A JOURNAL OF THE INSTITUTE FOR OPERATIONS RESEARCH AND THE MANAGEMENT SCIENCES

informs.

TRANSPORTATION SCIENCE

Volume 51 • Number 1 • February 2017

Special Issue: Facility Logistics

Special Issue of Transportation Science on Facility Logistics
René de Koster, Kevin Gue, Iris F. A. Vis

Layout Planning of Sortation Conveyors in Parcel Distribution
Stefan Fedtke, Nils Boysen

Polynomial Time Algorithms to Minimize Travel Time in AS/RS
Amir Hossein Gharehgozli, Yugang Yu, Xiandong Zhang, René de Koster

Evaluating and Optimizing Live-Cube Compact Storage
Nima Zaerpour, Yugang Yu, René de Koster

Wave Release Times for Order Fulfillment Systems with Deadlines
Erdem Çeven, Kevin R. Gue

Model for Interterminal Container Transportation
Nishant Mishra, Debjit Roy, Jan-Kees van Ommeren

Regular Articles

Integrated Disruption Management and Flight Planning
Lavanya Marla, Bo Vaaben, Cynthia Barnhart

Scheduling Trucks in a Cross-Dock
Peter Bodnar, René de Koster, Kaveh Azadeh

Competition Effects and the Entry of High Speed Rail
Luis Cadarso, Vikrant Vaze, Cynthia Barnhart, Ángel Marín

Control of Airport Service Rates
Alexandre Jacquillat, Amedeo R. Odoni, Mort D. Webster

Integrated Aircraft Routing, Crew Pairing, and Tail Assignment
Sebastian Ruther, Natashia Boland, Faramroze G. Engineer, Ian Evans

Demand-Oriented Integrated Scheduling
Oliver Faust, Jochen Gönsch, Robert Klein

Single Leg Flight Airline Revenue Management
J. B. G. Frenk, Behrooz Pourghannad, Semih O. Sezer

Integrated Fleeting, Routing, and Crew Pairing
Shengzhi Shao, Hanif D. Sherali, Mohamed Haouari

Real-World Integrated Airline Scheduling Problem
Valentina Cacchiani, Juan-José Salazar-González

Insights for Shuttle-Based Compact Storage Systems
Elena Tappia, Debjit Roy, René de Koster, Marco Melacini

Dynamic Cost Efficiency in Port Infrastructure
Beatriz Tovar, Alan Wall

Scheduling Automated Yard Crane Systems at Container Terminals
Ulf Speer, Kathrin Fischer

TBDP Approach for Operations Planning in Transshipment Yards
Alena Otto, Xiyu Li, Erwin Pesch

Continuum Approximation for Dynamic Facility Location
Xin Wang, Michael K. Lim, Yanfeng Ouyang

Reliable Hub Network Design
Trung Hieu Tran, Jesse R. O'Hanley, M. Paola Scaparra

Maximum Expected Covering Problem for District Design
Sardar Ansari, Laura Albert McLay, Maria E. Mayorga

Acknowledgment to Referees (2016)

The publication of the Transportation Science & Logistics Society of INFORMS

The Delivery Dispatching Problem with Time Windows for Urban Consolidation Centers

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

http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Articles in Advance, pp. 1–19
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

The Delivery Dispatching Problem with Time Windows for Urban Consolidation Centers

W. J. A. van Heeswijk,[a] **M. R. K. Mes,**[a] **J. M. J. Schutten**[a]

[a] Department of Industrial Engineering and Business Information Systems, University of Twente, 7522 NB Enschede, Netherlands

Contact: w.j.a.vanheeswijk@utwente.nl, http://orcid.org/0000-0002-5413-9660 (WJAvH); m.r.k.mes@utwente.nl (MRKM); m.schutten@utwente.nl (JMJS)

Received: December 18, 2015
Revised: September 23, 2016; February 10, 2017; April 7, 2017
Accepted: April 12, 2017
Published Online in Articles in Advance: August 28, 2017

https://doi.org/10.1287/trsc.2017.0773

Copyright: © 2017 INFORMS

Abstract. This paper addresses the dispatching problem faced by an urban consolidation center. The center receives orders according to a stochastic arrival process and dispatches them in batches for the last-mile distribution. The operator of the center aims to find the cost-minimizing consolidation policy, depending on the orders at hand, preannounced orders, and stochastic arrivals. We present this problem as a variant of the delivery dispatching problem that includes dispatch windows and define a corresponding Markov decision model. Larger instances of the problem suffer from intractably large state-, outcome-, and action spaces. We propose an approximate dynamic programming (ADP) algorithm that can handle such instances, using a linear value function approximation to estimate the downstream costs. To design the value function approximation, we construct various sets of basis functions, numerically evaluate their suitability, and discuss the properties of good basis functions for the dispatching problem. Numerical experiments on toy-sized instances show that the best set of basis functions approximates the optimal values with an error of less than 3%. To cope with large action spaces, we formulate an integer linear program to be used within our ADP algorithm. We evaluate the performance of ADP policies against four benchmark policies: two heuristic policies, a direct cost minimization policy, and a post-decision rollout policy. We test the performance of ADP on a variety of networks. ADP consistently outperforms the benchmark policies, performing particularly well when there is sufficient flexibility in dispatch times.

History: This paper has been accepted for the *Transportation Science* Special Issue on Recent Advances in Urban Transportation Through Optimization and Analytics.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2017.0773.

Keywords: urban distribution • transport planning • freight consolidation • approximate dynamic programming • dispatching problem

1. Introduction

The demand for goods in urban areas is continuously increasing and is expected to increase further in the future, leading to larger numbers of transport movements (Crainic, Ricciardi, and Storchi 2004). 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 contributes to this fragmentation. Transport capacity is therefore often utilized only partially. These trends result in inefficient transport movements within urban areas, giving 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 low-emission zones and road pricing for heavy vehicles. Because of 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 (Crainic, Ricciardi, and Storchi 2004). Trucks arriving from the long haul can unload at a consolidation center—usually located at the edge of the urban area—instead of making an inefficient delivery tour through the city. Such transhipments 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, dispatching decisions are generally bound by time windows as well as vehicle availability and storage capacity. The operator faces complete or partial uncertainty in the arrival process and wants to determine 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 receiving a notification in advance or are preannounced to arrive at a future point in time. Once orders arrive at the consolidation center, they may be dispatched to customers. Arriving orders may have a large variety in destinations, dispatch windows, load sizes, etc. When dispatching orders, 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 dispatching decisions.

To formalize the problem, we extend the delivery dispatching problem (DDP)—as introduced by Minkoff (1993)—by embedding time windows into the problem formulation. 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 as time windows are an integral component of logistics. Shipment consolidation policies stemming from the traditional DDP indicate when the complete accumulated set of orders should be dispatched. However, when orders are subject to time windows, one tends to 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 us to handle 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.

We formulate a Markov decision model to capture the dynamic and stochastic nature of the DDP-TW. Solving instances of our DDP-TW provides an optimal consolidation policy. In line with the DDP definition as provided by Minkoff (1993), we explicitly separate the dispatching decision from the routing decision; the focus of this paper is on the dispatching decision. We apply a cost function that estimates the transport costs for dispatching a given set of orders. After determining which orders to dispatch, a vehicle routing problem (VRP) algorithm can subsequently be applied to obtain detailed delivery tours. For completeness, we also demonstrate an integrated solution in which we explicitly solve the routing problem embedded into the dispatching problem. With this paper, we contribute to existing literature by formally defining the DDP-TW, designing an algorithm that can handle large instances of this problem class, and providing insights into our linear value function approximation and the corresponding explanatory variables that capture the cost structure of the problem.

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 transport problems dealing with both dynamic and stochastic properties. Section 3 gives a formal definition of the DDP-TW and presents the corresponding decision problem as a Markov decision model. Section 4 elaborates on our solution method and the design of a linear value function approximation by using basis functions. In Section 5, we describe our experimental setup; the numerical results are discussed in Section 6. Finally, we provide the main conclusions of our work in Section 7.

2. Literature Review

This section presents a literature overview that includes studies on the DDP and several related problems. The overview comprises recent literature studies for these related problems and highlights a number of studies that address characteristics comparable to our problem. We discuss various solution approaches to transport problems with dynamic and stochastic properties and evaluate their suitability to solve the DDP-TW. The section concludes 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, Simão, and Bouzaiene-Ayari 2012). Even though stochastic information is recognized as an integral aspect of optimization in transport problems, the majority of papers in the transport literature focus on deterministic problems. Traditionally, mathematical programming and (meta)heuristics are used to handle high-dimensional transport problems. However, these methods generally do not cope well with stochastic information being revealed over time (Powell, Simão, and Bouzaiene-Ayari 2012). Successful incorporation of stochastic information in solution methods is still an ongoing development (Powell, Simão, and Bouzaiene-Ayari 2012, SteadieSeifi et al. 2014). Suitable solution methods are usually based on either (i) approximating a value function that uses analytical expressions to define the stochastic process or (ii) sampling scenarios and solving the decision problems for the corresponding outcomes (Lium, Crainic, and Wallace 2009, Pillac et al. 2013). The latter method is generally applied to extend heuristics and mathematical programs designed for deterministic problems toward stochastic problems. A key challenge with scenario sampling is to correctly represent the stochastic process; incorrect sampling may yield poor results (Lium, Crainic, and Wallace 2009). When approximating a value function, the stochastic process is fully defined, but the corresponding expected values may be hard to compute. These expected values therefore tend to be computed offline; the resulting policies can subsequently be used in online decision making. As these policies return decisions as a function of the input state, they tend to be faster than online sampling methods.

In the DDP, 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 as experienced by the customer) 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 route costs are predefined input (Minkoff 1993). For our DDP-TW, we study recurrent consolidation policies, meaning that the dispatching 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 Markov decision models are useful to define decision problems, practical implementations generally suffer from intractably large state spaces and the inability to exactly compute expected values (Minkoff 1993, Powell 2011, Pillac et al. 2013). Çetinkaya (2005, p. 5) provides an overview on DDP literature describing the problem class as "inventory and shipment consolidation decisions." Relatively little work has been done on the optimization of consolidation policies; most studies focus on evaluating existing consolidation policies (Çetinkaya 2005, Mutlu, Çetinkaya, and Bookbinder 2010, Bookbinder, Cai, and He 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, Cai, and He (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, He, and Bookbinder (2014) present an algorithm that—under the assumption of a specific cost structure—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-TW. The VRP is concerned with routing rather than dispatching decisions. Despite this key distinction, it has some overlap with the DDP-TW. A common characteristic is that both problems are customer-driven. Assigning time windows to orders is common in VRPs. However, in a VRP, the time windows usually only affect the routing decision, not the dispatching decision as in the DDP-TW. Ritzinger, Puchinger, and Hartl (2016) provide a literature overview of the dynamic and stochastic VRP, which generally considers reoptimization during the execution of routes. The VRP variant that has the strongest relation to our DDP-TW combines dynamic order requests with stochastic customer information. Examples of works that address this variant are Simão et al. (2009), Ulmer, Brinkmann, and Mattfeld (2015), and Ulmer et al. (2017).

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 customer. The dispatching 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 among 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 by Coelho, Laporte, and Cordeau (2014), 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. A recent overview of the IRP literature can be found in Coelho, Cordeau, and Laporte (2014).

The next related problem class that we address is service network design (SND). SND entails the selection and timing of transport services. Known solution methods mostly use mathematical programming, (meta)heuristics, and graph theory. The majority of SND studies focus on deterministic instances. SteadieSeifi et al. (2014) mention a number of SND studies that incorporate stochastic demand, for example, Lium, Crainic, and Wallace (2009) and Meng, Wang, and Wang (2012). Solutions to the stochastic SND generally are multistage mathematical programs in which scenarios are added to reflect uncertainty in future demand. Lium, Crainic, and Wallace (2009) state that generating a compact yet representative scenario tree is one of the key challenges in this solution method.

Finally, we discuss the emerging problem class of same-day deliveries. In same-day delivery problems, the decision maker dynamically receives order requests during the day and must decide when to dispatch a vehicle that delivers (a subset of) the accumulated orders. If a request is not fulfilled during the day, a penalty is incurred. Voccia, Campbell, and Thomas (2017) define a corresponding Markov decision model and propose a solution method that takes into account information about future requests. Arslan et al. (2016) solve a ride-sharing variant of the problem using a rolling horizon approach. Klapp, Erera, and Toriello (2016) study a single-vehicle variant of this problem in which the customers are located on a line; as a consequence, route duration depends only on the customer located farthest away from the depot. They propose both a rollout algorithm and a solution based on approximate linear programming to address the stochasticity embedded in the problem.

The contribution of this paper is threefold. First, we formulate a Markov decision model for the DDP-TW, in this way formally defining the time-window 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 approximate dynamic programming (ADP) algorithm to solve the DDP-TW. The main contribution is the design of our value function approximation while also reflecting on several implementation issues that are encountered in the design phase. In particular, we contribute to the literature by studying a variety of basis functions that can be used to estimate the downstream costs of decisions. Third, we formulate an integer linear program (ILP) to solve the decision problem within the ADP algorithm. With the ILP, we find approximate solutions when the embedded decision problem cannot be enumerated within a reasonable time. The present work expands on van Heeswijk, Mes, and Schutten (2015). Compared with the earlier work, our key contributions are (i) extending the model with an ILP to solve large instances; (ii) more thorough, detailed, and realistic numerical experiments, focusing on the design of the linear value function approximation; and (iii) improved validation of the model by comparing with stronger benchmark policies.

3. Problem Formulation

This section introduces the decision problem, which we model as a Markov decision model. We aim to provide a generic formulation, such that it can be applied to a variety of instances. We assume that the characteristics of arriving orders are stochastic and have a probability distribution that is known to the operator of the consolidation center. When an order is preannounced 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 random variable can simply be replaced by the actual information regarding future orders, creating a restricted instance of our problem. We optimize over a finite set of decision moments $\mathcal{T} = \{0, 1, \ldots, T\}$, which are separated by equidistant time intervals. The dispatching decisions are made at decision moments $t \in \mathcal{T}$. In this paper, the equidistant time intervals that separate the decision moments correspond to a setting in which the consolidation center dispatches orders a few times a day. This choice is motivated by the common practice that consolidation centers often make two delivery rounds per day (Bohne, Ruesch, and Barrera 2015, Nesterova and Quak 2015). The practice of dispatching vehicles at fixed times may be reinforced by access time restrictions set by local governments, only allowing trucks within the city center during one or two time intervals per day (Nuzzolo, Crisalli, and Comi 2012). We stress that the aforementioned motivation applies to modeling decision moments at predefined times, not to the actual practice of commencing delivery tours at each decision moment. The model that we define does not require dispatching orders at every decision moment. Furthermore, our model does not place any restrictions on the length of time intervals between decision moments.

The following properties are modeled as random variables in our problem: (i) the number of orders arriving per decision moment, (ii) order volumes, (iii) order destinations, and (iv) the dispatch window. Every order must be shipped within its dispatch window. We choose to use dispatch windows at the consolidation center rather than the more common delivery windows at the receiver. The main reason for this choice is that it simplifies the presentation of our problem. In a typical urban logistics setting, adopting dispatch windows rather than delivery windows will not pose problems. Urban deliveries are characterized by relatively short travel times; when time intervals between dispatching moments are set relatively large, a feasible delivery plan can generally be constructed within this interval. In addition, Cherrett et al. (2012) state that the majority of independent retailers (the typical target customers for an urban consolidation center) have no influence on the exact delivery times. Nevertheless, we stress that delivery windows can be handled within our solution method; in Section 3.5, we provide suggestions for this extension.

At every decision moment $t \in \mathcal{T}$, we must choose which subset of the accumulated orders to dispatch. To take into account the impact of this decision on the future, we evaluate the effects of postponing the dispatch of orders. It may be possible to combine postponed orders into a dispatch batch with future orders, thereby decreasing overall costs. As such, we optimize over a planning horizon rather than at a single decision moment; we measure the impact of current decisions on the downstream costs, that is, 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 $\mathscr{G} = \{\mathscr{V} \cup 0, \mathscr{A}\}$ be a directed graph with $\mathscr{V} \cup 0$ being the set of vertices and $\mathscr{A}$ being the set of arcs. Vertex 0 represents the consolidation center in the network. The remaining vertices signify the set of order destinations $\mathscr{V} = \{1, 2, \ldots, |\mathscr{V}|\}$.

3.1. Fleet Description

For the transport of orders, we restrict ourselves to vehicles with identical capacity, although our method allows handling heterogeneous fleets as well. We distinguish between primary and secondary vehicles. The set of primary vehicles—denoted by $\mathscr{Q}^{pr}$—is finite and either represents a fleet owned by the consolidation center or a rented fleet. To ensure feasible solutions at all times, we assume that there are always enough secondary vehicles in the set $\mathscr{Q}^{se}$ to dispatch all accumulated orders and that secondary vehicles are more expensive than primary vehicles. We refer to an individual vehicle as $q \in \mathscr{Q}^{pr} \cup \mathscr{Q}^{se}$. 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. A dummy fleet with infinite costs effectively serves as a bound on vehicle capacity and eases the formulation of the model. We only dispatch secondary vehicles if all primary vehicles are in use.

A vehicle q dispatched at decision moment $t \in \mathscr{T}$ has a finite and deterministic route duration $r_{t,q} \in \{1, \ldots, \tau^{\text{MaxRoute}}\}$. The dispatched vehicle is available for dispatch again at $t + r_{t,q}$, where $r_{t,q}$ is determined by the deployed routing function $\Delta\colon \mathscr{V}_{t,q} \mapsto \{1, \ldots, \tau^{\text{MaxRoute}}\}$, $q \in \mathscr{Q}^{pr} \cup \mathscr{Q}^{se}$ with $\mathscr{V}_{t,q} \subseteq \mathscr{V}$ denoting the subset of locations visited. As we take downstream costs into account, we keep track of the dispatch availability of primary vehicles both now and in the future; the return time of secondary vehicles is not relevant. The earliest availability of vehicle q, relative to the current time t, is denoted by $\tau_{t,q} \in \{0, \ldots, \tau^{\text{MaxRoute}}\}$; vehicle q can only be dispatched at decision moment t if $\tau_{t,q} = 0$. We store the dispatch availability of each primary vehicle—based on preceding dispatching decisions—in the vector $Q_t = (\tau_{t,1}, \tau_{t,2}, \ldots, \tau_{t,|\mathscr{Q}^{pr}|})$.

3.2. Order Types

An order is characterized by four properties: destination v, load size l, earliest dispatch time t^e, and latest dispatch time t^l. We refer to every unique combination of these four properties as the order type (v, l, t^e, t^l). The order destination is represented by a vertex $v \in \mathscr{V}$. The size of an order is an element $l \in \mathscr{L}$, with $\mathscr{L} = \{1/k, 2/k, \ldots, 1\}$ being the discretized set of feasible load sizes, and $k \in \mathbb{N}$. The element $l = 1$ represents a full load for an urban vehicle. Next, we formulate the dispatch windows. Every order has a dispatch window within which the order must be dispatched from the depot. The time indices of the window are relative to the decision moment t. Therefore, the dispatch windows for orders held in inventory must be updated over time; we introduce this procedure in Section 3.6. Each dispatch window is defined by an earliest dispatch time $t^e \in \mathscr{T}^e$ and a latest dispatch time $t^l \in \mathscr{T}^l$. Order types with $t^e > 0$ describe preannounced orders for which all properties are known and deterministic but that have not yet arrived at the consolidation center. We impose that $t^e \geq 0$ as knowledge regarding dispatch times in the past does not impact our decision while t^e for a preannounced order is bounded from above by τ^{MaxAhead}. Hence, the set of feasible earliest dispatch times is

$$\mathscr{T}^e = \{0, \ldots, \tau^{\text{MaxAhead}}\}.$$

The latest dispatch time cannot be a moment in time before the earliest dispatch time; we therefore impose that $t^l \geq t^e$. Furthermore, we impose a maximum width $\tau^{\text{MaxWindow}} \in \mathbb{N}$ on the dispatch window, such that $t^l \leq t^e + \tau^{\text{MaxWindow}}$. Hence, we have

$$\mathscr{T}^l = \{t^e, \ldots, t^e + \tau^{\text{MaxWindow}}\}.$$

3.3. State Description

We have introduced all properties that define an order type. Let $I_{t,v,l,t^e,t^l} \in \mathbb{N}$ be the available number of orders of a given type at decision moment t. We capture our deterministic knowledge regarding both preannounced orders and accumulated orders at t as

$$I_t = (I_{t,v,l,t^e,t^l})_{v \in \mathscr{V}, l \in \mathscr{L}, t^e \in \mathscr{T}^e, t^l \in \mathscr{T}^l}.$$

For the sake of readability, we omit the set notation when referring to these elements. We continue to define the state of the problem. The state at decision moment t is denoted as S_t in state space $\mathscr{S}$; its definition combines the earliest possible dispatch time of each primary vehicle as embedded in vector Q_t and the deterministic order knowledge stored in vector I_t

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

3.4. Action Description

We now describe the actions that we can take in a state. Let $I^{\max} \in \mathbb{N}$ be the maximum number of orders that can be held in the consolidation center, that is, the maximum inventory remaining after a decision. At each decision moment t, we decide how many orders to dispatch of each order type in inventory. Orders that are not dispatched remain in the inventory and are available at the next decision moment. Let the integer variable $x_{t,v,l,t^e,t^l,q}$ describe the number of orders of a specific type to be dispatched at t, served by vehicle q. As a consequence of assigning orders to individual vehicles, the route durations of dispatched vehicles may vary. A feasible action at decision moment t is given by

$$x_t(S_t) = (x_{t,v,l,t^e,t^l,q})_{v,l,t^e,t^l,q}, \tag{1}$$

where

$$\sum_{v,l,t^l}\left(I_{t,v,l,0,t^l} - \sum_{q\in\mathcal{Q}^{pr}\cup\mathcal{Q}^{se}} x_{t,v,l,0,t^l,q}\right) \le l^{\max}, \tag{2}$$

$$\sum_{q\in\mathcal{Q}^{pr}\cup\mathcal{Q}^{se}} x_{t,v,l,t^e,t^l,q} \le I_{t,v,l,t^e,t^l}, \quad \forall\, v,l,t^e,t^l, \tag{3}$$

$$\sum_{q\in\mathcal{Q}^{pr}\cup\mathcal{Q}^{se}} x_{t,v,l,0,0,q} = I_{t,v,l,0,0}, \quad \forall\, v,l, \tag{4}$$

$$x_{t,v,l,t^e,t^l,q} = 0, \quad t^e > 0, \quad \forall\, v,l,t^l,q, \tag{5}$$

$$\sum_{v,l,t^l} x_{t,v,l,0,t^l,q} \cdot l \le 1, \quad \forall\, q, \tag{6}$$

$$r_{t,q} \le \tau^{\text{MaxRoute}}, \quad \forall\, q, \tag{7}$$

$$x_{t,v,l,t^e,t^l,q} \in \mathbb{N}, \quad \forall\, v,l,t^e,t^l,q. \tag{8}$$

Constraint (2) ensures that at most the maximum inventory remains after dispatching. Observe that only orders with $t^e = 0$ are part of the inventory. We note that one might instead place a bound on the maximum volume with a similar constraint. 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 t are dispatched. Constraint (5) prevents the dispatch of preannounced orders that are not yet in inventory. Constraint (6) ensures that the order volume assigned to a vehicle does not exceed its capacity whereas Constraint (7) prevents violations of the maximum route duration. Recall from Section 3.1 that the route duration is the outcome of routing function Δ. Finally, Constraint (8) ensures that the number of orders of any type dispatched is nonnegative. The set of feasible actions in a given state is described by $\mathcal{X}_t(S_t)$.

3.5. Cost Function

When evaluating a state, we must compute the direct costs $C(S_t, x_t)$ for all actions in $\mathcal{X}_t(S_t)$. We reiterate that we consider a two-phase solution method in which we first make the dispatching decision based on approximate routing costs and subsequently optimize the route for the selected dispatch action. Although we perform some numerical experiments in which we integrate the routing problem in the model and heuristically solve it, our main focus is on the dispatching decision. The approximate routing costs must be computed for every possible action for a large number of states. Therefore, an efficient cost function is required. For this study, we use the approximation formula of Daganzo (1984) to estimate the routing costs.

Daganzo's formula is known to provide good estimates for VRP distances (Robusté, Estrada, and López-Pita 2004). 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 (Daganzo 1984). After determining the total tour length with Daganzo's formula, we determine the number of vehicles required. We assume that the total distance to cover is equally distributed over the dispatched vehicles, thus resulting in identical route durations for all vehicles. We emphasize that these identical route durations only apply to the particular cost function that we define here, not to the model itself. After determining the required travel distance, we incrementally increase the number of vehicles until it satisfies the maximum route duration constraint and there is enough capacity to carry the total volume of dispatched orders. The costs are composed of three elements: a fixed cost per dispatched truck, distance-dependent route costs, and handling costs per location visited. To demonstrate that route durations may vary, we also perform experiments in Section 6 that allow vehicles dispatched at the same time to have different route durations.

The cost function deployed in this paper is subject to several major simplifications. We stress that more advanced expressions exist as well, which provide accurate estimates for more realistic VRPs. Figliozzi (2009) presents expressions for multiple VRP variants, for example, taking into account capacity constraints and time windows at the customer. The parameters for such expressions can be determined a priori by first generating a large number of routes (preferably with the routing algorithm that is eventually used) and then applying regression to estimate the parameters in the expression.

3.6. Transition Function

Based on the order arrivals that may occur during the planning horizon, we can compute a consolidation policy $\pi_t \in \Pi_t$ with $\pi_t\colon S_t \mapsto x_t$ and Π_t being the set of possible policies. To model the uncertainties with respect to the properties of arriving orders, we introduce five random variables. These are (i) the number of orders arriving O_t, (ii) the order destination V, (iii) the order size L, (iv) the earliest dispatch time T^e, and (v) the width of the dispatch window T^{window}. Based on the realizations of these random variables (given by a lowercase representation of the variables), we can deterministically compute the latest dispatch time with

$T^l = T^e + T^{\text{window}}$. The corresponding probability distributions are discrete and finite. To represent all probability distributions with a single variable, we define the exogenous information variable $\tilde{I}_{t,v,l,t^e,t^l} \in \mathbb{N}$, which indicates the number of arrivals of a specific order type after $t-1$ and before t. The generic variable W_t captures all exogenous information, that is, all orders that arrive between $t-1$ and t (i.e., before making decision x_t)

$$W_t = (\tilde{I}_{t,v,l,t^e,t^l})_{v,l,t^e,t^l}, \quad t \geq 1.$$

We now proceed to describe the transition from S_t to the next state S_{t+1}. The transition from S_t to S_{t+1} depends on our dispatching decision $x_t \in \mathscr{X}_t$ 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 orders that are not dispatched, the dispatch window is updated as it is defined relative to the decision moment. For a given action, we have $r_{t,q}$ as the route duration for a dispatched vehicle $q \in \mathcal{Q}^{pr}$. By combining the route durations for dispatched vehicles with the earliest dispatch times stored in Q_t, we can compute Q_{t+1}. This gives us the transition function S^M

$$S_{t+1} = (Q_{t+1}, I_{t+1}) = S^M(S_t, x_t, W_{t+1}), \tag{9}$$

where

$$I_{t+1,v,l,0,t^l} = \sum_{\tilde{t}^e=0}^{1} (I_{t,v,l,\tilde{t}^e,t^l+1}) - \sum_{q \in \mathcal{Q}^{pr} \cup \mathcal{Q}^{se}} x_{t,v,l,0,t^l+1,q} + \tilde{I}_{t+1,v,l,0,t^l}, \quad \forall v,l,t^l, \tag{10}$$

$$I_{t+1,v,l,t^e,t^l} = I_{t,v,l,t^e+1,t^l+1} + \tilde{I}_{t+1,v,l,t^e,t^l}, \quad t^e > 0, \forall v,l,t^l, \tag{11}$$

$$\tau_{t+1,q} = \max(0, \tau_{t,q} - 1, r_{t,q} - 1), \quad \forall q. \tag{12}$$

Constraint (10) states that the number of an order type characterized by $t_e = 0$ at $t+1$ is the number of the order type that we have in S_t minus the number of the order type that is dispatched at t plus the number of the order type that arrives between t and $t+1$. Constraint (11) is similar, but it takes into account that orders with $t^e > 0$ cannot be shipped. Constraint (12) ensures that Q_{t+1} is consistently updated.

3.7. Objective Function

The objective when solving the problem is to minimize the total dispatching costs over the full planning horizon: $\sum_{t\in\mathcal{T}} C(S_t, x_t)$. The costs corresponding to $t > 0$ cannot be computed deterministically beforehand as we do not know which orders will arrive within the planning horizon. Instead, we minimize the expected future costs, summing over all possible future outcomes. Let Ω_t be the set of all possible order arrivals and let $\omega_t \in \Omega_t$ be a particular realization of order arrivals. The random variable W_t may be any realization ω_t. We now introduce the optimality equation

$$V_t(S_t) = \min_{x_t \in \mathscr{X}_t(S_t)} \left(C(S_t, x_t) + \sum_{\omega_{t+1} \in \Omega_{t+1}} \mathbb{P}(W_{t+1} = \omega_{t+1}) \cdot V_{t+1}(S_{t+1} \mid S_t, x_t, \omega_{t+1}) \right). \tag{13}$$

We describe the expression for $\mathbb{P}(W_t)$. For $t \geq 1$, let o_t be the number of orders arriving between $t-1$ and t with $o_t \in \{0, 1, \ldots, o_t^{\max}\}$. We denote the probability of o_t orders arriving 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 $\mathbb{P}(V, L, T^e, T^l)$. For new arrivals $\mathbb{P}(W_t = \omega_t)$, with $t \geq 1$, the probability function is given by

$$\mathbb{P}(W_t = \omega_t) = \mathbb{P}(O_t = o_t) \frac{o_t!}{\prod_{\tilde{I}_{t,v,l,t^e,t^l} \in \omega_t} \tilde{I}_{t,v,l,t^e,t^l}!} \cdot \prod_{v,l,t^e,t^l} \mathbb{P}(V = v, L = l, T^e = t^e, T^l = t^l)^{\tilde{I}_{t,v,l,t^e,t^l}}. \tag{14}$$

4. Solution Method

For realistic instance sizes, our decision problem becomes intractably large in terms of state space, action space, and outcome space. Powell (2011, p. 3) refers to this phenomenon as the "three curses of dimensionality" and provides an ADP framework that addresses these problems. Based on this framework, we develop an ADP algorithm that we use to solve our DDP-TW. ADP is a simulation-based solution method to solve instances of problems represented by a Markov decision model. During an offline learning phase, we learn estimates for the downstream costs associated with state–action pairs. This is done by performing a Monte Carlo simulation in which we repeatedly (i) step forward in time, (ii) sample a random order arrival scenario, and (iii) learn the value of being in a given state by observing its associated costs. The goal is to obtain the optimal consolidation policy, which can subsequently be used for online decision making. In Sections 4.1 to 4.3, we discuss how we address the three curses of dimensionality. We give an overview of the ADP algorithm in Section 4.4. Section 4.5 concludes the description of our solution method with the design of the basis functions.

4.1. Dimensionality of the Outcome Space

We start by addressing the size of the outcome space. In traditional dynamic programming, when evaluating a given state–action pair, we compute its downstream costs by multiplying the value for every possible outcome state S_{t+1}, given (S_t, x_t, ω_{t+1}), with the probability that arrival scenario ω_{t+1} occurs, see Equation (13).

In our problem, the number of order arrival scenarios increases exponentially with the maximum number of arrivals. For realistic numbers of arrivals, this results in very large outcome spaces. To avoid having to enumerate the complete outcome space, 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 order arrivals ω_{t+1}. 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

$$S_t^x = S^{M,x}(S_t, x_t), \tag{15}$$

where

$$I^x_{t,v,l,t^e,t^l} = I_{t,v,l,t^e,t^l} - \sum_{q \in \mathcal{Q}^{pr} \cup \mathcal{Q}^{se}} x_{t,v,l,t^e,t^l,q}, \quad \forall\, v, l, t^e, t^l, \tag{16}$$

$$\tau^x_{t,q} = \max(r_{t,q}, \tau_{t,q}), \quad \forall\, q \in \mathcal{Q}^{pr}. \tag{17}$$

By attaching the downstream costs to the post-decision state instead of the next pre-decision states, we replace the probabilistic expression for the downstream costs with a single cost function $V_t^x(S_t^x)$

$$V_t^x(S_t^x) = \mathbb{E}\{V_{t+1}(S_{t+1}) \mid S_t^x\} = \sum_{\omega_{t+1} \in \Omega_{t+1}} \mathbb{P}(W_{t+1} = \omega_{t+1}) \cdot V_{t+1}(S_{t+1} \mid S_t, x_t, \omega_{t+1}). \tag{18}$$

For our problem, the post-decision state comprises (i) the number of orders per type remaining at the consolidation center and (ii) the updated vehicle availability after our dispatching decision. In the offline learning phase, we operate on the post-decision state as follows. For a given state S_t, we make a decision x_t to arrive at the post-decision state S_t^x. Then, we randomly sample an order arrival $\omega_{t+1} \in \Omega_{t+1}$ and observe the costs of the next state S_{t+1}. We use these costs to improve our estimate of the downstream cost $V_t^x(S_t^x)$; ultimately, the estimate is based on a limited number of samples from Ω_{t+1} instead of explicitly calculating the expectation over all possible realizations. By using an estimate of the value of the post-decision state rather then the values of all possible pre-decision states at $t+1$, we only need to evaluate a single outcome per action.

4.2. Dimensionality of the State Space

We proceed to address the problem of the large state space. We learn values by observing states in the offline learning phase, which we encounter by randomly sampling order arrivals. Suppose that we would use a lookup table containing estimates of the downstream costs for each post-decision state. Filling this table implies that we should visit every post-decision state at least once to estimate its associated downstream costs. As the size of our state space grows exponentially with the number of order types, for realistically sized instances, we would need to perform unfeasibly many simulation iterations to learn all these values. Therefore, we replace $V_t^x(S_t^x)$ with a function $\bar{V}_t^n\colon S_t^x \mapsto \mathbb{R}, S_t^x$ that approximates the downstream costs of all post-decision states regardless of whether we have visited a state previously. The key benefit of this approach, which is known as the value function approximation (VFA) for the downstream cost (Powell 2011), is that we no longer need to fill a lookup table with the expected downstream costs for all states. We apply value iteration to learn $\bar{V}_t^n(S_t^x)$, which is our estimate of $V_t^x(S_t^x)$ resulting from the nth iteration.

Before applying the learned policy in an online setting, the parameters of $\bar{V}_t^n$ must be learned offline via observations. In our solution method, $\bar{V}_t^n$ contains a number of variables that explain the cost structure of the problem, using so-called basis functions (Powell 2011). Adopting the ADP terminology, we refer to an explanatory variable as a "feature." Let $\mathcal{F}$ be a set of features (which we can compute based on the post-decision state) with $f \in \mathcal{F}$ representing an explanatory variable for the true value function, for example, the number of primary trucks available or the volume per receiver. Furthermore, let $\phi_f\colon S_t^x \mapsto \mathbb{R}$ be a basis function of feature f. Such a basis function is often equal to the feature itself but may also be a polynomial or another mathematical operation performed on the feature, for example, the number of primary vehicles squared. A set of basis functions is denoted as $\phi \supset \phi_f(S_t), \forall\, S_t \in \mathcal{S}$. Such a set may be viewed as an aggregate state description, retaining sufficient detail to capture the cost structure, but requiring fewer variables to do so. For example, the number of primary vehicles available at $t+1$ and the volume per receiver held in the consolidation center may be good indicators to estimate the downstream costs while requiring only $1 + |\mathcal{V}|$ variables instead of the full state description. In Section 4.5, we discuss the basis function design for our model. Finally, let $\theta^n_{t,f}$ be a weight corresponding to ϕ_f at decision moment t. As we consider a finite horizon problem, at each decision moment $t \in \mathcal{T}$ the downstream costs are computed over the time interval $t+1, \ldots, T$. It follows that downstream costs are time-dependent. Therefore, we learn weights separately for each decision moment $t \in \mathcal{T}$. We use the iteration counter n to track the number of iterations performed, and we use $\bar{V}_t^n(S_t^x)$ to denote our VFA during the offline learning phase. We obtain the VFA

$$\bar{V}_t^n(S_t^x) = \sum_{f \in \mathcal{F}} \theta^n_{t,f} \phi_f(S_t^x) \quad \forall\, t \in \mathcal{T}. \tag{19}$$

To improve our estimate of the downstream costs, we update the weights $\theta^n_{t,f}$, $f \in \mathcal{F}$, $t \in \mathcal{T}$ after every iteration n. For the weight updates, we use a regression procedure, which we discuss in Section 4.5.4. As we use a single set of weights to estimate all downstream costs, new observations adjust our cost estimates for all states that we may encounter.

4.3. Dimensionality of the Action Space

The final curse of dimensionality that we address relates to the action space. We have replaced the original value function with a function that estimates the downstream costs based on the post-decision state. As a consequence, our decision problem reduces to a deterministic minimization problem. By solving this problem at decision moment t in iteration n, using our most recent estimate $\bar{V}_t^{n-1}(S_t^x)$ that followed from iteration $n-1$, the best action $\tilde{x}_t^n$ is found

$$\tilde{x}_t^n = \arg\min_{x_t \in \mathscr{X}_t(S_t)} \big(C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x)\big). \tag{20}$$

Although the computational effort required to solve the decision problem sharply decreases by operating on the post-decision state rather than the full outcome space, we are still required to evaluate every action $x_t \in \mathscr{X}_t(S_t)$. For our problem, the action space increases exponentially with the number of orders. Therefore, complete enumeration of all decisions is not feasible for large instances. However, as we have transformed the decision problem into a discrete and deterministic problem, we are able to write it as an ILP that can be solved within the ADP for large action spaces. This ILP solves Equation (20) and is subject to both the action constraints (2)–(6) and the post-decision constraints (16)–(17). The post-decision variables are required to compute the basis functions. As ILPs can be solved relatively quickly with modern solvers, we can handle much larger decision problems than enumeration would allow. However, the decision problem remains computationally challenging because of the need to precompute $2^{|\mathscr{V}|}$ route durations to cover all subsets of destinations that might be visited; a corresponding decision variable must be assigned to each unique route. We therefore propose an additional simplification. Rather than explicitly computing all possible route distances, we define a set of approximate travel distance classes as input for the ILP. The formulation of the ILP enforces that the approximate distance used to compute the routing costs in the ILP is at least equal to the distance computed with Daganzo's formula. The more distance classes that we predefine, the better we can approximate the actual travel distance. To determine the number of distance classes, we measure the performance gap in preliminary numerical experiments. With the ILP, we can solve our large instances within a reasonable time and closely approximate the solutions from Daganzo's formula. The full ILP model is shown in the online appendix.

4.4. Algorithmic Outline

We have explained how we handle the dimensionality of the outcome-, state-, and action spaces of our problem. Algorithm 1 provides the outline of our ADP algorithm; a detailed description of the full procedure can be found in Powell (2011, p. 392). To update the value functions, we make use of a backward pass procedure (Sutton and Barto 1998) in which the VFAs are updated only after completing a full forward iteration. With every iteration n, we generate a random order arrival path $\omega^n = \{\omega_1, \ldots, \omega_T\}$. At every $t \in \{0, \ldots, T-1\}$, we solve Equation (20) to obtain the action that minimizes the sum of direct costs and estimated downstream costs. However, in the offline learning phase, it is advisable to also select different actions that allow the observation of new states and improvement of the VFA. Especially in the early iterations, our estimates for the downstream costs may be poor because of the low number of states observed. As a consequence, the algorithm might continuously visit the same suboptimal post-decision states while never learning the values of better states. To avoid this form of cycling, we do not always select the "best" action. Instead, with probability $\epsilon \in [0, 1]$, we select a random action $x_t \in \mathscr{X}_t(S_t)$. This so-called ϵ-greedy strategy allows us to explore new states (Powell 2011). However, too much exploration will decrease the quality of the estimates for the (near-)optimal states that we are interested in. We therefore test various settings for ϵ such that we find the right balance between visiting new states and accurately learning the values of good states. After selecting an action, we generate a random order arrival $W_{t+1}(\omega^n)$, and find the next state S_{t+1} by solving Equation (9). This procedure is repeated until reaching $t = T$. After completing the forward iteration, we move backward to recursively update $\bar{V}_t^{n-1}(S_t^x)$ based on the observed downstream costs $\hat{v}_t^n = C_t(S_t, x_t) + \hat{v}_{t+1}^n$. We perform this update with a regression procedure that we discuss in Section 4.5.4; the updating function is defined as follows:

$$\bar{V}_{t-1}^n(S_{t-1}^x) \longleftarrow U^V(\bar{V}_{t-1}^{n-1}(S_{t-1}^x), S_{t-1}^x, \hat{v}_t^n). \tag{21}$$

Algorithm 1 (ADP backward pass algorithm with post-decision states)

```
Step 0. Initialize
    Step 0a: Initialize $\bar{V}_t^0(S_t)$, $t \in \mathscr{T}$, $S_t \in \mathscr{S}$.
    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 1.
    Step 1a: Select an initial state $S_0^n$.
    Step 1b: 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 (20).
             With probability $\epsilon$, randomly select
                 $x_t \in \mathscr{X}_t$.
    Step 2b: Obtain post-decision state $S_t^x$ via
                 Equation (15).
    Step 2c: Obtain sample realization $W_{t+1}(\omega^n)$,
                 calculate $S_{t+1}$ with Equation (9).
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 (21).
             This step returns the weights $\theta_{t,f}^n$,
                 $f \in \mathscr{F}, t \in \mathscr{T}$
Step 4. Set $n := n + 1$.
        If $n \leq N$, then go to Step 1.
Step 5. Return $\bar{V}_t^N(S_t^x)$, $t \in \mathscr{T}$.
```

4.5. Basis Function Design

We conclude this section with the design of the basis functions. We discuss the following design-related topics: (i) a categorization of the basis functions for the DDP-TW, (ii) the selection of a set of basis functions, (iii) the pitfalls of nonsegregated basis functions, and (iv) the updating procedure to learn the weights corresponding to basis functions. Finally, we present the selected sets of basis functions that we test in our numerical experiments.

4.5.1. Categorization of Basis Functions. The selected set of basis functions should accurately represent the downstream costs. Hence, the basis functions need to capture the cost structure of the problem. In Section 5, we introduce the sets of basis functions that we test. For now, we restrict ourselves to presenting a categorization of our basis functions. 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. The basis functions should reflect one or more of these properties. It is important that basis functions within a set are independent of each other; as we learn the weights by means of regression, we must avoid collinearity. We divide our basis functions into three categories that are assumed to be independent: vehicle-based, location-based, and volume-based. A time component may be added to each category, for example, computing vehicle availability up until the maximum route duration. By carefully selecting a set of basis functions that incorporates these factors, we are able to estimate the values of post-decision states, even those we did not visit before. As such, the VFA enables us to cope with large state spaces.

4.5.2. Selecting Basis Functions. A key difficulty in the design of a value function approximation is to select a set of basis functions that enables us to accurately approximate the downstream costs. Given our categorization of basis functions that capture (part of) the cost structure, we may conjecture suitable basis functions, yet their usefulness must be numerically validated. We obtain a first sense of good basis functions by performing regression analysis on the exact values of states; these values are obtained by solving small instances to optimality. We compute the R^2 values (i.e., the coefficient of determination that describes the fit of the model) of various sets of basis functions to evaluate how well they explain the costs.

Although the R^2 values give an initial insight into the quality of basis functions, a high explanatory value does not guarantee that the corresponding policy that we learn also performs well. There are three key reasons for this. First, as we only observe a sample of the state space, the weights that we learn with ADP may differ from those obtained with regression analysis on the values of all states. Second, we can only compute exact values for small instances; these instances do not capture all of the dynamics embedded in the problem. Third, correctly estimating the downstream costs is important, yet good policies may be obtained without accurate estimates of the downstream costs (Chang et al. 2013). Essentially, we require estimates that allow us to rank the actions correctly, such that we are able to consistently select good actions. For these reasons, we also compare the performances resulting from our basis function designs.

4.5.3. Use of Nonsegregated Basis Functions. Nonsegregated basis functions (e.g., the ratio of vehicle capacity and volume per dispatch time) combine several features and therefore may have a higher explanatory value than segregated functions. However, they are difficult to combine with other basis functions because of the high probability of introducing collinearity. Another drawback is that the associated weights may be more difficult to learn as we try to simultaneously learn the impact of multiple explanatory variables. Furthermore, we also incorporate the basis functions within the ILP formulation. This is straightforward for linear basis functions, but the use of nonlinear basis functions potentially requires a large number of artificial variables. In the design of our basis functions, we therefore focus on basis functions that are both linear and segregated.

4.5.4. Regression Procedure to Update the Weights. After every simulation iteration, we must update the weights corresponding to the basis functions to improve our VFA. We do this by regressing the VFA on the observed data. The regression procedure that we use is known as recursive least squares for nonstationary data; Powell (2011, p. 352) provides a detailed description of this method. A shortcoming of this procedure is that it is sensitive to outliers. Unstable behavior of the regression is common for large sets of basis functions. This instability follows from observing deviating values, to which the regression procedure responds by assigning a very large weight to a basis function of small magnitude. Subsequent observations may then yield extreme estimates. Burger and Repiský (2012) discuss these problems as well as various other pitfalls in least squares regression. In the design of our basis functions, we aim to keep the number of basis functions relatively low; based on preliminary experiments, we consider sets that contain 10 or fewer basis functions as sufficiently small. Furthermore, by using only additive basis functions—rather than, for example, polynomials, multiplications, and divisions—we prevent basis functions from returning extremely small values; the smallest nonzero value that we may obtain relates to volume-based basis functions and is equal to $\min(l \mid l \in \mathscr{L})$. All other basis functions use integer counters, such that their smallest nonzero value is 1.

4.5.5. Selected Basis Functions. Having discussed the categories of basis functions for our problem and the design requirements, we now introduce the sets of basis functions that we test in our numerical experiments. Our selected basis functions reflect one or more of these categories; we first test sets containing separate basis functions and then sets that combine multiple aspects. We now describe the 13 sets of basis functions that we test in our numerical experiments with the aim of finding the basis functions that yield the best ADP policy. The sets are shown in Table 1. Every set of basis functions ϕ contains a constant to capture the average costs of a state. The set ϕ^1 contains only such a constant; note that this set assigns the same value to every post-decision state as such yielding the direct cost-minimization policy $\pi^{\min}$. In set ϕ^2, we consider the total number of destinations that we may visit at the current decision moment t. Set ϕ^3 describes the number of primary vehicles available from t to $t + \tau^{\text{MaxRoute}}$. Set ϕ^4 gives the volume per latest dispatch moment, for example, the urgent volume at various decision moments. In ϕ^5, we keep track of the volume per destination. Set ϕ^6 contains the number of orders per order type. As this set does not aggregate state data, it is not a well-designed set of basis functions; this set is added only for illustrative purposes. The remainder of the sets (ϕ^7–ϕ^{13}) are either linear combinations or nonsegregated combinations of the aforementioned sets. We stress that nonsegregated basis functions do not meet all our design requirements and are included only for the sake of illustration.

5. Experimental Setup

In this section, we describe the setup of our numerical experiments. The main goal of the experiments is to identify the set of basis functions that enables us to most accurately capture the cost structure of our problem and show how the resulting policy that we learn performs relative to a variety of benchmark policies. We distinguish between experiments in the offline learning phase and the online phase. In the offline learning phase, we evaluate the sets of basis functions as defined in Section 4.5.5. For each set of basis functions, we learn the corresponding dispatching policy; subsequently, we compare the performances of these policies with each other. In the online phase, we test the performance of the ADP policy obtained with the best set of basis functions only, relative to a number of benchmark policies. For large instances, we also report the computational times required to solve the decision problem.

We proceed to describe the main characteristics of the test instances. Customer locations are defined on a 10 km × 10 km area. The travel distances between locations are computed using Manhattan distances. We use a time horizon of $T = 10$ for all experiments. We divide our experiments into three classes—small (S), medium (M), large (L)—each class having its own purpose. Preliminary tests on medium-sized instances indicate that 5,000 iterations suffice to learn a policy; further observations do not significantly increase the quality of policies. We test the online performance of policies using 1,000 iterations. In the offline learning

Table 1. Selected Sets of Basis Functions

Set	Basis functions	No. of basis functions
ϕ^1	Const.	1
ϕ^2	Const. + # available destinations at $t^e = 0$	2
ϕ^3	Const. + # vehicles	$1 + \tau^{\text{MaxRoute}}$
ϕ^4	Const. + Total vol. per t^l	$1 + \tau^{\text{LatestDispatch}}$
ϕ^5	Const. + Total vol. per v	$1 + \lvert\mathscr{V}\rvert$
ϕ^6	Const. + # orders per type	$1 + \lvert\mathscr{I}\rvert$
ϕ^7	Const. + # vehicles/# available destinations	2
ϕ^8	Const. + # vehicles/Total vol. per t^l	$1 + \min(\tau^{\text{MaxRoute}}, \tau^{\text{LatestDispatch}})$
ϕ^9	Const. + # destinations per t^l	$1 + \tau^{\text{LatestDispatch}}$
ϕ^{10}	Const. + Vol. per destination per t^l	$1 + (\tau^{\text{LatestDispatch}}) \cdot \lvert\mathscr{V}\rvert$
ϕ^{11}	Const. + # available destinations + Total vol. per t^l	$2 + \tau^{\text{LatestDispatch}}$
ϕ^{12}	Const. + # vehicles + Total vol. per t^l	$1 + \tau^{\text{MaxRoute}} + \tau^{\text{LatestDispatch}}$
ϕ^{13}	Const. + # available destinations + # vehicles per τ_t + Total vol. per t^l	$2 + \tau^{\text{MaxRoute}} + \tau^{\text{LatestDispatch}}$

phase, we select a random action with probability $\epsilon = 0.05$. Preliminary tests on a representative instance indicated this as the best-performing setting, having tested $\epsilon \in \{0, 0.05, \ldots, 0.25\}$. The policy obtained under $\epsilon = 0$ performs the worst with a difference in solution quality of 3.3% compared with $\epsilon = 0.05$. Our algorithm is coded in Delphi XE6, and we use a computer with a 2.9 GHz Intel Core i7 processor and 8 GB RAM. The ILP is solved with CPLEX 12.2.

The outline of the remainder of this section is as follows. In Section 5.1, we define the properties of all our test instances. Section 5.2 describes the benchmark policies that we use to evaluate the performance of the policies that we learn with ADP. In Section 5.3, we describe the procedure that we use to generate a set of representative initial states from which we draw at the start of every iteration.

5.1. Instance Properties

In this section, we describe the properties of the instances on which we perform experiments. We define two small instances, 20 medium-sized instances, and two large instances. With our experiments on small instances, we primarily evaluate the behavior of ADP in the offline learning phase, comparing the results with the exact results. We perform tests on two small instances, named S1 and S2. For S1, we consider three customers located at equal distances from the depot while for S2 the depot–destination distances vary. We consider a dispatch window of width 1 and no preannounced orders. The instances are sufficiently small (93,000 states) to be solved exactly with dynamic programming. With the obtained values, we compute the explanatory power of the basis functions by computing the R^2 value and unveil possible dependencies between them. We do this for multiple sets of basis functions and then test how closely we approximate the exact values with the corresponding policies.

Our experiments on the medium-sized instances (# states $\gg 10^{22}$) provide insights into both the offline learning phase and the online phase. Table 2 shows the main properties of these instances. We consider up to 15 arrivals per decision moment and up to 30 orders in inventory. We are primarily interested in the performance of ADP in relation to two properties: (i) the degree of dynamism (DoD) and (ii) the flexibility in dispatching times. The degree of dynamism is the ratio between the number of orders with the property $t^e = 0$ (i.e., not announced in advance) and the total number of orders (Larsen, Madsen, and Solomon 2002). A DoD of 1 indicates that all orders become known dynamically whereas a DoD of 0 implies that all orders are preannounced. With the flexibility in dispatching times, we refer to the maximum width of the dispatch windows of orders. The first six instances

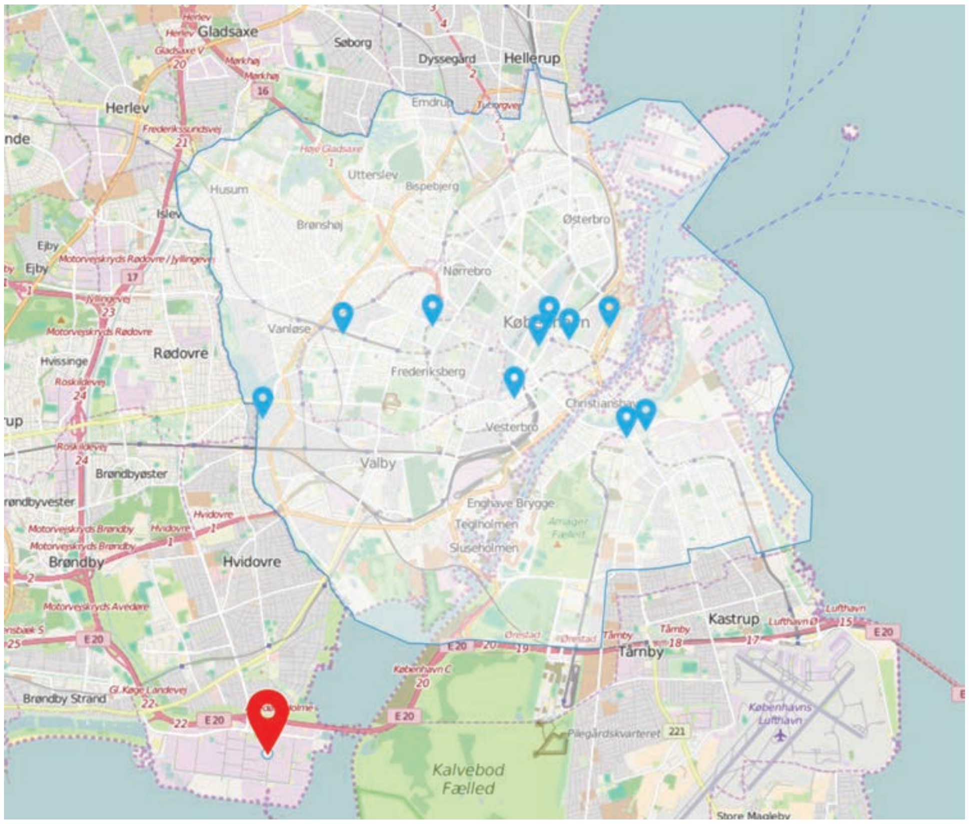
Table 2. Settings for the Medium-Sized Instances

Instance	Description
M1	DoD = 1, $\tau^{\text{MaxWindow}} = 1$
M2	DoD = 0.5, $\tau^{\text{MaxWindow}} = 1$
M3	DoD = 0, $\tau^{\text{MaxWindow}} = 1$
M4	DoD = 1, $\tau^{\text{MaxWindow}} = 2$
M5	DoD = 0.5, $\tau^{\text{MaxWindow}} = 2$
M6	DoD = 0, $\tau^{\text{MaxWindow}} = 2$
M7	As M2, two customers 50% demand
M8	As M5, two customers 50% demand
M9	As M2, but with last 10 customers R101
M10	As M5, but with last 10 customers R101
M11	As M2, but with four vehicles
M12	As M5, but with four vehicles
M13	As M2, but with unlimited fleet size
M14	As M5, but with unlimited fleet size
M15	As M2, but with 50 km by 50 km network
M16	As M5, but with 50 km by 50 km network
M17	As M2, but with routing algorithm
M18	As M5, but with routing algorithm
M19	As M17, but with Copenhagen network
M20	As M18, but with Copenhagen network

(M1–M6) vary the DoD and the flexibility in dispatching times. We take the first 10 customers of the 25-customer Solomon instance R101 scaled to a 10 km × 10 km area. We consider instances with maximum dispatch windows of 1 and 2, respectively, providing us with two levels of flexibility. Furthermore, we vary the DoD, testing ratios of 0, 0.5, and 1. The remaining instances are variants of M2 and M5, in which we test the effects of skewed demand (M7–M8), different customer locations (M9–M10), variations in fleet sizes (M11–M14), distances scaled to a 50 km × 50 km area (M15–M16), and tests in which we use a routing algorithm within the ADP method (Clarke–Wright savings algorithm with two-opt) instead of using Daganzo's formula (M17–M18). Finally, in M19 and M20, we test more realistic instances inspired by the city of Copenhagen; we refer to van Heeswijk, Larsen, and Larsen (2017) for a detailed description of the instance properties. We take the location of an existing urban consolidation center, 10 real customer locations within the low-emission zone of the city of Copenhagen (see Figure 1) and compute the corresponding distance matrix using the OpenStreetMap implementation of Luxen and Vetter (2011). We solve these instances with the routing algorithm as used in M17–M18.

Finally, we perform simulation experiments on two large instances (# states $\gg 10^{178}$). For these instances, we consider up to 40 arrivals per decision moment and inventories up to 60 orders. We test two instances: the 25-customer Solomon instance R101 (L1) and the 50-customer Solomon instance R101 (L2), scaled to a 10 km × 10 km area. The results in the online phase are compared with those of the benchmark policies that we describe in Section 5.2. The contribution of these

Figure 1. (Color online) Copenhagen Test Instance, Showing the Low-Emission Zone (Shaded Area), the Location of the Consolidation Center (Large Red Marker), and Customer Locations (Blue Markers)



Source. Adapted from OpenStreetMap, © OpenStreetMap contributors.

experiments is to demonstrate that realistically sized instances can be handled by applying the ILP for the decision problems. Finally, we provide insights into the computation times required to solve the ILP.

5.2. Benchmark Policies

This section defines the benchmark policies that we use to evaluate the performance of our ADP policies. We introduce three myopic policies and one lookahead policy.

The small instances of our problem can be solved to optimality, giving us an exact benchmark for our ADP method. However, these instances do not capture all of the dynamics embedded in the problem whereas medium-sized instances cannot be solved exactly. To evaluate the online performance, we compare ADP with three consolidation policies without lookahead as well as a rollout policy that incorporates a downstream cost estimate based on randomly generated future orders (Goodson, Thomas, and Ohlmann 2017). 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 sorting mechanism gives priority to orders that are more urgent and, for equally urgent orders, prioritizes smaller volumes. The third myopic policy is $\pi^{\min}$, 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 the 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.

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

```
Step 0  Sort orders.
   Step 0a: Sort available orders based on lowest $t^l$.
   Step 0b: Sort available orders with equal $t^l$ based on
            smallest size.
Step 1  While orders with $t^l = 0$ are unassigned do:
            Assign order with $t^l = 0$ to vehicle.
Step 2  While remaining inventory exceeds $l^{\max}$ do:
            Assign first capacity-feasible order on list
            to vehicle.
Step 3  If $\pi^{\text{direct}}$: 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.

We proceed to explain the lookahead policy π^{rollout}. We apply this benchmark policy to the medium-sized instances; later on, we explain why we do not use this lookahead policy for the large instances L1 and L2. The idea behind π^{rollout} is that we generate a number of sample arrival paths and heuristically solve the decision problems for these paths, thereby obtaining an estimate of the downstream costs. This benchmark policy is comparable to the rollout procedures as presented in, for example, Goodson, Ohlmann, and Thomas (2013) and Goodson, Thomas, and Ohlmann (2017). More precisely, our benchmark policy may be viewed as a version of the post-decision rollout that is described in Goodson, Thomas, and Ohlmann (2017), in which we look ahead for multiple time steps (in our implementation, we vary with horizons of 1, 2, and 5). To apply π^{rollout} at a given decision moment, we first generate a set Ω^m, which contains m random arrival paths of length τ^{rollout}, that is, $\omega^m = \{\omega^m_{t+1}, \ldots, \omega^m_{t+\tau^{\text{rollout}}}\}$. We then enumerate all actions x_t—meaning that the action space should be sufficiently small—to obtain the direct costs $C(S_t, x_t)$, $x_t \in \mathscr{X}_t$. Next, we compute the costs associated with 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 are calculated recursively for $t' = t+1, \ldots, t+\tau^{\text{rollout}}$ using

$$\tilde{V}_{t'}(S'_t, \omega^m) = C(S_{t'}, x_{t'}) + \tilde{V}_{t'+1}(S_{t'+1}, \omega^m), \tag{22}$$

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

$$\hat{V}_t(S^x_t) = \frac{\sum_{\omega^m \in \Omega^m} \tilde{V}_{t+1}(S_{t+1}, \omega^m)}{|\Omega^m|}. \tag{23}$$

After estimating the post-decision values, we obtain the best decision by solving

$$\operatorname*{arg\,min}_{x_t \in \mathscr{X}_t(S_t)} = C(S_t, x_t) + \hat{V}_t(S^x_t). \tag{24}$$

We do not compare the performance of ADP with post-decision rollout in L1 and L2 as the procedure is not directly scalable to large action spaces. To incorporate the downstream costs of the rollout procedure into the ILP, we need to compute the expected downstream costs for each decision. Although VFA only requires embedding a number of variables equal to the number of basis functions into the ILP, the rollout procedure requires an artificial variable and corresponding downstream costs for each decision. As the ILP is designed to handle outcome spaces that are too large to enumerate, we are also unable to pregenerate the downstream costs for all reachable post-decision states. We might be able to compute a lower number of downstream costs by either assigning downstream costs to aggregated post-decision states or by reducing the decision space. However, the design of such benchmark policies would justify a separate study, which is beyond the scope of this paper.

5.3. Generating a Set of Initial States

In this section, we describe the procedure to generate a set of initial states from which we draw at the start of each iteration. To learn a policy with ADP based on a given initial state, we might 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 might not repeat the entire offline learning phase for every state we encounter. Instead, we want to perform the offline 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 may result in policies based on states that we rarely encounter in practice. To obtain a set of realistic initial states $\mathscr{S}' \subset \mathscr{S}$, we perform a predefined number of simulation iterations with π^{direct}—starting every iteration with a randomly generated initial state—and store the state encountered at $T/2$ (thereby avoiding possible warm-up and cool-down effects) in $\mathscr{S}'$. Next, at the start of each iteration n, we select $S^n_0 \in \mathscr{S}'$ with a probability proportional to the frequency it was observed. As order arrivals are random, we still visit states $S_t \notin \mathscr{S}'$. However, by learning based on commonly encountered states, the results provide a better fit in practical settings.

6. Numerical Results

This section presents the numerical results of our experiments. First, we show the results for small instances, then for medium-sized instances, and finally for large instances. The presentation of the results for the offline learning phases is focused on the identification of suitable sets of basis functions. When discussing the results for the online phases, we compare the performance of the best ADP policy (using the most suitable set of basis functions) with various benchmark policies. For each class of instances, we provide a separate outline of the results that we discuss.

6.1. Experiments on Small Instances

We present the results of the offline learning phase and the online phase. In the offline learning phase, we test various sets of basis functions for which we assess how well they approximate the optimal outcomes obtained with dynamic programming while also discussing the properties of good sets of basis functions.

In the online phase, we apply the dispatching policy that we learned for the best set of basis functions and compare the results obtained with this policy with the optimal outcomes.

6.1.1. Results in the Offline Learning Phase. To assess the explanatory value for various sets of basis functions, we compute their R^2 values and test how these sets perform in the offline learning phase. Using the procedure described in Section 5.3, we obtain a state set $\mathcal{S}'$. For each simulation run, we use a state in $\mathcal{S}'$ as the initial state and learn the corresponding policy. We consider a set of basis functions as a candidate set to apply on larger instances if it satisfies the following criteria: (i) the basis functions in the set are independent of each other, (ii) 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) the numerical outcomes are robust for all states in S', and (vi) the average value gap (difference between optimal value and estimated value) is small.

The results for our sets of basis functions are shown in Table 3. We observe some discrepancies between the R^2 values and the actual performance of the corresponding policies that we learned. In general, we see that sets including the volume per t^l yield estimates of good quality, which indicates that this property well explains the ranking of values of states. During the experiments, we observed that combining more than three types of basis functions often results in unstable behavior. Similar instabilities are found when combining multiple basis functions of the same type, for example, 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. The set ϕ^{13} yields the best results of all tested sets. This set consists of the following basis functions: a constant, the number of available destinations, the number of vehicles available per dispatch time $t+1,\ldots,\tau^{\text{MaxRoute}}$, and the total volumes sorted per latest delivery time. We note that the set captures the key properties of the cost structure as listed in Section 4.5 while only requiring a total of four basis functions for the small instances to estimate the downstream costs with an error of less than 3%. By contrast, a set such as ϕ^6 contains many more basis functions but yields considerably worse estimates.

6.1.2. Results in the Online Phase. We illustrate the ADP performance using the policy learned with the set ϕ^{13} applied on S1, comparing ADP estimates to exact values for the states in $\mathcal{S}'$. Figure 2 shows the distribution of absolute deviations from the optimal value. For virtually all states, ADP yields estimates that are somewhat higher than the optimal value with an average absolute deviation of 1.9%. The tendency to

Table 3. R^2 Values and Value Gaps for Various Sets of Basis Functions

		S1		S2	
Set	Basis functions	R^2	Avg. gap (%)	R^2	Avg. gap (%)
ϕ^1	Const.	—	6.76	—	6.14
ϕ^2	Const. + # available destinations at $t^e=0$	0.16	2.80	0.14	3.57
ϕ^3	Const. + # vehicles	0.34	6.77	0.31	6.14
ϕ^4	Const. + Total vol. per t^l	0.35	2.11	0.32	2.78
ϕ^5	Const. + Total vol. per v	0.27	2.97	0.31	3.34
ϕ^6	Const. + # orders per type	0.35	5.27	0.40	5.36
ϕ^7	Const. + # vehicles/ # available destinations	0.41	5.63	0.37	6.21
ϕ^8	Const. + # vehicles/ Total vol. per t^l	0.23	6.48	0.20	6.70
ϕ^9	Const. + # destinations per t^l	0.14	2.92	0.12	3.50
ϕ^{10}	Const. + Vol. per destination per t^l	0.35	2.97	0.39	3.34
ϕ^{11}	Const. + # available destinations + Total vol. per t^l	0.48	2.05	0.43	2.76
ϕ^{12}	Const. + # vehicles + Total vol. per t^l	0.69	2.31	0.62	2.92
ϕ^{13}	Const. + # available destinations + # vehicles per τ_t + Total vol. per t^l	0.82	2.03	0.74	2.75

overestimate costs follows from using suboptimal policies to estimate the downstream costs (Powell 2011). For 0.24% of the states, the estimated and optimal values differ by more than 5%; these deviations cause the tail on the right. In Figure 3, we show the ADP estimates for all $S_0 \in \mathcal{S}'$, plotted against the optimal values. This plot provides insight into the ranking of ADP estimates compared with the true ranking. Ideally, the order of the ADP estimates on the y-axis corresponds with the (true) order on the x-axis. Even if the cost estimates deviate from the optimal values, a good ranking enables distinguishing actions with low downstream costs from actions with high downstream

Figure 2. Histogram of Absolute Differences Between Exact Values and ADP Estimates for All States in $\mathcal{S}'$

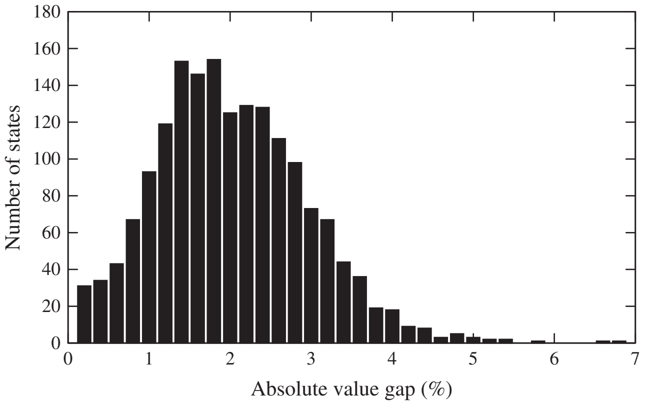
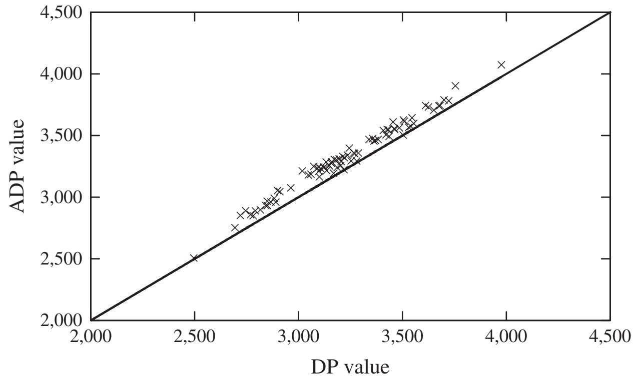


Figure 3. ADP Estimates Plotted Against Exact Values for All States in $\mathcal{S}'$



costs. However, accuracy remains important as well, as we optimize over the sum of direct costs and estimated downstream costs. Generally, we see that the ranking is rather well preserved and that the estimates are close to the true values; ADP will not select actions that deviate much from the optimum.

6.2. Experiments on Medium-Sized Instances

We present the results of the offline learning phase and the online phase. In the offline learning phase, we test the performance for various sets of basis functions to evaluate their performance under increased complexity, this time comparing with the direct cost-minimization policy (i.e., set ϕ^1). In the online phase, we take the best set of basis functions to obtain our ADP policy; we compare this policy with a variety of benchmark policies on 20 different instances. Based on the results, we describe the instance properties under which ADP performs best. Furthermore, we discuss how the performance of ADP relates to both the performance of myopic policies and the rollout policy.

6.2.1. Results in the Offline Learning Phase. In our medium-sized instances, the number of order types is much larger than in the small instances. Therefore, we test how our sets of basis functions perform under this increased complexity. From Table 3, we take the sets of basis functions for which the corresponding policies yielded a value gap smaller than 5% on both S1 and S2. We repeat the offline learning phase for each set of basis functions meeting this criterion, yielding eight ADP policies. We reevaluate the performance of these sets on M2 and M5 as these may be considered hybrids of the other medium-sized instances. To learn π^{ADP}, we randomly draw an initial state from $\mathcal{S}'$ at every iteration. We measure the performance of all policies relative to the performance using set ϕ^1. The results are shown in Table 4. Similar to the results for the small instances, for both M2 and M5 the best results are obtained with ϕ^{13}. Observe that the relative improvement in M5 is notably stronger than with M2; it appears that ADP performs better when there is more flexibility

Table 4. Performance of Various ϕ on M2 and M5, Relative to ϕ^1

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

in the dispatching decision, that is, when the dispatch windows of orders are wider.

6.2.2. Results in the Online Phase. In this section, we test the performance of ADP policies on the medium-sized instances M1–M20, using the policies obtained with the set of basis functions ϕ^{13}. We randomly draw an initial state from $\mathcal{S}'$ at every iteration. We learn π^{ADP} individually for each medium-sized instance and evaluate the performance to the benchmark policies. We run the rollout policy π^{rollout} for various numbers of sample paths and lengths of the sample path, for example, $\pi^{\text{sample}}_{2,5}$ denotes a policy with two sample paths of five time units each.

The computational results of the simulation phase are shown in Table 5. We highlight the main takeaways from the results. First, we see that ADP consistently and significantly outperforms all myopic benchmark policies. The values obtained by the rollout procedure are closer although ADP typically outperforms the post-decision rollout as well. These results indicate that looking into the future pays off for the DDP-TW. Second, we observe that ADP outperforms rollout by a greater margin when considering wider dispatch windows. Voccia, Campbell, and Thomas (2017) suggest that—because of the larger number of possible arrival paths—a longer lookahead horizon requires more sample paths. This hypothesis appears to be confirmed by our results: when using only two or five sample paths, the rollout policy performs better, on average, with a lookahead interval of one. When comparing ADP to the rollout procedures, the results seem to indicate that ADP with VFA is more useful when considering decisions that have an impact on multiple future decision moments, taking into account the large number of paths required for the rollout procedure. Third, ADP performs best when the problem instance offers sufficient flexibility in terms of dispatch times while vehicle capacity is scarce (M11–M14). Fourth, we see

Table 5. Simulation Performance Benchmark Policies Relative to ADP with ϕ^{13}

Instance	π^{direct} (%)	π^{post} (%)	π^{min} (%)	$\pi^{\text{rollout}}_{2,1}$ (%)	$\pi^{\text{rollout}}_{2,2}$ (%)	$\pi^{\text{rollout}}_{2,5}$ (%)	$\pi^{\text{rollout}}_{5,1}$ (%)	$\pi^{\text{rollout}}_{5,2}$ (%)	$\pi^{\text{rollout}}_{5,5}$ (%)	$\pi^{\text{rollout}}_{20,1}$ (%)	$\pi^{\text{rollout}}_{20,2}$ (%)	$\pi^{\text{rollout}}_{20,5}$ (%)
M1	9.46	10.13	8.92	5.27	5.32	6.33	3.62	3.92	5.08	1.95	0.88	3.04
M2	9.25	9.18	7.86	3.22	4.49	5.91	4.00	3.76	4.63	3.62	1.61	2.05
M3	3.31	3.31	2.29	−2.32	−2.63	−1.48	−3.11	−2.34	−1.26	−4.28	−4.83	−4.87
M4	15.55	16.68	14.95	7.44	11.47	15.52	6.75	9.65	13.37	3.89	6.62	13.26
M5	14.70	15.65	13.98	6.43	12.22	16.34	6.09	10.75	13.88	4.71	5.22	8.95
M6	13.94	14.90	13.30	6.95	9.73	13.12	7.41	9.09	12.42	7.11	6.95	10.01
M7	8.80	8.63	6.66	1.09	3.55	4.24	0.37	2.18	2.93	−0.88	0.73	1.07
M8	15.32	16.21	11.34	6.28	11.34	14.14	5.86	8.81	12.94	4.54	4.62	7.67
M9	9.29	9.26	7.66	9.14	4.64	6.21	8.22	4.10	4.59	6.87	2.03	2.31
M10	14.32	13.01	15.29	8.53	11.70	15.54	8.19	10.31	13.52	6.78	4.80	8.68
M11	9.55	9.43	8.31	1.40	3.62	4.40	0.41	2.44	4.10	−0.43	−0.79	−0.46
M12	20.17	20.04	16.74	8.58	10.39	14.56	7.73	12.37	13.79	6.73	4.41	8.59
M13	15.08	7.35	6.00	−1.28	0.17	0.71	−1.84	−0.24	0.67	−2.17	−2.36	−1.86
M14	28.59	14.24	12.04	5.33	2.19	2.75	4.33	1.42	1.80	3.51	0.54	1.06
M15	15.45	15.29	13.57	5.42	4.61	5.18	4.41	2.99	4.90	2.92	2.53	3.14
M16	25.15	26.07	23.54	12.67	11.25	15.13	11.99	8.85	15.05	10.27	7.92	12.71
M17	12.69	7.59	3.24	2.85	1.81	1.47	2.74	0.79	2.66	2.31	1.47	1.79
M18	23.98	12.19	5.26	5.88	5.59	5.86	5.35	4.33	7.05	4.97	4.90	5.72
M19	12.51	7.56	3.35	2.96	2.00	1.75	3.01	0.99	2.98	2.48	1.73	2.05
M20	23.61	12.35	5.28	5.97	5.93	6.32	5.50	4.72	7.50	5.16	5.26	6.38
Average	15.04	12.57	9.86	5.09	5.97	7.70	4.55	4.95	7.13	5.16	2.25	4.09

that for larger networks (M15–M16), ADP performs relatively strongly. For these instances, travel costs have a relatively high weight compared with visiting costs whereas restrictions on maximum route duration become more stringent. Finally, we note that the performance of ADP decreases when the basis functions do not properly address problem-specific characteristics (e.g., in the case of uneven demand distributions, see M7–M8). A similar effect is observed for more realistic instances (M17–M20), which are solved with a routing algorithm. Compared to the results of π^{min}, the outcomes suggest that lookahead has less added value for these instances. Interestingly, this effect is also observed for the rollout benchmarks; increasing the number of sample paths does not necessarily improve performance for these instances. As the outcomes of the routing algorithm are more difficult to learn than those of Daganzo's formula, additional basis functions might be necessary to improve the performance of π^{ADP} for these instances.

6.3. Experiments on Large Instances

This section reflects on the suitability of our solution method to handle large action spaces. First, we compare the results of the ADP policy (learned using the best set of basis functions) with those obtained with the myopic benchmark policies. Second, we provide statistics on the computational time required to solve the ILP.

For the large instances, we deal with state spaces of size $\gg 10^{178}$ and action spaces with sizes up to 2^{120}. Enumerating all actions within a reasonable time is not possible for these instances. In our experiments, we solve the decision problem at each decision moment with the ILP. In Section 4.3, we explained that the ILP requires defining a set of predefined input distances. The smaller this set is, the faster we can solve the ILP yet this comes at the cost of reduced accuracy. Based on preliminary testing on medium-sized instances, we define 30 distinct distance-to-depot sums to estimate the actual travel distance of a decision. On average, this setting yields an absolute cost difference between the ILP (using the approximate distances) and Daganzo's formula of less than 2%. The results of our experiments on L1 and L2 are shown in Table 6. As stated previously, we compare the performance of ADP with the myopic benchmark policies. The benchmark policies are clearly outperformed although the performance gaps are not as large as for the medium-sized experiments. A possible explanation for the smaller performance differences is that it might be more challenging to learn a good ADP policy because of the larger state space (more iterations may be required to properly learn values) and the introduction of the distance approximation error (as the approximation to some degree overestimates the direct costs). In addition to

Table 6. Simulation Performance of ADP Policies Against Benchmark Policies Relative to ADP with ϕ^{13}

Instance	π^{direct} (%)	π^{post} (%)	π^{min} (%)	Avg. solving time ILP (s)	Solved within 60 s (%)	Diff. with cost function (%)
L1	7.9	6.7	20.8	3.08	100.00	1.44
L2	6.9	6.0	18.9	5.96	99.31	1.81
Average	7.40	6.35	19.85	4.52	99.66	1.63

the performance gaps, we provide statistics on the time it takes to solve the ILP with CPLEX. 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 is shown to be fast enough to solve the decision problem in the online application.

7. 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 of 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 dispatch windows. We formulated this problem as a Markov decision model. Like many stochastic models, this model becomes computationally intractable for larger instances of the problem. To be able to solve larger instances, we provided an ADP algorithm based on value function approximation that overcomes the computational problems associated with large state spaces and outcome spaces. The consolidation policy is learned offline and might be periodically updated, for example, if the stochastic arrival process changes. The learned policy can be applied to online decision problems. To address computational problems with large action spaces, we formulated an ILP model to solve the decision problem within the ADP algorithm.

In our numerical experiments, we demonstrated a three-step methodology to find suitable VFAs for realistic-sized problems. First, we showed how to use small instances to evaluate basis functions, using the exact values obtained by dynamic programming as a benchmark. We tested various sets of basis functions. The set of basis functions yielding the best approximations contains the following explanatory variables: (i) the number of possible destinations to visit, (ii) the number of vehicles available per dispatch time, and (iii) the total volumes per latest delivery time. The resulting policy returns values that, on average, deviate 2.39% from the optimal values. Second, we evaluated the performance of the 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 with two policies based on heuristic rules, with a direct cost-minimization policy, and with a post-decision rollout policy. Using the best set of basis functions, we obtained an ADP policy that consistently outperformed the benchmark policies. The myopic benchmark policies are typically outperformed by 10%–15%, the rollout benchmark policies are outperformed by 2%–8%. Third, we solved two large instances—having state spaces of sizes $\gg 10^{178}$ and action spaces of sizes up to 2^{120}—using the ILP to solve the decision problems within ADP. The results illustrate the applicability of our algorithm for realistically sized instances. The resulting ADP policy outperformed the benchmark policies; for these instances, the myopic benchmark policies are outperformed by 6% to 20%. On average, we solved the ILP in 4.52 seconds in our large instances, which is fast enough for online decision making. However, because of 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 might be beneficial before implementing our approach in practice.

Acknowledgments

This study was partially funded by Dinalog, the Dutch Institute for Advanced Logistics. The authors are grateful for the reviews of two anonymous referees; their comments helped to substantially strengthen the structure and presentation of this paper. The authors thank Rune Larsen from the Technical University of Denmark for his help with the Copenhagen test instance.

References

Arslan A, Agatz N, Kroon LG, Zuidwijk RA (2016) Crowdsourced delivery–a pickup and delivery problem with ad-hoc drivers. Working paper, Erasmus University Rotterdam, Rotterdam, Netherlands.

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

Bohne S, Ruesch M, Barrera G (2015) Best practice handbook 2. Technical report, BESTFACT. https://www.researchgate.net/publication/283150066_BESTFACT_Best_Practice_Handbook_2.

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

Burger M, Repiský J (2012) Problems of linear least square regression. *Proc. ARSA-Adv. Res. Sci. Areas* 1(1):257–262.

Cai Q, He Q-M, Bookbinder JH (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. Geunes J, Akçali E, Pardalos PM, Romeijn HE, Shen Z-JM, eds. *Applications of Supply Chain Management and E-Commerce Research* (Springer Science and Business Media, New York), 3–51.

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

Chang HS, Hu J, Fu MC, Marcus SI (2013) *Simulation-Based Algorithms for Markov Decision Processes* (Springer, London).

Cherrett T, Allen J, McLeod F, Maynard S, Hickford A, Browne M (2012) Understanding urban freight activity–key issues for freight planning. *J. Transport Geography* 24:22–32.

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

Coelho LC, Laporte G, Cordeau J-F (2014) Heuristics for dynamic and stochastic inventory-routing. *Comput. Oper. Res.* 52:55–67.

Crainic TG, Ricciardi N, Storchi G (2004) Advanced freight transportation systems for congested urban areas. *Transportation Res. Part C: Emerging Tech.* 12(2):119–137.

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

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

Goodson JC, Ohlmann JW, Thomas BW (2013) Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Oper. Res.* 61(1):138–154.

Goodson JC, Thomas BW, Ohlmann JW (2017) A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *Eur. J. Oper. Res.* 258(1):216–229.

Higginson JK, Bookbinder JH (1995) Markovian decision processes in shipment consolidation. *Transportation Sci.* 29(3):242–255.

Klapp MA, Erera AL, Toriello A (2016) The one-dimensional dynamic dispatch waves problem. *Transportation Sci.*, ePub ahead of print May 20, http://dx.doi.org/10.1287/trsc.2016.0682.

Larsen A, Madsen OBG, Solomon M (2002) Partially dynamic vehicle routing—Models and algorithms. *J. Oper. Res. Soc.* 53(6):637–646.

Lium A-G, Crainic TG, Wallace SW (2009) A study of demand stochasticity in service network design. *Transportation Sci.* 43(2):144–157.

Luxen D, Vetter C (2011) Real-time routing with OpenStreetMap data. Agrawal D, Cruz I, Jensen CS, Ofek E, Tanin E, eds. *Proc. 19th ACM SIGSPATIAL Internat. Conf. Adv. Geographic Inform. Systems, GIS '*11 (ACM, New York), 513–516.

Meng Q, Wang T, Wang S (2012) Short-term liner ship fleet planning with container transshipment and uncertain container shipment demand. *Eur. J. Oper. Res.* 223(1):96–105.

Minkoff AS (1993) A Markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Oper. Res.* 41(1):77–90.

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

Nesterova N, Quak H (2015) Validating freight electric vehicles in urban Europe. Technical report, TNO, The Hague, Netherlands, http://frevue.eu/wp-content/uploads/2016/05/FREVUE-D1.3-State-of-the-Art-add1.pdf.

Nuzzolo A, Crisalli U, Comi A (2012) A delivery approach modeling for urban freight restocking. *J. Civil Engrg. Architecture* 6(3):251–267.

Pillac V, Gendreau M, Guéret C, Medaglia AL (2013) A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* 225(1):1–11.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Vol. 842 (John Wiley and Sons, Hoboken, NJ).

Powell WB, Simão HP, Bouzaiene-Ayari B (2012) Approximate dynamic programming in transportation and logistics: A unified framework. *Eur. J. Transportation Logist.* 1(3):237–284.

Ritzinger U, Puchinger J, Hartl RF (2016) A survey on dynamic and stochastic vehicle routing problems. *Internat. J. Production Res.* 54(1):215–231.

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

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

SteadieSeifi M, Dellaert NP, Nuijten WPM, Van Woensel T, Raoufi R (2014) Multimodal freight transportation planning: A literature review. *Eur. J. Oper. Res.* 233(1):1–15.

Sutton RS, Barto AG (1998) *Reinforcement Learning: An Introduction*, Vol. 1 (MIT Press, Cambridge, MA).

Topaloglu H, Powell WB (2006) Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS J. Comput.* 18(1):31–42.

Ulmer MW, Brinkmann J, Mattfeld DC (2015) Anticipatory planning for courier, express and parcel services. Dethloff J, Haasis H-D, Kopfer H, Kotzab H, Schönberger J, eds. *Logistics Management* (Springer International Publishing, Cham, Switzerland), 313–324.

Ulmer MW, Goodson JC, Mattfeld DC, Henning M (2017) Offline-online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Sci.* Forthcoming.

van Heeswijk WJA, Larsen R, Larsen A (2017) An urban consolidation center in the city of Copenhagen: A simulation study. Beta Working Paper Series, TU Eindhoven, Eindhoven, Netherlands.

van Heeswijk WJA, Mes MRK, Schutten JMJ (2015) An approximate dynamic programming approach to urban freight distribution with batch arrivals. Corman F, Voss F, Negenborn RR, eds. *Computational Logisitics*, Lecture Notes Comput. Sci., Vol. 9335 (Springer International Publishing, Cham, Switzerland), 61–75.

Voccia SA, Campbell AM, Thomas BW (2017) The same-day delivery problem for online purchases. *Transportation Sci.*, ePub ahead of print May 19, http://dx.doi.org/10.1287/trsc.2016.0732.