An approximate dynamic programming approach to urban freight distribution with batch arrivals

Wouter van Heeswijk, Martijn Mes, Marco Schutten

Beta Working Paper series 474

BETA publicatie	WP 474 (working paper)
ISBN	
ISSN	
NUR	804
Eindhoven	April 2015

An approximate dynamic programming approach to urban freight distribution with batch arrivals

Wouter van Heeswijk, Martijn Mes and Marco Schutten

University of Twente
Department of Industrial Engineering and Business Information Systems
P.O. Box 217, 7500 AE Enschede, The Netherlands
{w.j.a.vanheeswijk,m.r.k.mes,m.schutten}@utwente.nl

Abstract. We study a dispatch problem with uncontrolled batch arrivals of LTL orders at an urban consolidation center. These arrivals reflect the delivery of goods by independent carriers. The specific order properties (e.g., destination, size, delivery window) may be highly varying in city logistics, and directly distributing an incoming batch may yield high costs. Instead, the hub operator may decide to wait for incoming batches that allow for more efficient distribution. A waiting policy is required to decide which orders to ship and which orders to hold. We model the dispatching problem as a Markov decision problem. Dynamic Programming (DP) is applied to solve toy-sized instances. To solve realistic instances, we propose an Approximate Dynamic Programming (ADP) approach. Through numerical experiments, we show that the ADP approach closely approximates the optimal values of DP for small instances, and outperforms two benchmark policies for larger instances.

Keywords: urban distribution, transportation planning, consolidation, approximate dynamic programming

1 Introduction

In the field of urban freight logistics, the need for consolidation centers at the edge of urban areas is becoming increasingly important [10]. Due to the external costs of freight transport – such as congestion, air pollution, and noise hindrance – there is a desire for higher efficiency of goods transport within the city center. Governments actively seek to reduce the negative influence of large trucks in urban areas. Imminent regulations are, for example, restricted access areas and road pricing for heavy vehicles. Such developments spur the need for transshipments at the edge of urban areas. Transshipments allow both for bundling goods – such that more efficient routes can be generated – and dispatching environment-friendly vehicles such as electric vehicles on the last mile. However, the introduction of an additional transshipment in the supply chain also poses new challenges. We adopt the perspective of the party in charge of the consolidation center, to which we refer as the 'hub operator'. We focus on the timing of dispatching urban freighters in an environment where the batch arrivals of goods at the hub are not controlled by the hub operator.

In our problem settings, orders arrive at the consolidation center in batches (e.g., a trailer load), and subsequently need to be distributed to the customers in the city. A batch may contain orders with dispersed destinations, various delivery windows, distinct load sizes, etc. Directly distributing an arriving batch may therefore render poor solutions. Instead, waiting for other batches to arrive could yield order clusters for which better solutions are available. However, various uncertainties may play a role, such as when a new batch arrives and what the exact contents of such a batch are. Though faced with uncertainty, the operator may have a certain degree of knowledge regarding orders that may arrive in the future, such as forecasts or probability distributions. Based on the available knowledge regarding current orders and expectations on orders yet to arrive, the operator is able to make informed waiting decisions. This can be accomplished by deploying a waiting policy that makes dispatch decisions given the information and beliefs of the operator. Three basic types of waiting policies can be distinguished [2]. *Time-based* policies consider only the time until delivery as basis for dispatching decisions. *Quantity-based* policies base the decision whether or not to depart on the achieved fill rate of the vehicle (e.g., when it can make a tour utilizing at least 70% of its capacity). *Time-and-quantity* based approaches take both factors into account. Time-based policies generally result in better service to the customer in terms of lateness, while quantity-based policies perform better in terms of capacity utilization [3]. In practice, policies are generally time-and-quantity based [2]. Furthermore, a distinction can be made between myopic policies and policies that explicitly take into account information regarding future demand, the latter vastly increasing complexity [5, 8]. The use of information regarding future events in transportation is recognized as an important aspect of optimization, yet its incorporation in solution methods is still an ongoing development [9, 5, 12].

Our dispatch problem can be classified as a Delivery Dispatch Problem (DDP). The concept of queuing with bulk service – where customers arrive independently and are served in batches – was introduced by Bailey [1]. Neuts [7] expanded on the subject by explicitly studying control policies for the timing of vehicle dispatch. Traditional solutions to the DDP rely on analytical approaches such as queuing models. Route properties are considered to be a given input [6], such that routing decisions do not play a role. A more generic approach based on a batch Markovian arrival process is presented in [2], allowing to define arrival properties that do not necessarily follow well-defined distributions. Wang and Odoni [14] study a problem with batch arrivals of passengers at a train station, who are subsequently transported to their final destinations. They present a queuing approach aimed at the design of such transportation systems. To estimate the routing costs, they apply the approximation of Daganzo [4]. Using this approach, they find an average performance gap of 2% with the solutions of well-known routing heuristics. A key distinction between our DDP and the problem studied by Wang and Odoni is that they do not consider delivery windows;

their objective is to minimize service time for the passengers. Transportation requests are planned directly; the opportunity to defer dispatch decisions to future decision moments is not included in their approach. When considering more complex variants of the DDP, the stochastic and dynamic nature of the problem gives rise to Markov decision problems [6]. Although a Markov decision model is a useful framework to describe finite-horizon decision models with stochastic elements, practical implementations generally suffer from intractably large state spaces and expected values that cannot be calculated exactly [8]. The DDP is therefore too hard to solve exactly in most instances [6].

To model the stochastic and dynamic nature of our order arrival process, we formulate our DDP as a Markov decision process. Dynamic programming (DP) can be used to solve such models to optimality, but as stated in the previous paragraph, the sheer size of such problems prevents us from applying DP on larger instances. Mathematical programming has traditionally been applied to handle high-dimensional problems in transportation. However, this method generally does not cope well with stochastic information revealed over time [9]. Topaloglu and Powell [13] present a generic framework for solving dynamic resource-allocation problems. In this type of problems, the arrival of tasks is probabilistic, and coupling tasks to resources yields a reward. The methodology they apply is known as approximate dynamic programming (ADP). This simulation-based approach uses intelligent sampling to approximate the optimal solutions of DP [8]. ADP allows to retain the complexities embedded in the dispatch problem, while circumventing the computational effort required to solve the Markov model with stochastic dynamic programming. Various successful ADP applications can be found in transportation literature [12]. Provided that the value functions of the defined problem can be accurately approximated, ADP is a powerful approach to solve realistic transportation problems. Following frameworks such as [9, 13, 8], we develop an ADP approach to solve our dispatch problem. We aim to contribute to existing literature with (i) the formulation of a Markov model for DDPs with time windows, and (ii) an ADP approach to provide high-quality solutions for realistic-sized DDPs.

2 Problem Formulation

This section introduces the planning problem. We describe the problem in a generic way, making it applicable to a variety of instances. We assume that the characteristics of arriving orders are stochastic and have a known associated probability distribution. Certain variables that are treated as stochastic in our description may be known in practice. In that case, we can replace the stochastic variable by the actual information regarding future orders, creating a restricted instance of our problem. We consider a finite planning horizon, during which batches of orders can arrive at the consolidation center. Dispatching decisions are made at fixed decision moments within the planning horizon, with constant time intervals separating the decision moments. We model the arrival rates of order batches, the number of orders in a batch, order sizes, order destinations, and dispatch times as stochastic variables. When a batch of orders arrives at the center, the exact properties of the orders are revealed.

The decision problem that we address is the choice which orders to dispatch at the current decision moment. To make an informed decision, we require insight in the effects of postponing the dispatch of orders, e.g., distributing them from the center to their destination in the city. Postponed orders may be combined with future orders against lower costs than when dispatched at the current decision moment. We therefore consider optimization over a planning horizon; all orders arriving after the current decision moment are probabilistic. We assess all possible realizations of future order arrivals as scenarios, and plan the future arrivals as if they were actual orders. For both the actual and future orders belonging to a given scenario, we compute the costs of dispatching. The costs of dispatching probabilistic orders are required to quantify the expected costs. By assessing the realizations of probabilistic orders, we can compute the expected costs of the various dispatch decisions for the orders currently in inventory.

Consider an urban area with a fixed set of recipients (i.e., order destinations), which are delivered via a consolidation center at the edge of the area. Our representation of the urban distribution network is as follows. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$ be a directed and complete graph with $\mathcal{V}$ being the set of vertices and $\mathcal{A}$ being the set of arcs. The element $\{0\} \in \mathcal{V}$ represents the consolidation center in the network. The remaining vertices signify the subset of order destinations $\mathcal{V}' = \{1, 2, \ldots, |\mathcal{V}'|\}$, with $\mathcal{V}' = \mathcal{V} \setminus \{0\}$. A distance matrix gives the distance between any pair of vertices in the graph.

We consider a planning horizon that contains decision moments with fixed intermediate time intervals. For instance, we may have a one-day horizon with hourly decision moments. Let $\mathcal{T} = \{0, 1, \ldots, T\}$ be the set containing all decision moments, and $t \in \mathcal{T}$ describe any decision moment within the planning horizon. Furthermore, we consider a homogeneous fleet (i.e., a set of identical vehicles), though our method is able to cope with heterogeneous fleets as well. We distinguish between a primary fleet $\mathcal{Q}^{pr}$ and a secondary fleet $\mathcal{Q}^{se}$. We assume that the secondary fleet has an infinite size, and is either an actual transport alternative (e.g., renting an additional vehicle in case of shortage) or a dummy fleet with infinite costs. A dummy fleet serves as bound on capacity, without having to explicitly calculate the capacity constraints for each decision. We only assign vehicles in $\mathcal{Q}^{se}$ if no more vehicles in $\mathcal{Q}^{pr}$ are available. We assume that dispatching a secondary vehicle is always more expensive than dispatching a primary vehicle. To ease the presentation, we assume that every dispatched vehicle has a fixed route duration of $\tau_{route} \geq 1$ (this assumption can be easily relaxed). When dispatching at t, the vehicle will be available again at $t + \tau_{route}$. For decision-making purposes, we keep track of the availability of primary vehicles now and in the future. Let $r \in [0, \tau_{route} - 1]$ be the number of time intervals before a dispatched vehicle returns. Because all vehicle are available again at $t + \tau_{route}$, we only keep track of availability up to $t + \tau_{route} - 1$. Let $q_{t,r}$ denote the number of primary vehicles available for dispatch at $t + r$. It follows that

$q_{t,0}$ vehicles are available for dispatch at t. We record primary fleet availability in the vector $Q_t = (q_{t,0}, q_{t,1}, \dots, q_{t,t+\tau_{route}-1})$.

Every order is characterized by four properties: destination, load size, earliest dispatch time, and latest dispatch time. The order destination (i.e., the customer location) is represented by a vertex $v \in \mathcal{V}'$. Let $\mathcal{L} = \{\frac{1}{k}, \frac{2}{k}, \dots, 1\}$ be the discretized set of viable load sizes, with integer $k \geq 1$, and 1 representing a full load for an urban vehicle. The size of an order is given by $l \in \mathcal{L}$. The hard dispatch window of an order is given by earliest dispatch time $e \in \mathcal{E}$ and latest dispatch time $d \in \mathcal{D}$. Both indices are relative to the decision moment t; at the decision moment $t + 1$ all indices of orders in inventory are reduced by 1. Note that order types with $e > 0$ can be used to describe future orders. We define a maximum length of the dispatch window τ_{window}, such that $d \in [e, e + \tau_{window}]$. Every unique combination of the four properties represents an order type. Let $I_{t,v,l,e,d} \in \mathbb{Z}_+$ be the number of a given order type in inventory at t. We denote the information regarding all orders in inventory at t as $I_t = (I_{t,v,l,e,d})_{v\in\mathcal{V}',l\in\mathcal{L},e\in\mathcal{E},d\in\mathcal{D}}$. The state of the system at t, $S_t \in \mathcal{S}$, combines primary fleet availability with available orders, and is represented by

$$S_t = (Q_t, I_t)_{\forall v\in\mathcal{V}',l\in\mathcal{L},e\in\mathcal{E},d\in\mathcal{D}} \ , \forall t \in \mathcal{T} \ . \tag{1}$$

For $t \geq 1$, let $\mathcal{O}_t = \{0, 1, \dots, o_t^{max}\}$ be the set containing the number of possible order arrivals between decision moments $t - 1$ and t. Let $o_t \in \mathcal{O}_t$ be a realization of the number of orders arriving between $t-1$ and t. Furthermore, we set $l^{max} \in \mathbb{Z}_+$ as the maximum number of orders that can be held in inventory, i.e., the maximum inventory remaining after a decision.

For every decision moment t in the planning horizon, we decide which orders in inventory to dispatch. Orders that are not dispatched remain in inventory, and are available at the next decision moment. Let the integer variable $x_{t,v,l,e,d}$ describe the number of a specific order type to be dispatched at t. A feasible action at decision moment t is given by

$$x_t(S_t) = (x_{t,v,l,e,d})_{\forall v\in\mathcal{V}',l\in\mathcal{L},e\in\mathcal{E},d\in\mathcal{D}} \ , \tag{2}$$

where

$$\sum_{v\in\mathcal{V}',l\in\mathcal{L},e\in\mathcal{E},d\in\mathcal{D}} (I_{t,v,l,e,d} - x_{t,v,l,e,d}) \leq l^{max} \ , \tag{3}$$

$$x_{t,v,l,e,d} \leq I_{t,v,l,e,d} \ , \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D} \ , \tag{4}$$

$$x_{t,v,l,e,0} = I_{t,v,l,e,0} \ , \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E} \ , \tag{5}$$

$$x_{t,v,l,e,d} = 0 \ , e > 0 \ , \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall d \in \mathcal{D} \ , \tag{6}$$

$$x_{t,v,l,e,d} \in \mathbb{Z}_+ \ , \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D} \ . \tag{7}$$

Constraint (3) ensures that after dispatching, no more than the maximum inventory remains at the consolidation center. According to (4), it is not possible to dispatch more orders of a certain type than available at the decision moment t. Constraint (5) states that all orders that have a latest dispatch time equal to the maximum delay must be dispatched. (6) prevents orders with an earliest dispatch time in the future from being dispatched. (7) states that only nonnegative integer amounts of orders can be dispatched. The set of feasible actions in a given state is described by $\mathcal{X}_t(S_t)$.

3 Markov Model

We model the operator's decision problem as a Markov model. This model considers all possible realizations of orders arrivals during the planning horizon. With this knowledge, we can make the optimal dispatch decision for the current decision moment. In realistic instances, the state space, action space, and outcome space for such a model will be intractably large. Solving the Markov model in an exact manner is therefore not possible within a reasonable time. In Section 5, we solve some toy-sized instances of the Markov model using DP. The ADP approach as outlined in Section 4 is applied to larger instances.

Every action $x_t(S_t)$ has associated direct costs $C(S_t, x_t)$. The direct costs are the sum of fixed dispatching costs per vehicle, variable transportation costs, and handling costs. As the focus of this paper is on the dispatching decisions, we do not explicitly consider routing. Instead, we use the classic procedure of [4] to estimate the transportation costs for a dispatched set of orders. This procedure is known to provide good estimates of total route distances [11], given constraints on vehicle capacity, number of destinations, and shape of the service area. These constraints are likely to be fulfilled in an urban distribution setting. The approximation is based on the average distances between the depot and the customers, the number of customer locations visited, the size of the service area, and the capacity of the vehicles. We consider fixed handling costs per visited customer; note that this provides an incentive to simultaneously deliver multiple orders to a customer.

To model the uncertainties with respect to the properties of arriving orders, we introduce six stochastic variables. These are (i) the number of orders arriving O_t, (ii) the destination V, (iii) the order size L, (iv) the earliest dispatch time E, (V) the length of the dispatch window D_{window}, and (vi) the latest dispatch time $D = E + D_{window}$. The corresponding probability distributions are discrete and finite. To capture all probability distributions into a single variable, we define the exogenous information variable $\tilde{I}_{t,v,l,e,d} \in \mathbb{Z}_+, t \geq 1$, which indicates the number of arrivals of a specific order type. Furthermore, we introduce a generic variable W_t that describes all exogenous information, i.e., all orders arriving between $t-1$ and t:

$$W_t = [\tilde{I}_{t,v,l,e,d}]_{\forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D}} \ \ , t \geq 1 \ \ . \tag{8}$$

There exists a finite number of realizations of W_t. Let Ω_t be the set of possible batch arrivals between $t-1$ and t, and $\omega_t \in \Omega_t$ be a realization of the random variables occurring with $P(W_t = \omega_t)$.

We proceed to describe the transition from a state S_t to the next state S_{t+1}. The transition is affected by the action x_t and the new arrivals W_{t+1}. We first describe the effects of x_t. Orders not dispatched at t remain in inventory, hence must be included in S_{t+1}. As indices e and d are adjusted over time, we introduce two new variables to properly process the conversion. Let $e' = \max\{0, e-1\}$ and $d' = d-1$. Since $e < 0$ does not affect our decision making, capping e' at 0 reduces the number of possible order types. Let $\bar{q}_t \in \{0, \dots, |\mathcal{Q}^{pr}|\}$ be the number of primary vehicles dispatched at t; combined with Q_t this information suffices to compute Q_{t+1}. We represent new arrivals with the information variable W_{t+1}. This gives us the transition function

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

where

$$I_{t+1,v,l,e',d'} = I_{t,v,l,e,d} - x_{t,v,l,e,d} + \tilde{I}_{t+1,v,l,e',d'} \ , \tag{10}$$
$$\forall t \in \mathcal{T}, \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D} \ ,$$

$$q_{t+1,r} = \begin{cases} q_{t,r+1} - \bar{q}_t & \text{if } r < \tau_{route} - 1 \ , \\ |\mathcal{Q}^{pr}| & \text{if } r = \tau_{route} - 1 \ , \end{cases} \forall r \in [0, \tau_{route} - 1] \ . \tag{11}$$

Constraint (10) states that for every order type, we have the amount of the order type in state S_t, minus the amount of the order type that was dispatched, plus the amount of the order type that arrived between t and $t+1$. Constraint (11) ensures that Q_{t+1} is consistently updated. Having described the transition function, we now introduce the optimality equation that must be solved:

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

We proceed to describe the state space. Between every two consecutive decision moments $t-1$ and t, we can have $o_t \in \{0, \dots, |\mathcal{O}_t| - 1\}$ new orders arriving. Every arriving order can have any of the unique order types, given the constraints on the dispatch window. Before the new arrivals occur, we can have up to l^{max} orders in inventory. Hence, we can have at most $l^{max} + |\mathcal{O}_t| - 1$ orders at a given decision moment. A state can be any feasible combination of order types available at t, combined with any vector Q_t.

Next, we describe the action space. At every decision moment, we decide which orders to dispatch. Every combination of orders to dispatch represents a unique action. Orders that are not dispatched remain in inventory, and may be dispatched at the next decision moment. As we do not consider routing options, a unique selection of orders to dispatch equals exactly one action.

The transition from one state to another is determined by the current state, the used action, and the realization of the random variables. The remaining inventory before new orders arrive is deterministic. The probability of o_t orders arriving is given by $P(O_t = o_t)$. The probability of an arriving order being of a certain order type is given by the multivariate distribution $P(V, L, E, D)$. V, L and E are independent random variables, while D is the sum of the realizations of E and D_{window}.

The outcome space is dependent on the state S_t, the action x_t, and the realization of new arrivals ω_{t+1}. Orders not shipped at decision moment t are with certainty included in S_{t+1}. As route duration is deterministic, so is the change in fleet availability. Therefore, only the order arrivals account for stochasticity. To account for the multiple permutations corresponding to o_t order arrivals, we must multiply the probability of o_t orders arriving with a multinomial coefficient [15]. We obtain the following probability function for new arrivals:

$$P(W_t = \omega_t) = P(O_t = o_t) \frac{o_t!}{\prod\limits_{\tilde{I}_{t,v,l,e,d} \in \omega_t} \tilde{I}_{t,v,l,e,d}!} \prod_{v \in \mathcal{V}, l \in \mathcal{L}, e \in \mathcal{E}, d \in \mathcal{D}} P(V = v, L = l, E = e, D = d)^{\tilde{I}_{t,v,l,e,d}} \quad , t \geq 1 \ . \tag{13}$$

4 Solution approach

Realistic problems become extremely large in terms of state space, action space, and outcome space, making them intractable for the DP method. We address this problem with an approximate dynamic programming approach. With ADP, we retain the full level of detail in the state description, without having to enumerate the full state space. By means of Monte Carlo simulation, we are able to approximate the exact values of the DP method [8].

In our ADP implementation, we use the concept of the post-decision state [8]. The post-decision state S^x_t is the state immediately after action x_t, but before the arrival of new information ω_{t+1}. Given our action x_t, we have a deterministic transition from S_t to the so-called post-decision state S^x_t. We express this transition in the function

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

where

$$I_{t,v,l,e,d} = I_{t,v,l,e,d} - x_{t,v,l,e,d} \ , \qquad \forall t \in \mathcal{T}, \forall v \in \mathcal{V}', \forall l \in \mathcal{L}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D} \ , \tag{15}$$

$$q_{t,r} = q_{t,r} - \bar{q}_t \ , \forall r \in [0, \tau_{route} - 1] \ . \tag{16}$$

The post-decision state has a corresponding value function

$$V_t(S_t^x) = \mathbb{E}\{V_{t+1}(S_{t+1})|S_t^x\} \ . \tag{17}$$

Adopting the concept of the post-decision state allows us to represent our problem as a deterministic minimization problem. Although this reduces the computational effort, Equation (17) still requires to evaluate all states in the outcome space. In ADP, we therefore replace this value function with a single value function approximation $\bar{V}_t^{n-1}(S_t^x)$. n is an iteration counter, representing that we use an estimate from iteration $n-1$ at iteration n. At every decision moment, we take the best decision given our value function approximation. Incoming arrivals are generated according to Equation (13). Utilizing the post-decision state and value function approximation for future costs, we find the best action by solving the following deterministic minimization problem for the n-th iteration:

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

Equation (18) provides the action that minimizes the value $\hat{v}_t^n$, with the function to compute $\hat{v}_t^n$ being

$$\hat{v}_t^n = \min_{x_t \in \mathcal{X}_t(S_t)} (C_t(S_t, x_t) + \bar{V}_t^{n-1}(S_t^x)) \ . \tag{19}$$

Once we obtain our estimate $\hat{v}_t^n$, we can update $\bar{V}_{t-1}^{n-1}(S_{t-1}^x)$. For this, we use the following function:

$$\bar{V}_{t-1}^n(S_{t-1}^x) \leftarrow U^V(\bar{V}_{t-1}^{n-1}(S_{t-1}^x), S_{t-1}^x, \hat{v}_t^n) \ . \tag{20}$$

Table 1 provides an outline of our ADP algorithm.

We briefly discuss two options for the function U^V to update $\bar{V}_t^n(S_{t-1}^x)$: lookup and value function approximation (VFA). With the lookup approach, we store an estimate $\bar{V}_t^n(S_t^x)$ for every post-decision state, which is updated based on our observation at the next decision moment. We can speed up this procedure by first completing a full iteration, and then update all post-decision values at once (a procedure known as double pass, see [8]). Though the lookup ADP resolves several computational challenges of dynamic programming, we still need to visit a state to learn about its value. Hence, in order to properly learn the value of being in a specific state, we need to visit a large number of states. Instead, we want to learn about the value of many states with a single observation. To achieve this, we make use of VFA with the so-called basis function approach, see [8]. Let $\mathcal{F}$ be a set of features, with $f \in \mathcal{F}$ being some variable that partially explains the costs of being in a state. Relevant features for our dispatch problem are, e.g., the total volume of orders in inventory, the number of orders with

Table 1. ADP algorithm with post-decision states

```
Step 0 Initialize
    Step 0a:    Initialize $\bar{V}_t^0(S_t)$, $\forall t \in \mathcal{T}$, $\forall S_t \in \mathcal{S}$
    Step 0b:    Set iteration counter to $n = 1$, and set
                the maximum number of iterations to $N$.
    Step 0c:    Select an initial state $S_0$.
Step 1 For $t = 0$ to $T$ do:
    Step 1a:    Find the best action $\tilde{x}_t^n$ by solving Equation (18).
    Step 1b:    If $t > 0$, then update $\bar{V}_t^{n-1}(S_t)$ using Equation (20).
    Step 1c:    Obtain the post-decision state $S_t^x$ via Equation (14).
    Step 1d:    Obtain a sample realization $W_{t+1}$, calculate $S_{t+1}$
                with Equation (9)
Step 2 Set $n := n + 1$.
    If $n \leq N$, then go to Step 1.
Step 3 Return $\bar{V}_t^N(S_t^x), \forall t \in \mathcal{T}$.
```

$d = 0$, and the number of distinct destinations. Let $\phi_f(S_t^x)$ be a basis function of feature f – for example, a cross-product or a polynomial of f – that returns a certain value given S_t^x. Let θ_f^n be a weight corresponding to feature f. Our value function approximation becomes

$$\bar{V}_t^n(S_t^x) = \sum_{f \in \mathcal{F}} \theta_f^n \phi_f(S_t^x), \forall t \in \mathcal{T} \ . \tag{21}$$

Following [8], the weights θ_f^n are updated using recursive least squares for nonstationary data. Using this procedure, we are able to learn about the value of many states by sampling just a single state. Using VFA, it is therefore not necessary to visit all states in the state space to learn about their value, allowing to handle large state spaces. The key difficulty with VFA is to define basis functions that closely approximate the exact values of states. Good insight in the structure of the problem is required to select features that allow to accurately approximate the true values.

After learning the appropriate weights by completing the procedure in Table 1, VFA can be applied for practical decision making. By calculating the values for the post-decision states corresponding to our initial state, we are able to obtain the best action given the estimate. Only the features of the states, the basis functions, and the corresponding weights are necessary for decision making.

5 Numerical experiments

First, we solve a toy-sized instance with dynamic programming. We show how both the lookup approach and the VFA approach approximate the exact DP-values. Next, we consider larger problems. As an exact benchmark is missing for larger instances, we cannot show convergence results for these. Instead, we assess the *performance* of ADP. We take the results from the ADP procedure – i.e., the feature weights obtained from the VFA approach – to make decisions. We then compare ADP-based simulation results to the results of two benchmark policies. The first benchmark policy ('Postpone') we deploy in this paper is given in Table 2, and essentially aims to postpone as many orders as possible, until a suitable consolidation opportunity arises. The second benchmark policy ('DirectShipment') always ships orders, as long as primary vehicle capacity is available. 'DirectShipment' sorts and assigns orders just as 'Postpone' describes, and dispatches in the same way when secondary vehicles are required.

Table 2. Benchmark policy – Postpone

Step 0	Sort orders.	
	Step 0a:	Sort available orders based on lowest d.
	Step 0b:	Sort available orders with same d based on smallest size.
Step 1	While orders with $d = 0$ are unassigned do:	Assign order with $d = 0$ to vehicle.
Step 2	While remaining inventory exceeds l^{max} do:	Assign first order on list to vehicle.
Step 3	While capacity from already dispatched vehicles remains do:	Assign first order on list to vehicle.

We first describe the properties of our toy problem. We consider a fleet of two primary vehicles; secondary vehicles are twice as expensive as primary vehicles. We consider three distinct customer locations, a random order size from $\{0.2, 0.4, 0.6, 0.8, 1\}$, a maximum inventory of two orders, and a maximum of two arrivals per decision moment. We fix the tour length at $\tau_{route} = 1$. We set $e = 0$ for all orders, and randomly select d from $\{0, 1\}$. All probability distributions are uniform. We define a planning horizon with five decision moments. To provide some insight in the dimensionality of our dispatch problem: this toy-sized instance already has a state space of about 140,000 states.

The features we use for our VFA are (i) a constant, (ii) the number of vehicles available at the decision moment, (iii) the number of distinct order destinations, (iv) the total volume of orders in inventory, and (v) the square of the volume of orders in inventory. In Figure 1 and Figure 2, we show for two initial states (one without initial inventory, the other with four orders at the decision moment) how both the lookup approach and the VFA approach converge to the optimal values found with DP. In the first number of iterations, the estimates fluctuate due to the inability to accurately compute expected costs. However, by learning the values of visited states, ADP starts recognizing good actions.

From here on, we focus only on ADP with VFA using basis functions. The values we learn with VFA – by completing the algorithm in Table 1 – result in a set of weights for every decision moment. When in a given state, these weights allow us to estimate the values of all post-decision states that can be

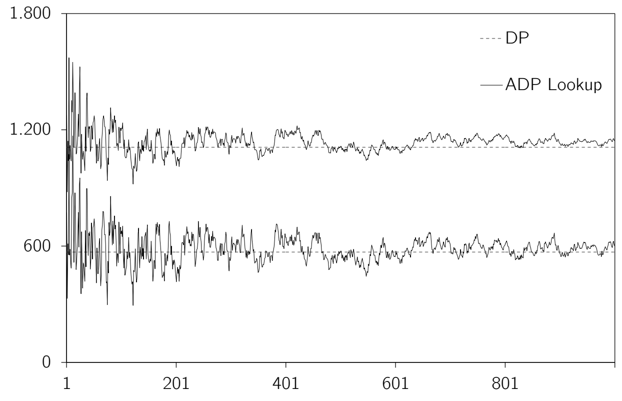

Fig. 1. Approximation of exact value with Lookup

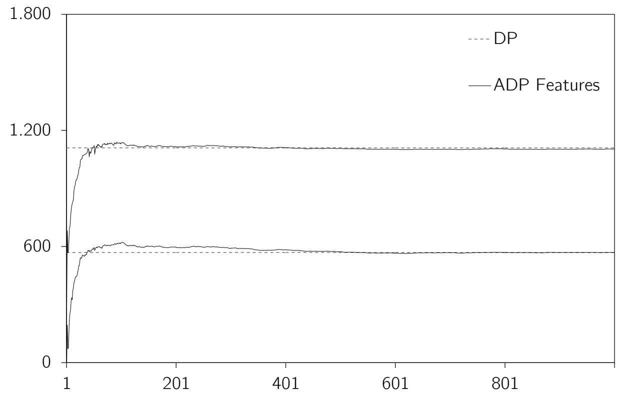

Fig. 2. Approximation of exact value with VFA

reached. Hence, ADP results in a policy, which we can use to solve a deterministic decision problem. We apply the learned policy in a Monte Carlo simulation on a variety of initial states, comparing the performance of the learned policy to both DP and the benchmark policies. For all simulations, we use the same realizations of order arrivals, and perform 10,000 simulation replications over the planning horizon. We do this for ten initial states, selected to represent a variety of properties. Table 3 shows the comparison between DP, ADP, and the two benchmark policies. The percentages indicate the average difference in costs between the optimal solution and the simulation results. By applying our ADP policy, we incur costs that are on average 0.60% higher than the optimal solution. ADP thereby outperforms the policy 'Postpone' by 3.14%, and the policy 'DirectShipment' by 11.56%. With ADP, we postpone 24% less orders than 'Postpone' does. For the initial states where 'DirectShipment' actually postpones orders – for some initial states it never will – ADP postpones 203% more.

Table 3. Comparison between ADP and benchmark policies for small instances

	Average costs	Average deviation from optimal	Lowest deviation from optimal	Highest deviation from optimal
DP	876	0.00%	0.00%	0.00%
ADP	881	0.60%	0.45%	0.99%
Postpone	908	3.76%	1.98%	9.41%
DirectShipment	1033	12.23%	8.52%	18.66%

Finally, we performs tests on five larger instances, with 10 customers, 2-5 vehicles, 10 order sizes, 5-10 arrivals per time unit, a maximum inventory of 5-10, a maximum dispatch window of 2, and a time horizon of 10. To illustrate how quickly the size of the state space grows, we give a small numerical example. With the mentioned settings, we have $10 \cdot 10 \cdot 3 = 300$ order types. Suppose we have 10 orders – all of different types – at a given decision moment. We then already have $\frac{300!}{290!10!}$ states only for this particular combination of order arrivals. This multinomial coefficient should be computed for all possible combinations of order arrivals; it follows that the size of the state space is $\gg 10^{18}$. Clearly, an exact benchmark for such instances cannot be provided.

Table 4 shows the results of our experiments on our larger instances. On average, ADP outperforms the policy 'Postpone' with 10.11% and the policy 'DirectShipment' with 19.71%. Particularly in cases with scarce vehicle capacity, ADP performs notably better. Under both benchmark policies, secondary vehicles are dispatched more frequently. Also, the benefits of ADP are greater when considering higher numbers of arrivals; the benchmark heuristics have no rules to cope with the larger complexity in dispatching decisions.

Table 4. Comparison between ADP and benchmark policies for larger instances

	Average costs	Average deviation from ADP	Lowest deviation from ADP	Highest deviation from ADP
ADP	3116	0.00%	0.00%	0.00%
Postpone	3443	10.11%	5.01%	12.93%
DirectShipment	3658	19.71%	10.33%	27.41%

6 Conclusions

In this paper, we proposed an ADP approach to make dispatch decisions at urban consolidation centers. We optimized decisions for a certain planning horizon, taking into account stochastic order arrivals during this horizon. We have shown that ADP is able to closely approximate the optimal values obtained by DP for toy-sized instances of our problem. For larger instances, ADP clearly and consistently outperforms both benchmark policies.

The ADP approach as described in this paper resolves the intractability of the state space and outcome space. However, we have not yet addressed the size of the action space, which can become problematically large as well. A possible approach to tackle this problem – without affecting the quality of decision-making – is to express the decision problem as an integer linear program that can be solved with limited computational effort. This requires the basis functions in VFA to be defined in such a way that they are linear with the decision problem.

Our numerical experiments have shown that even for small instances, simple waiting policies are inadequate to capture the complexity of waiting decisions. We expect that the ability of ADP to capture such complexities will be even more pertinent when increasing the size and complexity of instances. Further research will focus on the evaluation of realistically-sized instances and comparison with more sophisticated benchmark policies. The basis functions as proposed in this paper may not work well on every instance. Insights in appropriate VFAs for a variety of problem structures will yield a valuable contribution to existing literature. Both the ADP approach and its benchmarks need to be refined in order to provide an in-depth analysis of the applicability of ADP in realistic-sized dispatch problems.

Bibliography

[1] Bailey, N.T.: On queueing processes with bulk service. Journal of the Royal Statistical Society. Series B (Methodological) pp. 80–87 (1954)

[2] Cai, Q.: Shipment Consolidation in Discrete Time and Discrete Quantity: Matrix-Analytic Methods. Master's thesis, University of Waterloo (2011)

[3] Cetinkaya, S., Lee, C.Y.: Stock replenishment and shipment scheduling for vendor-managed inventory systems. Management Science 46(2), 217–232 (2000)

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

[5] Ichoua, S., Gendreau, M., Potvin, J.Y.: Exploiting knowledge about future demands for real-time vehicle dispatching. Transportation Science 40(2), 211–225 (2006)

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

[7] Neuts, M.F.: A general class of bulk queues with poisson input. The Annals of Mathematical Statistics pp. 759–770 (1967)

[8] Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, vol. 842. John Wiley & Sons (2011)

[9] Powell, W.B., Topaloglu, H.: Stochastic programming in transportation and logistics. Handbooks in operations research and management science 10, 555–635 (2003)

[10] Quak, H.: Sustainability of urban freight transport: Retail distribution and local regulations in cities. No. EPS-2008-124-LIS, Erasmus Research Institute of Management (ERIM) (2008)

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

[12] SteadieSeifi, M., Dellaert, N., Van Woensel, T.: A multimodal network flow problem with product quality preservation, transshipment, and asset management (2014)

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

[14] Wang, H., Odoni, A.: Approximating the performance of a last mile transportation system. Transportation Science (2014)

[15] Wells, M.B.: Elements of combinatorial computing. Elsevier (2014)