l, t^{lh}, t^{es}, t^{ls} \quad , && (30) \\
I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}^x &= I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} - x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} && (31) \\
& \qquad \forall t, v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \quad .
\end{aligned}$$

Constraints (32) and (33) are included to compute the artificial binary variable $z_{\tau_r}^{away}$ for all route durations. $z_{\tau_r}^{away} = 1$ indicates that dispatched vehicles are still away at $t + \tau_r^{away}$, while $z_{\tau_r}^{away} = 0$ indicates that the dispatched vehicles are available again. Travel times depend on the number of locations n visited, the approximate sum of distances-to-depot δ_d, and the total number of dispatched vehicles $q^{pr} + q^{se}$. Feasible travel times are given by the parameter $s_{n,d,q^{pr}+q^{se}} \leq \tau^{MaxRoute}$.

$$z^{away}_{\tau_r} \geq \frac{\sum_{n=0}^{|\mathcal{V}|}\sum_{d=0}^{d^{max}}\sum_{q^{pr}=0}^{q^{pr,max}}\sum_{q^{se}=0}^{q^{se,max}} \left(z_{n,d,q^{pr},q^{se}} \cdot s_{n,d,q^{pr}+q^{se}}\right) - \tau_r}{\tau^{MaxRoute}} \qquad \forall \tau_r \in \{0,\dots,\tau^{MaxRoute}\} \ , \tag{32}$$

$$\sum_{\tau_r \in \{0,\dots,\tau^{MaxRoute}\}} z^{away}_{\tau_r} = \sum_{n=0}^{|\mathcal{V}|}\sum_{d=0}^{d^{max}}\sum_{q^{pr}=0}^{q^{pr,max}}\sum_{q^{se}=0}^{q^{se,max}} z_{n,d,q^{pr},q^{se}} \cdot s_{n,d,q^{pr}+q^{se}} \ . \tag{33}$$

The variable q^{pr}_{t,τ_r} indicates the number of vehicles available for dispatch at $t+\tau_r$. Variable $q^{x,pr}_{t,\tau_r}$ indicates the same for the post-decision state. With constraints (34), (35), and (36), we consistently update the post-decision availability of primary vehicles based on the average dispatch time:

$$q^{x,pr}_{t,\tau_r} \leq q^{pr}_{t,\tau_r} \qquad \forall \tau_r \in \{0,1,\dots,\tau^{MaxRoute}\} \ , \tag{34}$$

$$q^{x,pr}_{t,\tau_r} \leq q^{pr}_{t,\tau_r} - \sum_{q^{pr}=0}^{q^{pr,max}} z^{pr}_{q^{pr}} \cdot q^{pr} + (1 - z^{away}_{\tau_r}) q^{pr,max} \qquad \forall \tau_r \in \{0,1,\dots,\tau^{MaxRoute}\} \ , \tag{35}$$

$$q^{x,pr}_{t,\tau_r} \geq q^{pr}_{t,\tau_r} - \sum_{q^{pr}=0}^{q^{pr,max}} z^{pr}_{q^{pr}} \cdot q^{pr} \qquad \forall \tau_r \in \{0,1,\dots,\tau^{MaxRoute}\} \ . \tag{36}$$

The binary variable z^{dis}_d corresponds to the distance parameter δ_d. Constraint (37) states that the capacity of the dispatched vehicles should be sufficient to carry all dispatched orders. Constraints (38) and (39) the approximate sum of depot-to-customer distances is at least equal to the actual sum. Constraint (40) ensures that no secondary vehicles are dispatched unless all primary vehicles have been dispatched. Finally, constraints (41) and (42) state that only one integer number of vehicles may be dispatched, both for primary and for secondary vehicles.

$$\sum_{q^{pr}=0}^{q^{pr,max}} z^{pr}_{q^{pr}} \cdot q^{pr} + \sum_{q^{se}=0}^{q^{se,max}} z^{se}_{q^{se}} \cdot q^{se} \geq \sum_{l \in \mathcal{L}} \sum_{v,t^{eh},t^{lh},t^{es},t^{ls}} x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \cdot l \ , \tag{37}$$

$$\sum_{v \in \mathcal{V}} z^{vis}_v \cdot d_{0,v} \leq z^{dis}_d \cdot \delta_d \qquad \forall d \ , \tag{38}$$

$$\sum_{d=0}^{d^{max}} z^{dis}_d = 1 \ , \tag{39}$$

$$z^{pr}_{q^{pr,max}} \geq z^{se}_{q^{se}} \qquad \forall q^{se} \in \{1,\dots,q^{se,max}\} \ , \tag{40}$$

$$\sum_{q^{pr} \in \{0,\dots,q^{pr,max}\}} z^{pr}_{q^{pr}} = 1 \ , \tag{41}$$

$$\sum_{q^{se} \in \{0,\dots,q^{se,max}\}} z^{se}_{q^{se}} = 1 \ . \tag{42}$$

Constraints (43) and (44) ensure that $z^{vis}_v = 1$ if and only if location v is visited. Constraint (45) states that the total number of locations visited is equal to the sum of unique locations visited; its correct working is asserted by Constraint (45).

$$z_v^{vis} \geq \frac{\sum\limits_{\forall l,t^{eh},t^{lh},t^{es},t^{ls}} x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}}{I^{total}} \qquad \forall v \in \mathcal{V} \ , \tag{43}$$

$$z_v^{vis} \leq \sum_{\forall l,t^{eh},t^{lh},t^{es},t^{ls}} x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} \qquad \forall v \in \mathcal{V} \ , \tag{44}$$

$$\sum_{n\in\{0,1,\ldots,|\mathcal{V}|\}} n z_n^{loc} = \sum_{v\in\mathcal{V}} z_v^{vis} \ , \tag{45}$$

$$\sum_{n\in\{0,1,\ldots,|\mathcal{V}|\}} z_n^{loc} = 1 \ . \tag{46}$$

The set of constraints (48)–(51) enforce that $z_{n,d,q^{pr},q^{se}} = 1$ for exactly one combination of the number of locations visited, total tour distance, primary vehicles dispatched, and secondary vehicles dispatched.

$$z_{n,d,q^{pr},q^{se}} \leq z_n^{loc} \qquad \forall n, d, q^{pr}, q^{se} \ , \tag{47}$$

$$z_{n,d,q^{pr},q^{se}} \leq z_{q^{pr}}^{pr} \qquad \forall n, d, q^{pr}, q^{se} \ , \tag{48}$$

$$z_{n,d,q^{pr},q^{se}} \leq z_{q^{se}}^{se} \qquad \forall n, d, q^{pr}, q^{se} \ , \tag{49}$$

$$z_{n,d,q^{pr},q^{se}} \leq z_d^{dis} \qquad \forall n, d, q^{pr}, q^{se} \ , \tag{50}$$

$$z_{n,d,q^{pr},q^{se}} \geq z_{q^{pr}}^{pr} + z_{q^{se}}^{se} + z_d^{dis} + z_n^{loc} - 3 \qquad \forall n, d, q^{pr}, q^{se} \ , \tag{51}$$

Finally, the values of the decision variables are subject to:

$$\begin{aligned}
z_{\tau_r}^{away} &\in \{0,1\} && \forall \tau_r \in \{0,\ldots,\tau^{MaxRoute}\} \\
z_{q^{pr}}^{pr} &\in \{0,1\} && \forall q^{pr} \in \{0\ \ldots, q^{pr,max}\} \\
z_{q^{se}}^{se} &\in \{0,1\} && \forall q^{se} \in \{0\ \ldots, q^{se,max}\} \ , \\
z_n^{loc} &\in \{0,1\} && \forall n \in \{0,1,\ldots,|\mathcal{V}|\} \ , \\
z_v^{vis} &\in \{0,1\} && \forall v \in \mathcal{V} \ , \\
z_d^{dis} &\in \{0,1\} && \forall d \in \{0,1,\ldots,d^{max}\} \ , \\
z_{n,d,q^{pr},q^{se}} &\in \{0,1\} && \forall n, d, q^{pr}, q^{se} \ , \\
I_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}}^{x} &\in \mathbb{N} && \forall v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \ , \\
q_{t,\tau_r}^{x,pr} &\in \mathbb{N} && \forall \tau_r \in \{0,1,\ldots,\tau^{MaxRoute}\} \ , \\
x_{t,v,l,t^{eh},t^{lh},t^{es},t^{ls}} &\in \mathbb{N} && \forall v, l, t^{eh}, t^{lh}, t^{es}, t^{ls} \ .
\end{aligned}$$

References

Bertazzi, L., Bosco, A., Guerriero, F., and Lagana, D. (2013). A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27:89–107.

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

Burger, M. and Repiskỳ, J. (2012). Problems of linear least square regression. *Proceedings in ARSA-Advanced Research in Scientific Areas*, 1(1):257–262.

Cai, Q., He, Q.-M., and Bookbinder, J. H. (2014). A tree-structured markovian model of the shipment consolidation process. *Stochastic Models*, 30(4):521–553.

Çetinkaya, S. (2005). Coordination of inventory and shipment consolidation decisions: A review of premises, models, and justification. In *Applications of supply chain management and e-commerce research*, pages 3–51. Springer.

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

Chang, H. S., Hu, J., Fu, M. C., and Marcus, S. I. (2013). *Simulation-based algorithms for Markov decision processes*. Springer Science & Business Media.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Coelho, L. C., Laporte, G., and Cordeau, J.-F. (2012). *Dynamic and stochastic inventory-routing*. Technical Report CIRRELT 2012-37. CIRRELT.

Daganzo, C. F. (1984). The distance traveled to visit n points with a maximum of c stops per vehicle: An analytic model and an application. *Transportation Science*, 18(4):331–350.

Figliozzi, M. A. (2009). Planning approximations to the average length of vehicle routing problems with time window constraints. *Transportation Research Part B: Methodological*, 43(4):438–447.

Higginson, J. K. and Bookbinder, J. H. (1995). Markovian decision processes in shipment consolidation. *Transportation Science*, 29(3):242–255.

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

Meng, Q., Wang, T., and Wang, S. (2012). Short-term liner ship fleet planning with container transshipment and uncertain container shipment demand. *European Journal of Operational Research*, 223(1):96–105.

Minkoff, A. S. (1993). A markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Operations Research*, 41(1):77–90.

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

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Powell, W. B. (2009). What you should know about approximate dynamic programming. *Naval Research Logistics*, 56(3):239–249.

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

Powell, W. B., Simão, H. P., and Bouzaiene-Ayari, B. (2012). Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3):237–284.

Quak, H. (2008). *Sustainability of urban freight transport: Retail distribution and local regulations in cities*. Number EPS-2008-124-LIS. Erasmus Research Institute of Management (ERIM).

Ritzinger, U., Puchinger, J., and Hartl, R. F. (2015). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, (DOI: 10.1080/00207543.2015.1043403t):1–17.

Robusté Antón, F., Estrada, M., and López-Pita, A. (2004). Formulas for estimating average distance traveled in vehicle routing problems in elliptic zones. *Transportation Research Record: Journal of the Transportation Research Board*, 1873(1):64–69.

Simão, H. P., Day, J., George, A. P., Gifford, T., Nienow, J., and Powell, W. B. (2009). An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197.

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

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Topaloglu, H. and Powell, W. B. (2006). Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1):31–42.

Transmodal, M. (2012). DG MOVE European Commission: Study on urban freight transport.

Ulmer, M. W., Brinkmann, J., and Mattfeld, D. C. (2015). Anticipatory planning for courier, express and parcel services. In *Logistics Management*, pages 313–324. Springer.

Van Heeswijk, W. J., Mes, M. R., and Schutten, M. J. (2015). An approximate dynamic programming approach to urban freight distribution with batch arrivals. *Lecture Notes in Computer Science*, 9335:61–75.