

# Approximate Dynamic Programming for a Class of Long-Horizon Maritime Inventory Routing Problems\*

Dimitri J. Papageorgiou, Myun-Seok Cheon

Corporate Strategic Research

ExxonMobil Research and Engineering Company

1545 Route 22 East, Annandale, NJ 08801 USA

{dimitri.j.papageorgiou,myun-seok.cheon}@exxonmobil.com

George Nemhauser, Joel Sokol

H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

765 Ferst Drive NW, Atlanta, Georgia, 30332

{gnemhaus,jsokol}@isye.gatech.edu

## Abstract

We study a deterministic maritime inventory routing problem with a long planning horizon. For instances with many ports and many vessels, mixed-integer linear programming (MIP) solvers often require hours to produce good solutions even when the planning horizon is 90 or 120 periods. Building on the recent successes of approximate dynamic programming (ADP) for road-based applications within the transportation community, we develop an ADP procedure to generate good solutions to these problems within minutes. Our algorithm operates by solving many small subproblems (one for each time period) and by collecting information about how to produce better solutions. Our main contribution to the ADP community is an algorithm that solves MIP subproblems and uses separable piecewise linear continuous, but not necessarily concave or convex, value function approximations and requires no off-line training. Our algorithm is one of the first of its kind for maritime transportation problems and represents a significant departure from the traditional methods used. In particular, whereas virtually all existing methods are “MIP-centric,” i.e., they rely heavily on a solver to tackle a nontrivial MIP to generate a good or improving solution in a couple of minutes, our framework puts the effort on finding suitable value function approximations and places much less responsibility on the solver. Computational results illustrate that with a relatively simple framework, our ADP approach is able to generate good solutions to instances with many ports and vessels much faster than a commercial solver emphasizing feasibility and a popular local search procedure.

**Keywords:** approximate dynamic programming, deterministic inventory routing, maritime transportation, mixed-integer linear programming, time decomposition.

---

\*Accepted for publication in Transportation Science

# 1 Introduction

We consider a deterministic maritime inventory routing problem (MIRP) in which a supplier is responsible for both the routing of vessels to distribute a single product and the inventory management at all ports in the supply chain network. The supplier controls a fleet of vessels for the entire planning horizon and knows inventory bounds as well as production and consumption rates at all locations. Compared to other inventory routing applications, the routing decisions in this problem are relatively simple as heterogeneous vessels make direct deliveries and fully load and fully discharge at a port. The main complexity stems from the long planning horizon considered and the wide range of travel times required to deliver product to customers.

Inventory routing problems (IRPs) arise naturally in a vendor managed inventory setting where a supplier (e.g., a vertically integrated company) manages the distribution and inventory levels of product at his customers [6]. Surveys of general IRPs for all modes of transportation are given in [1] and [9]. Idiosyncrasies and optimization models of IRPs specific to maritime settings are discussed in [7, 8, 25].

There are two motivating applications for our solution methodology. First, from a strategic planning perspective, the model considered in this paper can aid in the analysis of supply chain design decisions for applications involving valuable bulk goods such as liquefied natural gas (LNG). Because the model is needed for strategic purposes, long planning horizons must be considered and methods that attempt to solve a single MIP model often become hampered as the time dimension increases. A diverse set of users, including some without formal training in operations research or optimization, frequently wants to experiment with many scenarios to understand the impact of various design choices on the profitability of a supply chain. These decisions include: fleet size and mix, long-term contracts for vessels and with customers, investment in equipment, infrastructure and capacity limits, as well as other factors not present in the traditional IRP. Meanwhile, constructing a high-fidelity integrated model to address all of the issues faced by various business users within a stochastic programming or robust optimization framework is out of the question – agreement on model fidelity and the scenarios or uncertainty sets from users across different business units would be difficult to obtain. With this backdrop, the first motivation for our algorithm is to assist “experimentalist” business users interested in solving numerous instances in a small amount of time to analyze different business questions. In this setting, speed in generating good solutions trumps the importance of finding provably optimal solutions.

Second, from a tactical planning perspective, this model may be useful within a decomposition framework for a more detailed MIRP. For example, in [24], a MIRP with multiple ports per region and split pickups and split deliveries is considered. The problem is then solved using a decomposition approach in which the model is first aggregated by region, i.e., all ports within a region are thought of as one “super-port,” and solutions routing vessels between regions are generated. The model and algorithm presented here are well suited for this setting.

There are several reasons why we chose to explore an approximate dynamic programming (ADP) framework over the myriad algorithms available for deterministic IRPs. First, ADP has a proven track record of generating high-quality solutions to dynamic resource allocation problems, of which dynamic fleet management is a special case [38]. When modeled as a dynamic program, our problem shares many features of the dynamic fleet management problem (DFMP). In this context, our problem evolves as a sequence of dispatching problems where in each time period vessels in ports are available to be dispatched to another port. The key difference between our problem and the DFMP is the presence of an inventory management component. A second reason is the success ADP has enjoyed when solving large-scale dynamic problems

in road-based and rail-based applications found in industry [32, 5]. A third motive is that ADP has the ability to accommodate stochasticity without drastic changes to the framework or implementation. Despite the fact that we only consider deterministic problems, being able to adapt the framework developed here to stochastic variants of the underlying deterministic problem is of great interest.

The primary contributions of this paper are: (1) The development of an ADP algorithm for a class of long-horizon maritime inventory routing planning problems that outperforms well-known approaches including a popular MIP-based local search heuristic and a commercial MIP solver emphasizing feasibility. (2) An ADP approach that uses separable piecewise linear continuous, *but not necessarily concave*, value function approximations and requires no off-line training. (3) Further evidence that the added complexity of solving MIP subproblems has the potential to yield good solutions.

This paper is organized as follows. Section 2 contains a literature review of research germane to maritime transportation and ADP. In Section 3, we present a detailed description of our problem along with a mixed-integer linear programming formulation and a dynamic programming formulation for it. In Section 4, we provide our solution methodology using an ADP framework. Finally, computational results in Section 5 illustrate the effectiveness of our ADP approach.

## 2 Literature Review

### 2.1 Maritime Applications

From an application perspective, this paper focuses on long-horizon maritime inventory routing problems such as those arising in the LNG industry, which are known as LNG-IRPs. A MIRP can be defined as “a planning problem where an actor has the responsibility for both the inventory management at one or both ends of the maritime transportation legs, and for the ships’ routing and scheduling” [8]. Using this definition, previous approaches applied to LNG-IRPs can be divided into two groups based on whether the actor has control of both the production and consumption ports, or just one of the two. Rakke et al. [28, 27], Stålhane et al. [34], and Halvorsen-Weare and Fagerholt [20] treat the case where the actor only has control of production by attempting to generate annual delivery plans for the world’s largest LNG producer. The producer has to fulfill a set of long-term customer contracts. Each contract either outlines monthly demands, or states that a certain amount of LNG is to be delivered fairly evenly spread throughout the year to a given consumption port. Over- and under-deliveries are accepted, but incur a penalty. In contrast, there are also LNG-IRPs that arise for vertically integrated companies who have control of both the production and consumption side of the supply chain [18, 19, 11, 16, 17, 31]. In some applications, the opportunity to sell LNG in the spot market using short-term contracts is also present.

Several solution methods for the case when the actor only has control of production have been investigated. Rakke et al. [28] propose a rolling horizon heuristic in which a sequence of overlapping MIP subproblems are solved. Each subproblem involves at most 3 months of data and consists of a one-month “central period” and a “forecasting period” of at most two months. Once a best solution is found (either by optimality or within a time limit), all decision variables in the central period are fixed at their respective values and the process “rolls forward” to the next subproblem. Stålhane et al. [34] propose a construction and improvement heuristic that creates scheduled voyages based on the availability of vessels and product while keeping inventory feasible. Halvorsen-Weare and Fagerholt [20] study a simplified version of the LNG-IRP problem where cargoes for each long-term contract are pre-generated with defined time windows, and the fleet of ships can be divided

into disjoint groups. The problem is decomposed into a routing subproblem and a scheduling master problem where berth, inventory and scheduling decisions are handled in the master problem, while routing decisions are dealt with in the subproblem. Unlike branch-and-price, the subproblems are solved only once. Most recently, Rakke et al. [27] developed a branch-price-and-cut approach that relies on delivery patterns at the customers.

Solution techniques for MIRPs faced by a vertically integrated company are presented in [25], where MIRPs with inventory tracking at every port are surveyed. Grønhaug et al. [19] introduce a branch-and-price method in which the master problem handles the inventory management and the port capacity constraints, while the subproblems generate the ship route columns. Fodstad et al. [11] solve a MIP directly while Uggen et al. [40] present a fix-and-relax heuristic. Goel et al. [16] present a simple construction heuristic and adapt the local search procedure of Song and Furman [33] to generate solutions to instances with 365 time periods. Their model seeks to minimize penalties and does not consider travel costs.

## 2.2 Approximate Dynamic Programming

Over the past few decades, approximate dynamic programming has emerged as a powerful tool for certain classes of multistage stochastic dynamic problems. The monographs by Bertsekas and Tsitsiklis [2], Sutton and Barto [35], and Powell [26] provide an introduction and solid foundation to this field. In the last decade, Powell [26] and his associates have successfully applied ADP to large-scale applications arising in transportation and logistics.

Our work builds on the ideas presented by Powell and his associates in the context of stochastic dynamic resource allocation problems. Dynamic fleet management problems are a special case in this problem class. When modeled as MIPs, these problems take place on a time-space network involving location-time pairs. Service requests (demands for service) from location  $i$  to location  $j$  appear over time (randomly, in the stochastic setting) and profit is earned by assigning vehicles of different types to fulfill these service requests. Myopically choosing the vehicle type that maximizes the immediate profit is often not best over a longer horizon. Empty repositioning is also a key issue.

Our point of departure is the class of the dynamic fleet management problems (DFMPs) studied in [14, 15, 38, 36, 37]. In Godfrey and Powell [14], a stochastic DFMP is studied in which requests for vehicles to move items from one location to another occur randomly over time and expire after a certain number of periods. Once a vehicle arrives at its destination node (location-time pair), it is available for servicing another request or for traveling empty to a new location. A single vehicle type with single-period travel times is considered and an ADP algorithm in which a separable piecewise linear concave value function approximation is shown to yield strong performance. This work is extended in [15] to handle multiperiod travel times between locations. Further extensions are made to allow for deterministic multiperiod travel times with multiple vehicle types [38], random travel times with a single vehicle type [36], and random travel times with multiple types [37]. In all of these studies, separable piecewise linear concave value function approximations are used and shown to work well.

There are two important observations to make regarding the above papers. First, they all treat dynamic fleet management problems, not inventory routing problems. In the dynamic fleet management setting, tasks occur over time and must be completed before some expiration period. In the IRP setting, the notion of a task does not exist since product is being continuously produced and consumed. Consequently, for the DFMP, the movement of vehicles is critical, while the amount of a product on the vehicles or at each location

is not an issue and, therefore, is not modeled. Second, they all use value function approximations that are only a function of the *vehicle state*. That is, they value the number of each vehicle type that will be available at each location over future time periods.

In contrast, Toriello et al. [39] use value function approximations that are a function of the *inventory state* at each location in order to address a deterministic IRP with a planning horizon of 60 periods. Their problem involves a fleet of homogeneous vehicles that transport a single product between a single loading region and a single discharging region. Each region may have multiple ports. They assume that (1) the inter-regional travel time is a constant regardless of which location is last visited in the loading region and which location is the first visited in the discharging region, and that (2) all locations visited in a region by the same vehicle are visited in the same time period. With these assumptions, the problem reduces to an inventory routing problem with single-period travel times. In addition, after traveling from the loading region to the discharging region, vehicles exit the system as they are assumed to behave like voyage-chartered vessels as in [12, 33, 10, 21]. They solve a nontrivial MIP in each subproblem. They employ separable piecewise linear concave value functions of the inventory to generate high-quality solutions much faster than solving a large MIP model with a commercial solver. However, their value function approximation requires hours of off-line training to construct; all of our computations are performed on-line.

Bouzaiene-Ayari et al. [5] tackle locomotive planning problems arising at Norfolk Southern. One of their models involves much more detail than the one presented here and ADP is successfully employed to generate high-quality solutions. Like Toriello et al. [39], they also solve MIP subproblems. They updated the slopes of their value function approximations using the duals of the LP relaxation.

In this paper, we extend the ideas above for the DFMP by considering a deterministic IRP with a single loading region, multiple discharging regions, and multiperiod travel times. One-way travel times range between 5 and 37 periods. Like Toriello et al. [39], we employ value function approximations that are only a function of the inventory state. However, the presence of multiple discharging regions, multiperiod travel times, and longer time horizons makes our problem arguably more complex.

### 3 Problem Description and Formulations

In this section, we present a mixed-integer linear programming formulation as well as a dynamic programming formulation of the problem. We begin with a description of the problem and introduce relevant notation.

Let  $\mathcal{T}$  be the set of time periods and let the full horizon be of length  $T = |\mathcal{T}|$ . Let  $\mathcal{J}^P$  and  $\mathcal{J}^C$  denote the set of production (or loading) and consumption (or discharging) ports, respectively, and let  $\mathcal{J} = \mathcal{J}^P \cup \mathcal{J}^C$  be the set of all ports. Let the parameter  $\Delta_j$  be 1 if  $j \in \mathcal{J}^P$  and -1 if  $j \in \mathcal{J}^C$ . Each port  $j$  has a berth limit  $B_j$ . We assume that there is exactly one port within each region (consequently, we will use the terms “port” and “region” interchangeably). Note that each port may have only one classification: loading or discharging. Let  $d_{j,t}$  denote the amount of product produced or consumed at port  $j$  in time period  $t$ . Each port has an inventory capacity of  $S_{j,t}^{\max}$ . The amount of inventory at the end of time period  $t$  must be between 0 and  $S_{j,t}^{\max}$ .

Since it may not be possible to satisfy all demand or avoid hitting tank top, we include a simplified spot market so that a consumption port may buy product and a production port can sell excess inventory whenever necessary. The penalty parameter  $P_{j,t}$  denotes the unit cost associated with the spot market at port  $j$  in time period  $t$ . We assume that  $P_{j,t} > P_{j,t+1}$  for all  $t \in \mathcal{T}$  so that the spot market is only used as late as possible, i.e., to ensure that a solution will not involve lost production (stockout) until the inventory level

reaches capacity (falls to zero). When the penalty parameters are large (like a traditional “Big  $M$ ” value), inventory bounds can be considered “hard” constraints. When they are small, however, inventory bounds can be treated as “soft” constraints. This “soft” interpretation may be beneficial in strategic planning problems for several reasons. First, a user may attempt to solve an instance with a demand forecast that cannot be met by the existing fleet in order to understand the limitations of the current infrastructure (see also [16]). Second, the inventory bounds given as input may be overly conservative in order to make the solution more robust when, in fact, slight bound violations may be acceptable. Third, incurring a small penalty for a particular solution (as opposed to declaring it strictly infeasible) can mitigate minor unwanted effects of using a discrete-time model [25].

Vessels travel from port to port, loading and discharging product. We assume vessels fully load and fully discharge at a port and that direct deliveries are made. Each vessel belongs to a vessel class  $vc \in \mathcal{VC}$ . Vessel class  $vc$  has capacity  $Q^{vc}$ . Vessels are owned by the supplier or time-chartered for the entire planning horizon. We assume that port capacity always exceeds vessel capacity, i.e.,  $S_{j,t}^{\max} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$  and that vessels can fully load or discharge in a single period. These assumptions allow vessels to load or discharge in the same period in which they leave a port so that loading and discharging decisions do not need to be explicitly modeled.

In both formulations below, it is convenient to model the problem on a time-expanded network. The network has a set  $\mathcal{N}_{0,T+1}$  of nodes and a set  $\mathcal{A}$  of directed arcs. The node set is shared by all vessel classes, while each vessel class has its own arc set  $\mathcal{A}^{vc}$ . The set  $\mathcal{N}_{0,T+1}$  of nodes consists of a set  $\mathcal{N} = \{(j, t) : j \in \mathcal{J}, t \in \mathcal{T}\}$  of “regular” nodes, or port-time pairs, as well as a source node  $n_0$  and a sink node  $n_{T+1}$ .

Associated with each vessel class  $vc$  is a set  $\mathcal{A}^{vc}$  of arcs, which can be partitioned into source, sink, waiting, and travel arcs. A source arc  $a = (n_0, (j, t))$  from the source node to a regular node represents the arrival of a vessel to its initial destination. A sink arc  $a = ((j, t), n_{T+1})$  from a regular node to the sink node conveys that a vessel is no longer being used and has exited the system. A waiting arc  $a = ((j, t), (j, t + 1))$  from a port  $j$  in time period  $t$  to the same port in time period  $t + 1$  represents that a vessel stays at the same port in two consecutive time periods. Finally, a travel arc  $a = ((j_1, t_1), (j_2, t_2))$  with  $j_1 \neq j_2$  represents travel between two distinct ports, where the travel time  $(t_2 - t_1)$  between ports is given. If a travel or sink arc is taken, we assume that a vessel fully loads or discharges immediately before traveling. The cost of traveling on arc  $a \in \mathcal{A}^{vc}$  is  $C_a^{vc}$ .

The set of all travel and sink arcs for each vessel class are denoted by  $\mathcal{A}^{vc, \text{inter}}$  (where “inter” stands for “inter-regional”). The set of incoming and outgoing arcs associated with vessel  $vc \in \mathcal{VC}$  at node  $n \in \mathcal{N}_{0,T+1}$  are denoted by  $\mathcal{RS}_n^{vc}$  (for reverse star) and  $\mathcal{FS}_n^{vc}$  (for forward star), respectively. Similarly,  $\mathcal{FS}_n^{vc, \text{inter}}$  denotes the set of all outgoing travel and sink arcs at node  $n$  for vessel class  $vc$ . For our strategic planning problem, modeling the flow of vessel classes avoids the additional level of detail associated with modeling each individual vessel. Moreover, we found that modeling vessel classes could remove symmetry and improve solution times by more than an order of magnitude on large instances.

**Assumptions:** For ease of reference, we collect the assumptions made throughout this paper: (1) There is exactly one port within each region; (2) Port capacity always exceeds the capacity of the vessels, e.g.,  $S_{j,t}^{\max} \geq \max\{Q^{vc} : vc \in \mathcal{VC}\}$ ; (3) Travel times are deterministic; (4) Vessels can fully load or discharge in a single period (in other words, the time to load/discharge is deterministic and built into the travel time); (5) Production and consumption rates are known; (6) There is a single loading port as is typically the case for LNG-IRPs problems [27, 20, 34, 16]; (7) In a single time period, at most one vessel per vessel class may begin an outgoing voyage to a given discharging region or a return voyage to the loading region.

### 3.1 A Discrete-Time Arc-Flow Mixed-Integer Linear Programming Model

Before describing a MIP formulation of the problem, we need to define the decision variables. Let  $x_a^{vc}$  be the number of vessels in vessel class  $vc$  that travel on arc  $a \in \mathcal{A}^{vc}$ . Let  $s_{j,t}$  be the ending inventory at port  $j$  in time period  $t$ . Initial inventory  $s_{j,0}$  is given as data. Finally, let  $\alpha_{j,t}$  be the amount of inventory bought from or sold to the spot market near port  $j$  in time period  $t$ .

We consider the following discrete-time arc-flow MIP model:

#### MIP Model

$$\max \sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{A}^{vc}} -C_a^{vc} x_a^{vc} + \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} -P_{j,t} \alpha_{j,t} \quad (1a)$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{FS}_n^{vc}} x_a^{vc} - \sum_{a \in \mathcal{RS}_n^{vc}} x_a^{vc} = \begin{cases} +1 & \text{if } n = n_0 \\ -1 & \text{if } n = n_{T+1} \\ 0 & \text{if } n \in \mathcal{N} \end{cases}, \quad \forall n \in \mathcal{N}_{0,T+1}, \forall vc \in \mathcal{VC} \quad (1b)$$

$$s_{j,t} = s_{j,t-1} + \Delta_j \left( d_{j,t} - \sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,inter}} Q^{vc} x_a^{vc} - \alpha_{j,t} \right), \quad \forall n = (j,t) \in \mathcal{N} \quad (1c)$$

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc,inter}} x_a^{vc} \leq B_j, \quad \forall n = (j,t) \in \mathcal{N} \quad (1d)$$

$$\alpha_{j,t} \geq 0, \quad \forall n = (j,t) \in \mathcal{N} \quad (1e)$$

$$s_{j,t} \in [0, S_{j,t}^{\max}], \quad \forall n = (j,t) \in \mathcal{N} \quad (1f)$$

$$x_a^{vc} \in \{0, 1\}, \quad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc,inter} \quad (1g)$$

$$x_a^{vc} \in \mathbb{Z}_+, \quad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc} \setminus \mathcal{A}^{vc,inter}. \quad (1h)$$

The objective is to minimize the sum of all transportation costs and penalties for lost production and stockout. (We write the model as a maximization problem to coincide with the framework used in our dynamic programming formulation below, where it is typical to maximize a value function.) Constraints (1b) require flow balance of vessels within each vessel class. Constraints (1c) are inventory balance constraints at loading and discharging ports, respectively. Berth limit constraints (1d) restrict the number of vessels that can attempt to load/discharge at a port at a given time. This formulation requires that a vessel must travel at capacity from a loading region to a discharging region and empty from a discharging region to a loading region. This model does not require decision variables for tracking inventory on vessels (vessel classes), nor does it include decision variables for the quantity loaded/discharged in a given period.

This model is similar to the one studied in Goel et al. [16]. The major differences are that they do not include travel costs in the objective function; they model each vessel individually (in other words, there is only one vessel per vessel class); they model consumption rates as decision variables with upper and lower bounds; and they include an additional set of continuous decision variables to account for cumulative unmet demand at each consumption port.

### 3.2 Dynamic Programming Formulation

We now formulate our MIRP as a finite-horizon dynamic programming problem. It is convenient to interpret this DP representation as a sequence of dispatching problems. At each point in time, a regional manager has a set of vessels available for dispatching in his region. If enough inventory is available for a vessel to

fully load or enough excess capacity is available for a vessel to fully discharge, then the manager faces three options for each available vessel: send the vessel to another region, have the vessel remain in the region, or force the vessel to exit the system.

With this interpretation in mind, we now describe the DP formulation. The state of the system at time  $t$  is given by the vector tuple  $(\mathbf{r}_t, \mathbf{s}_t)$  where

$$\begin{aligned} \mathbf{r}_t &= [r_{j,u,t}^{vc}]_{j \in \mathcal{J}, u=t, \dots, T, vc \in \mathcal{VC}}, \text{ a vector of current and future vessel positions} \\ \mathbf{s}_t &= [s_{j,u,t}]_{j \in \mathcal{J}, u=t, \dots, T}, \text{ a vector of current and future inventory levels} \\ r_{j,u,t}^{vc} &= \text{Just before making decisions in time period } t \text{ (i.e., in the time } t \\ &\quad \text{subproblem), the number of vessels in vessel class } vc \text{ that are or will be} \\ &\quad \text{available for service at location } j \text{ in the beginning of time period } u \\ &\quad \text{when decisions are made in time period } u \text{ (} u \geq t \text{)} \\ s_{j,u,t} &= \text{The number of units of inventory "available" at location } j \text{ at the end of} \\ &\quad \text{time period } u, \text{ after making and executing all decisions in the time } t \text{ subproblem.} \end{aligned}$$

Here, "available" inventory refers to inventory that is either in storage at the port (i.e., has already been discharged) or is on vessels that are at the port but have yet to discharge. The initial state of the system, i.e., inventories and vessel positions, is given. Let  $s_{j,t-1,t}$  denote the initial inventory available at port  $j$  in the beginning of time period  $t$  prior to any events (e.g., decisions, deliveries, consumptions, etc.) taking place.

Given a time period  $t$  and the state of the system, we have restrictions on the number and weighted combination of vessels that may leave a port in a given time period:

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc, \text{inter}}} x_a^{vc} \leq B_j, \quad \forall n = (j, t) \in \mathcal{N} \quad (2a)$$

$$\sum_{vc \in \mathcal{VC}} \sum_{a \in \mathcal{FS}_n^{vc, \text{inter}}} Q^{vc} x_a^{vc} \leq \begin{cases} s_{j,t-1,t} + d_{j,t} & \text{if } j \in \mathcal{J}^P \\ S_{j,t}^{\max} - s_{j,t-1,t} - d_{j,t} & \text{if } j \in \mathcal{J}^C \end{cases}, \quad \forall n = (j, t) \in \mathcal{N}. \quad (2b)$$

Constraints (2a) are berth limit restrictions (identical to Constraints (1d)) and limit the number of vessels that may take an inter-regional or sink arc in time period  $t$ . Constraints (2b) ensure that the maximum amount of inventory that can be loaded (discharged) onto all vessels leaving a port does not exceed the amount of available inventory (remaining capacity) at that port.

Next, we have to model the dynamics of the system, i.e., the transition of vessels and inventory over time. To model the flow of vessels, we have the following requirements:

$$\sum_{a \in \mathcal{FS}_n^{vc}} x_a^{vc} = r_{j,t,t}^{vc}, \quad \forall n = (j, t) \in \mathcal{N}, \forall vc \in \mathcal{VC} \quad (3a)$$

$$r_{j,u,t+1}^{vc} - \sum_{a=((t,t),n) \in \mathcal{RS}_n^{vc}} x_a^{vc} = r_{j,u,t}^{vc}, \quad \forall n = (j, u) \in \mathcal{N} : u > t, \forall vc \in \mathcal{VC}. \quad (3b)$$

Equations (3a) state that all vessels available at time  $t$  must transition by remaining at the same port, moving to another port, or exiting the system. Equations (3b) keep track of the number of vessels in each vessel class that will become available in some future time period  $u > t$ . Inventory at ports is updated according to the equations

$$s_{j,u,t} = \begin{cases} s_{j,u-1,t} + d_{j,u} - \alpha_{j,u} - q_{j,u}^{\text{out}} & \text{if } j \in \mathcal{J}^P \\ s_{j,u-1,t} - d_{j,u} + \alpha_{j,u} + q_{j,u}^{\text{in}} & \text{if } j \in \mathcal{J}^C \end{cases}, \quad \forall n = (j, t) \in \mathcal{N}, \forall u \geq t \quad (4)$$



where  $q_{j,u}^{\text{in}}$  and  $q_{j,u}^{\text{out}}$  represent the quantity of inventory incoming to and outgoing from port  $j$  at time  $u$  after decisions in time  $t$  have been made. Specifically, define  $q_{j,u}^{\text{out}} = \sum_{vc \in \mathcal{VC}} Q^{vc} \left( \sum_{a \in \mathcal{FS}_{(j,u)}^{vc, \text{inter}}} x_a^{vc} \right)$  if  $u = t$  and 0 if  $u > t$ , and  $q_{j,u}^{\text{in}} = \sum_{vc \in \mathcal{VC}} Q^{vc} \left( \sum_{a \in \mathcal{XS}} x_a^{vc} + r_{j,u,t}^{vc} \right)$  with  $\mathcal{XS} = \mathcal{FS}_{(j,u)}^{vc, \text{inter}}$  if  $u = t$  and  $\mathcal{XS} = \mathcal{RS}_{(j,u)}^{vc}$  if  $u > t$ . Lastly, before transitioning from the time  $t$  subproblem to the time  $t + 1$  subproblem, we must initialize  $s_{j,t,t+1} = s_{j,t,t}$  for all  $j \in \mathcal{J}$ .

Using the principle of optimality, we can write our time  $t$  optimization problem as

$$V_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \max \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}_{(j,t)}^{vc}} -C_a^{vc} x_a^{vc} - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + V_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t) \quad (5a)$$

$$\text{s.t. } (2), (3), (4) \quad (5b)$$

$$\alpha_{j,u} \geq 0, \quad \forall n = (j, u) \in \mathcal{N} : u \geq t \quad (5c)$$

$$s_{j,u,t} \geq 0, \quad \forall n = (j, u) \in \mathcal{N} : u \geq t \quad (5d)$$

$$x_a^{vc} \in \{0, 1\}, \quad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc, \text{inter}} : a = ((\cdot, t), (\cdot, \cdot)) \quad (5e)$$

$$x_a^{vc} \in \mathbb{Z}_+, \quad \forall vc \in \mathcal{VC}, \forall a \in \mathcal{A}^{vc} \setminus \mathcal{A}^{vc, \text{inter}} : a = ((\cdot, t), (\cdot, \cdot)). \quad (5f)$$

Note that  $V_t$  is a function of  $\mathbf{r}_t$  and  $\mathbf{s}_{t-1}$ , not  $\mathbf{s}_t$ . This is because we have followed the standard notation in inventory models where a variable  $s_t$  denotes the *ending* inventory in time period  $t$ . Also note that we only require the inventory variables  $s_{j,u,t}$  to be nonnegative and not below port capacity. This is because, according to our definition,  $s_{j,u,t}$  represents the amount of inventory in storage or on a vessel at port  $j$  in some future time period  $u$ , and therefore could easily exceed capacity at a port.

## 4 Solution Methodology

Solving dynamic programming problems is notoriously challenging due to the curse of dimensionality: As the dimension of the state space grows, the time required to solve the problem exactly grows exponentially. The MIRP studied here is no exception. Attempting to solve Bellman's equation (5) exactly is futile. Instead, we try to solve it approximately using ADP methods.

We accomplish this by replacing the future value function  $V_{t+1}$  with a suitable approximation  $\hat{V}_{t+1}$  and solving the approximate problem

$$\tilde{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \max \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}_{(j,t)}^{vc}} -C_a^{vc} x_a^{vc} - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + \hat{V}_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t) \quad (6a)$$

$$\text{s.t. } (5b) - (5f). \quad (6b)$$

Now we describe our algorithm. Pseudocode of our approach is shown in Algorithm 1. The most common ADP methods step forward in time. The decisions made in the time  $t$  subproblem are guided by the current value function approximation, as shown in Step 5. After a solution to the time  $t$  subproblem is obtained, we typically collect information to determine what the marginal benefit would be from having an additional vessel or an additional unit of inventory available at a given port and future time. Next, we update the state of the system. Once all subproblems have been solved, a solution to the full planning problem exists and we update the value function approximations using information obtained from the current solution and from each of the subproblems.

---

**Algorithm 1** Basic Deterministic ADP Algorithm

---

1: Initialization: Choose an approximation  $\hat{V}_t$  for all  $t \in \mathcal{T}$ .

2: **for**  $n = 1$  to  $N$  **do**

3:   Initialize the state of the system  $(\mathbf{r}_1, \mathbf{s}_0)$ .

4:   **for**  $t = 1$  to  $T$  **do**

5:     Solve the time  $t$  subproblem

$$\max_{\mathbf{x}, \alpha} \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}_{(j,t)}^{vc}} -C_a^{vc} x_a^{vc} - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + \hat{V}_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t).$$

6:     Obtain marginal value information.

7:     Update the state of the system using Equations (3) and (4).

8:   **end for**

9:   Update the value function approximation:  $\hat{V}_t \leftarrow \text{Update}(\hat{V}_t, \mathbf{r}_t, \mathbf{s}_t, \pi_t)$  for all  $t \in \mathcal{T}$ .

10: **end for**

11: **return** The best solution found and its corresponding value function approximations.

---

## 4.1 Value Function Approximations

For dynamic resource allocation maximization problems, separable piecewise linear (PWL) concave value function approximations (VFAs) have enjoyed much success. Toriello et al. [39] note that PWL concave functions are appropriate for several reasons. From a modeling viewpoint, they can easily be embedded into a MIP (when solved as a maximization problem). From a practical perspective, concavity captures the diminishing returns one expects to gain from future inventories. Finally, from a theoretical perspective, they are the “closest” continuous functions to true MIP value functions, which are known to be piecewise linear, superadditive, and upper semi-continuous, but possibly discontinuous [3, 4]. Separability in space/location is also quite natural for problems in which vehicles always fully load and fully discharge at a single location [38, 37]. Meanwhile, separability in time is a fairly major issue and is less understood, but has proven to be effective in a stream of research papers for dynamic fleet management applications [15, 38, 37, 29]. It should be noted that ADP may struggle on DFMPs in which a load/activity can be served over multiple time periods.

In this work, we also use a value function approximation that is a separable piecewise linear continuous function, but we deviate by removing the concavity restriction. From a modeling point of view, removing this restriction offers more flexibility in the approximation. Meanwhile, this added flexibility introduces several concerns. A first concern is that the resulting optimization subproblem will be much more challenging to solve. As will be shown in our computational experiments, this turns out not to be the case for our basic implementation. General discontinuous piecewise linear functions can be modeled well using MIP techniques and, in our case, the resulting MIPs are easily solved in seconds or less. Another worry is that the lack of concavity will prolong the convergence of the algorithm. Indeed, concavity has been shown to accelerate the rate of convergence of ADP algorithms for certain problem classes [22, 23]. This, too, does not appear to be a major hurdle for our algorithm. A third complexity stems from the infamous “exploration vs. exploitation” question in ADP algorithms: Should we make a decision because, given our current value function approximation, it appears to be optimal or should we explore a new state to garner additional information (see, e.g., Chapter 12 of [26])? Without concavity, it may be necessary to introduce some form

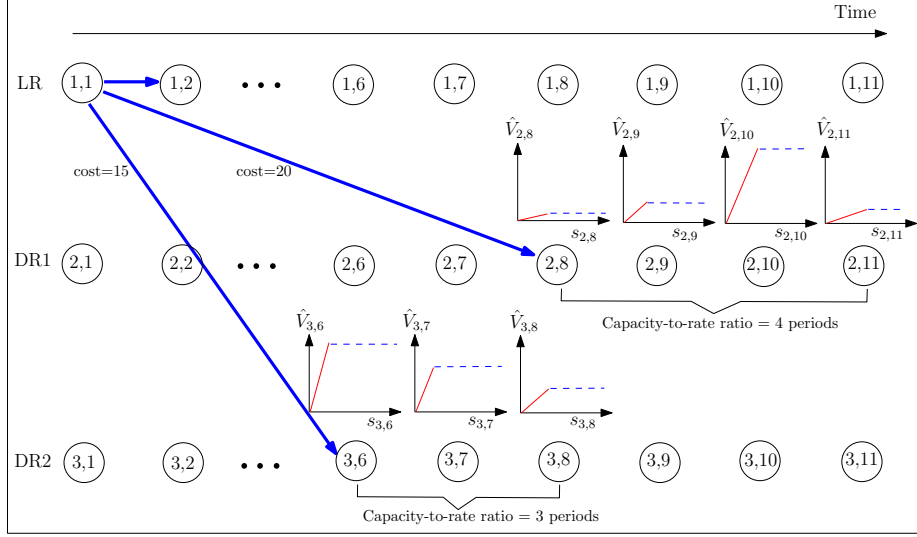


Figure 1: Example of dispatching decisions faced by the loading port in a given subproblem when only one vessel class is present. A separable piecewise linear concave value function approximation with two slopes exists at each node. The subscript  $t$  on the variables  $s_{j,u,t}$  has been omitted.

of exploration.

To construct our approximation, we replace the true value function  $V_{t+1}(\mathbf{r}_{t+1}, \mathbf{s}_t)$  with an approximate value function that depends only on future inventory levels at discharging ports:

$$\hat{V}_{t+1}(\mathbf{s}_t) = \sum_{j \in \mathcal{J}^C} \sum_{u \geq t + \tau_j} \hat{V}_{j,u}(s_{j,u,t}).$$

Here,  $\hat{V}_{j,u}$  is a univariate piecewise linear concave function defined by two slopes  $\hat{v}_{j,u}^1 \geq 0$  and  $\hat{v}_{j,u}^2 = 0$  and two breakpoints  $\beta_{j,u}^1 = d_{j,u}$  and  $\beta_{j,u}^2 = \infty$  for each discharging region  $j \in \mathcal{J}^C$  and for each time period  $u \in \mathcal{T}$ .  $\tau_j$  is the travel time between the loading region and discharging region  $j$ . No value is given to inventory in the loading region. Note that this approximation ignores the number of vessels that will be available in the future. Although it might at first seem like a significant amount of information is not being used, in fact, it is not the case. Since vessels always fully discharge, knowing the future amount of available inventory  $s_{j,u,t}$  at a discharging port is more useful than knowing the number of vessels in each vessel class that will make the delivery. On the other hand, some information is lost at the loading port, namely, the availability of future vessels to deliver product.

With this approximation, the term  $\hat{V}_{t+1}(\mathbf{s}_t)$  becomes

$$\sum_{j \in \mathcal{J}^C} \sum_{u \geq t + \tau_j} \sum_{k \in \{1,2\}} \hat{v}_{j,u}^k w_{j,u,t}^k,$$

where  $w_{j,u,t}^k$  are continuous decision variables that relate to the inventory variables  $s_{j,u,t}$  through the con-

straints

$$s_{j,u,t} = \sum_{k \in \{1,2\}} w_{j,u,t}^k \quad \forall n = (j,u) \in \mathcal{N} : j \in \mathcal{J}^C, u \geq t \quad (7a)$$

$$0 \leq w_{j,u,t}^k \leq \beta_{j,u}^k \quad \forall k \in \{1,2\}, \forall n = (j,u) \in \mathcal{N} : j \in \mathcal{J}^C, u \geq t. \quad (7b)$$

As a final approximation, rather than consider the value of all future inventories at a port  $j$  on and after time period  $t + \tau_j$ , we truncate the time horizon of the subproblem based on travel times and so-called capacity-to-rate ratios. In particular, let  $\text{C2R}_{j,t}$  be the capacity-to-rate ratio at discharging port  $j$  beginning in time period  $t$ , i.e., the number of periods it will take for port  $j$  to run out of inventory when starting full in time period  $t$ . Then, in time period  $t$ , we only value inventory up to time period  $t + u_{j,t}$  where  $u_{j,t} = \tau_j + \text{C2R}_{j,t} - 1$ . The rationale for this truncation is to avoid giving ports with a high consumption rate and a short travel time an artificially high reward for sending a vessel. For example, suppose there are two discharging ports and the travel times to port 1 and port 2 are 5 and 30 periods, respectively. Then our subproblem needs to consider at least 30 periods. Suppose the subproblem includes 35 periods. If port 1 has a high consumption rate and port 2 has a low consumption rate, then valuing future inventory at port 1 from time period 5 to 35 would make it very attractive to send vessels to port 1. This is a situation we would like to avoid. With this truncation, the approximation becomes

$$\hat{V}_{t+1}(\mathbf{s}_t) = \sum_{j \in \mathcal{J}^C} \sum_{u=t+\tau_j}^{t+u_{j,t}} \sum_{k \in \{1,2\}} \hat{v}_{j,u,t}^k w_{j,u,t}^k. \quad (8)$$

Figure 1 illustrates how our approximation is used. Given an available vessel in the loading region (LR) in time period 1, the time 1 subproblem considers the tradeoff between the immediate cost of moving the vessel and the reward associated with satisfying future demands. In this example, discharging region 1 (DR1) and 2 (DR2) have a capacity-to-rate ratio of four and three periods, respectively. The first slope of each VFA  $\hat{V}_{j,u}$  reflects the value of having inventory  $s_{j,u}$  in that specific period. By considering a single region  $j$  and summing over all future time periods in which a piecewise linear concave function is shown, we obtain a single value of future inventory  $\hat{V}_{t+1}(\mathbf{s}_t)$  that is weighted by the value of inventory in each period as shown in Equation (8).

Although our value function approximations at each node are shown to be piecewise linear concave with respect to the known future inventory at that node, the resulting approximation may not be concave with respect to the future inventory at time period  $t + \tau_j$ . Recall that vessels dispatched from the loading port in time period  $t$  will arrive at discharging port  $j$  at time period  $t + \tau_j$ . Because travel times are assumed to be known and identical for all vessel classes, all vessels dispatched in an earlier time period will arrive before time period  $t + \tau_j$ . Thus, the ending inventory in time period  $u > t + \tau_j$  at port  $j$  is related to the ending inventory at time period  $t + \tau_j$  by the equation  $s_{j,u,t} = [s_{j,t+\tau_j,t} - \sum_{u' > t+\tau_j}^u d_{j,u'}]^+$  where  $[a]^+ = \max\{0, a\}$ . Writing our VFA solely in terms of functions  $\hat{W}_{j,t+\tau_j}$  that depend on the ending inventories at time  $t + \tau_j$ , we obtain an equivalent representation of our VFA as the sum of nonconvex PWL functions:

$$\begin{aligned} \hat{V}_{t+1}(\mathbf{s}_t) &= \sum_{j \in \mathcal{J}} \sum_{u \geq t+\tau_j} \hat{V}_{j,u}(s_{j,u,t}) \\ &= \sum_{j \in \mathcal{J}} \hat{W}_{j,t+\tau_j}(s_{j,t+\tau_j,t}). \end{aligned}$$

An example of this interpretation is shown in Figure 2.

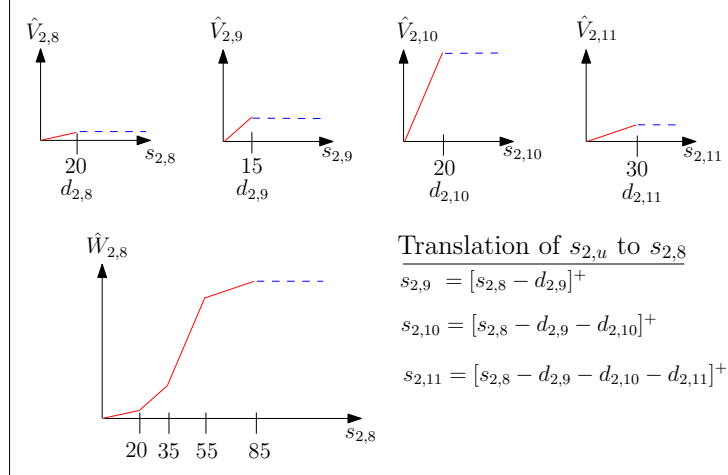


Figure 2: From concave to nonconvex piecewise linear value function approximations. In terms of ending inventory at each node, a separable PWL concave VFA with two slopes is used. In terms of ending inventory in the time period a delivery is made, a single PWL nonconvex VFA is used.

In summary, our time  $t$  MIP subproblem is

$$\hat{V}_t(\mathbf{r}_t, \mathbf{s}_{t-1}) = \max \sum_{vc \in \mathcal{VC}} \sum_{j \in \mathcal{J}} \sum_{a \in \mathcal{FS}_{(j,t)}^{vc}} - C_a^{vc} x_a^{vc} - \sum_{j \in \mathcal{J}} P_{j,t} \alpha_{j,t} + \sum_{j \in \mathcal{J}^C} \sum_{u \geq t + \tau_j} \sum_{k \in \{1,2\}} \hat{v}_{j,u}^k w_{j,u,t}^k \quad (9a)$$

$$\text{s.t. } (2), (4), (5c) - (5f), (7) \quad (9b)$$

$$s_{j,u,t} \alpha_{j,u} = 0, \quad \forall n = (j, u) \in \mathcal{N} : u \geq t, \quad (9c)$$

where we linearize the constraint  $s_{j,u,t} \alpha_{j,u} = 0$  using big  $M$  parameters:  $s_{j,u,t} \leq M y_{j,u,t}$  and  $\alpha_{j,u} \leq M(1 - y_{j,u,t})$  where  $y_{j,u,t}$  is a binary decision variable taking value 1 if  $s_{j,u,t}$  is positive and 0 if  $\alpha_{j,u}$  is positive.

It is important to mention that our approach does not allow vessels to leave the system until the very last time period of the horizon and therefore there is no value for this option. Consequently, some needless trips at the end of the horizon may take place. The rationale for removing the option to take a vessel out of service is due to the fact that, in our instances, there is not an overabundance of vessels and so all vessels are continually in operation. Moreover, some might argue that our problem is an infinite horizon problem and should not be truncated. Thus, as a final step in our solution approach, we have a simple routine, which we call “end effect polishing,” to remove these needless trips that are an artifact of the finite horizon.

## 4.2 Updating the Value Function Approximation

Having described our value function approximation, we turn to the question of how we update it from iteration to iteration. Although our VFA is relatively simple, i.e., the only parameters that may be changed are the slopes  $\hat{v}_{j,u}^1$ , devising a suitable updating scheme takes some care.

Just as there are numerous choices for designing a value function approximation, there are also a number of techniques commonly found for updating the VFAs (see, e.g., George and Powell [13]). Perhaps, the

most important consideration is to determine what the goal of the update is. In early iterations of an ADP algorithm, it is often beneficial to explore the solution space. Thus, it is usually preferred to have a fast update rule that results in substantial changes to the value function. On the other hand, in later iterations, some sort of convergence is often desired, in which case small changes are sought. Regression and batch least-squares are sometimes used [26]. Toriello et al. [39] suggest other fitting procedures.

Our focus is on generating one or more good solutions quickly; convergence is less of a concern. Consequently, we prefer rapid updates over those that require a significant amount of fitting at each iteration. To this end, we update each slope  $\hat{v}_{j,u}^1$  using a generic smoothing technique:

$$\hat{v}_{j,u}^{1,(n+1)} = (1 - \gamma_n)\hat{v}_{j,u}^{1,(n)} + \gamma_n\nu_{j,u}^{(n)}.$$

In words, to obtain the slope  $\hat{v}_{j,u}^{1,(n+1)}$  to be used on iteration  $n + 1$ , we take a convex combination of the previous estimate of the slope  $\hat{v}_{j,u}^{1,(n)}$  with some new estimate  $\nu_{j,u}^{(n)}$  obtained from the solution at iteration  $n$ . Here,  $\gamma_n \in [0, 1]$  is a parameter that depends on the iteration.

With this updating procedure, there are two issues: the choice of the stepsize  $\gamma_n$  and how the new estimate  $\nu_{j,u}^{(n)}$  of the slope is obtained. We found that a simple harmonic stepsize rule  $\gamma_n = 1/(C + n)$  was adequate where  $C$  is some positive integer. In fact, our methods exhibited similar performance for value of  $C \in \{1, \dots, 10\}$ . Consequently,  $C = 1$  in all of our computation results. The more challenging issue was how to obtain the new estimate of the slope.

Our first attempt at obtaining new estimates of the slopes  $\hat{v}_{j,u}^1$  was based on using the values of the dual variables  $\pi_{j,u,t}$  associated with each inventory balance equation (4). For the dynamic fleet management problem, this approach has been shown to work well because the subproblem can be recast as a network flow problem where dual information is readily available [14, 15]. When the subproblem is a MIP, it is not clear whether similar dual information can be obtained. Bouzaiene-Ayari et al. [5] solve MIP subproblems and report good results from updating the slopes of their VFAs using the duals of the LP relaxation. In all of our attempts, however, using the values of the dual variables  $\pi_{j,u,t}$  from the LP relaxation was not productive.

Ultimately, we turned to an updating rule that we believe is quite simple and works well, but may not be applicable for other mainstream problems. One of the dominant components in the objective function is the penalty incurred for stocking out at a discharging port. Thus, after obtaining a solution to the full planning problem, we identified all nodes at which a stockout occurred by observing the spot market quantities  $\alpha_{j,u}$ . We then set  $\nu_{j,u}^{(n)} = P_{j,u}\alpha_{j,u}$ , i.e., the marginal value of an additional unit of inventory is proportional to the price paid to satisfy the stockout in the period.

### 4.3 A Multiperiod Lookahead

There are at least two potential drawbacks of the basic ADP approach outlined above. First, our value function approximation  $\hat{V}_t$  is not parameterized by the vector  $\mathbf{r}_t$  and thus completely ignores the information related to future vessel arrivals at the loading port. Second, regardless of the VFA chosen, the success of the algorithm relies almost entirely on the quality of the VFA. That is, by choosing to construct a solution to a long-horizon problem by solving a sequence of single-period subproblems, we are dependent on the VFAs used in each subproblem to help us make wise dispatching decisions that ultimately lead to a good solution.

In order to decrease this dependence on the VFAs, we also explore the strategy of solving multiperiod subproblems. Powell refers to this as a hybrid procedure under the name ‘‘Rolling Horizon Procedures with Value Function Approximations’’ (see Section 6.5.2 of [26]). The idea is straightforward. In the

time  $t$  subproblem, rather than consider routing decisions for only those vessels that are available to be dispatched in time period  $t$ , consider instead decisions for all vessels available for dispatching in time periods  $t, t + 1, \dots, t + H$ , where  $H$  is the lookahead. Once this more complex subproblem is solved, fix only those dispatching decisions for vessels dispatched in time period  $t$ , and continue to the next subproblem starting in time period  $t + 1$ . The hope is that by looking ahead to see which vessels will be available in the near future as well as how these vessels will be dispatched, our routing decisions for vessels in time period  $t$  will be improved.

For a multiperiod lookahead of  $H$  periods, we modify the value function approximation as follows:

$$\hat{V}_{t+1}(\mathbf{s}_t) = \sum_{j \in \mathcal{J}^C} \sum_{u=t+H+\tau_j}^{t+H+u_{j,t,H}} \sum_{k \in \{1,2\}} \hat{v}_{j,u,t}^k w_{j,u,t}^k. \quad (10)$$

Approximation (10) is the same as the VFA in (8) with modifications to the second summation. First, since the last period in which a vessel can arrive at port  $j$  is  $t + H + \tau_j$ , we sum over future inventories from time period  $t + H + \tau_j$  onwards (as opposed to  $t + \tau_j$  onwards). Second, since multiple vessels may be dispatched to the same region and, therefore, discharge more product than in the no lookahead case, we extend the number of time periods over which we value future inventory using the parameter  $u_{j,t,H} (\geq u_{j,t})$ . In our computational experiments below, we set  $u_{j,t,H} = u_{j,t} + 5$  as this parameter setting gave the best performance, on average, over all settings with  $u_{j,t,H} = u_{j,t} + k$  for  $k \in \{0, 1, \dots, 10\}$ .

#### 4.4 Optimization-based Local Search heuristic for MIP Model (1)

We attempt to solve MIP Model (1) with two methods. These methods will act as a benchmark for our ADP algorithm. Our first approach is to solve MIP Model (1) directly using a MIP solver. One could also tighten the formulation by appending, for example, lot-sizing based cuts to the model as presented in [25]. In our experiments, we found that adding such cuts to the model improved the dual bound, but did not help the solver find good feasible solutions faster; in fact, they often hampered primal performance.

Our second approach is to apply an optimization-based heuristic introduced by Song and Furman [33] for MIRPs, but discussed for another application in [30]. In their setting, decisions for each individual vessel are modeled. Their local search procedure is akin to a 2-opt procedure and works as follows: after obtaining an initial feasible solution, the decision variables associated with all but two vessels are fixed and an exact optimization algorithm is called to locally optimize the decisions for these two vessels. This procedure is applied for up to  $\binom{|\mathcal{V}|}{2}$  iterations, where  $|\mathcal{V}|$  is the number of vessels and vessel pairs are chosen randomly in each iteration. Goel et al. [16] adapt this local search procedure to generate solutions to MIRP instances with 365 time periods. Their main algorithmic contribution is to show how vessel pairs should be chosen to improve solution quality and reduce total solution time. This local search is also applied in [21].

Since our problem does not deal with individual vessels, we modified the above approach to work with vessel classes. In our implementation, we construct an initial feasible solution by solving MIP Model (1) directly using a MIP solver for up to 30 seconds. Although one could employ other techniques to construct an initial feasible solution, we believe that this approach is the most sensible for comparison since it does not involve any additional algorithms to be implemented. Moreover, in other applications, the local search procedure's performance has been shown to be nearly independent of the starting solution [16]. With a feasible solution in hand, we fix all routing decisions associated with all but one vessel class and locally optimize the routing decisions for that one vessel class. We optimize one vessel class at a time because there

are multiple vessels within each class (thus, multiple vessels are simultaneously being re-routed) and for large instances simultaneously optimizing two vessel classes led to challenging MIPs. Since the resulting MIP is still challenging to solve to provable optimality, we set a time limit of 30 seconds. For ease of implementation, we simply cycle through the vessel classes one at a time for a given time limit. Given the past success and popularity of this type of heuristic, we believe this implementation offers a meaningful benchmark.

## 4.5 Combining ADP and Local Search

Finally, we describe another hybrid procedure of potential interest. In many instances, one of our ADP methods quickly finds high quality solutions within the first few iterations. However, it may be the case that our ADP “gets stuck” and is unable to find a solution of higher quality in the remaining iterations. This happens when the slope updates are unable to drive the ADP to a better set of slopes. To find better solutions, we apply the local search described above to the solution returned by our ADP after  $N$  iterations. In this setting, we treat local search like a post-processing step to correct local suboptimalities in our best ADP solution.

## 5 Computational Experiments

In this section, we compare the performance of variants of our ADP method with that of the commercial MIP solver Gurobi 5.0 solving the MIP Model (1) and the MIP-based local search procedure outlined in Section 4.4. In all experiments in which the performance of Gurobi and local search are evaluated, we set Gurobi’s MIPFocus parameter to 1 to emphasize feasibility so that more time and effort is spent trying to find good feasible solutions. Gurobi’s default settings were used to solve the MIP subproblems in our ADP methods. All value function approximations are initialized with zero slopes and no off-line training is performed. All models and algorithms were coded in Python and run on a single thread. All experiments were carried out on a Linux machine with kernel 2.6.18 running a 64-bit x86 processor equipped with two 2.27 GHz Intel Xeon E5520 chips and 32GB of RAM.

Because we set a time limit of 30 seconds to search over each neighborhood in our local search implementation, our local search is no longer deterministic due to idiosyncrasies of MIP solvers. Therefore, results in which local search is used are averaged over ten runs. In general, we observed little variability across these runs.

Computational experiments were conducted on a subset of instances from the Maritime Inventory Routing Problem Library (MIRPLib) [25], a library of MIRPs available at [mirplib.scl.gatech.edu](http://mirplib.scl.gatech.edu). These instances are inspired by real-world MIRPs, but do not represent any particular real-world data set. The 24 instances (labeled “Group 2 Instances” in MIRPLib) can be categorized as easy, moderate, or hard. The easy instances include two discharging ports, whereas the hard instances involve as many as 12 discharging ports, 10 vessel classes, and one-way travel times of 37 periods.

The main metric that we use to compare solution methods is the average fraction of the relative gap closed over time, where the average is taken over all 24 instances. Here the relative gap is defined as  $(z^{\text{method}} - z^{\text{best}})/z^{\text{method}}$ , where  $z^{\text{method}}$  is the objective function value of the method used (e.g., ADP, local search, or Gurobi emphasizing feasibility) and  $z^{\text{best}}$  is the objective function value of the best known solution found after days of computing. Specifically, for a particular instance, the value  $z^{\text{best}}$  was computed by warm-starting the local search procedure with the best ADP solution found, applying local search for five



hours, and then warm-starting Gurobi with this solution and solving for 10 days.

Table 1: Algorithms Compared

Algorithm	Description	Section
ADP_LA0	Basic ADP with no lookahead	4.1
ADP_LAH	ADP with a lookahead of $H$ periods	4.3
LS	Local Search	4.4
GRB	Gurobi 5.0 emphasizing feasibility	4.4
ADP_LA0_LS	Basic ADP followed by local search	4.5

We compare the various algorithms listed in Table 1 on instances with a 120-period horizon. Although LNG-IRPs usually require one to solve instances with horizons of 90 to 365 periods, we believe that using a 120-period horizon gives the strongest algorithmic comparison for two reasons. First, we performed preliminary experiments in which we attempted to solve MIP Model (1) with a 180- and 360-period horizon directly using Gurobi or by using local search, but found ADP to be vastly superior. Indeed, for instances with a 360-period horizon, our ADP with no lookahead could close, on average, 81% of the relative gap in 800 seconds, whereas Gurobi and local search could close 27% and 56%, respectively, in 1800 seconds.

Our second reason for focusing on 120-period horizon instances is because they offer a reasonable comparison with rolling horizon heuristics, arguably the most common heuristics applied to long-horizon problems. For example, to generate solutions to planning problems with a 365-period horizon, Rakke et al. [28] solve subproblems involving 90 periods and piece together the solutions to these overlapping subproblems to create a solution for the full planning horizon. For several of our instances, one-way inter-regional travel times are over 30 periods in duration and we found that solving a reduced MIP with a 90-period time horizon could lead to solutions with odd end behavior. Extending these horizons over 120 periods seemed to yield more stable results. Preliminary testing on several 360-period horizon instances showed that, when incorporated into a rolling horizon framework, our ADP could close, on average, 91% of the relative gap compared to 84% (only some easy and medium instances were compared) without the rolling horizon framework. We believe this experiment supports our claim that ADP can find good solutions to 360-period instances. The implementation of our rolling horizon heuristic was naive and only semi-automated. We solved six 120-period instances rolling forward 60 periods after each solve. Rather than attempt to improve this rolling horizon implementation further, we decided to focus on solving 120-period instances to better understand the performance of our ADP algorithm.

## 5.1 Comparison of ADP with Traditional MIP Methods

Figure 3 shows the average fraction of the relative gap closed over time by four methods. The most important observation is that our basic ADP with no lookahead (ADP\_LA0) outperforms both Gurobi emphasizing feasibility and the local search procedure with respect to solution time (time to best) and quality. ADP\_LA0 nearly reaches its best solution within roughly two to three minutes, whereas Gurobi and local search require more time. Moreover, on average, the quality of the ADP solution is 92% of the best known, while local search is near 84% and Gurobi is at 72% after 30 minutes of CPU time.

A second observation is the stagnation of ADP\_LA0 and local search after a given amount of time. In

other words, after 10 to 30 iterations, ADP with no lookahead is unable to find a set of slopes that produce an improving solution. In light of the fact that our primary goal is to find good solutions quickly, this observation is not disconcerting since this goal is achieved in two or three minutes. However, if more time is available, we would like to improve the solution quality further. One option of overcoming this stagnation is to apply local search on the best solution found by an ADP method. The performance of this approach is shown in Figure 3 where we see that it is capable of closing 96% of the gap, on average, after 900 seconds.

Another way to overcome this stagnation is to introduce more deliberate exploitation and exploration schemes. We attempted to achieve this by modifying the ADP slope updating procedure. Borrowing from other popular heuristics (e.g., tabu search), we implemented an intensification strategy in which, after a certain number of iterations, we set the slopes used in the VFA equal to those that produced the best known solution found thus far in the search process. Like other intensification strategies, the hope is that by returning to a good set of slopes at a later iteration when the stepsize  $\gamma_n$  is smaller, the slope updates would stay closer to those that produced the best solution and find a better neighboring solution. We also experimented with a diversification strategy. Let  $rgap_n = \min\{(z_n - z^{\text{inc}})/z_n, 1\}$  denote the relative gap computed on the  $n$ th iteration of the ADP algorithm, where  $z^{\text{inc}}$  is the objective function value of the incumbent solution (the best solution found by iteration  $n$ ) and  $z_n$  is the objective function value found on the  $n$ th iteration of the algorithm. After a certain number of iterations, we altered the stepsize updating rule to  $\gamma_n = (C + (1 - rgap_n)n)^{-1}$ . If the objective function value  $z_n$  of the solution found in iteration  $n$  is

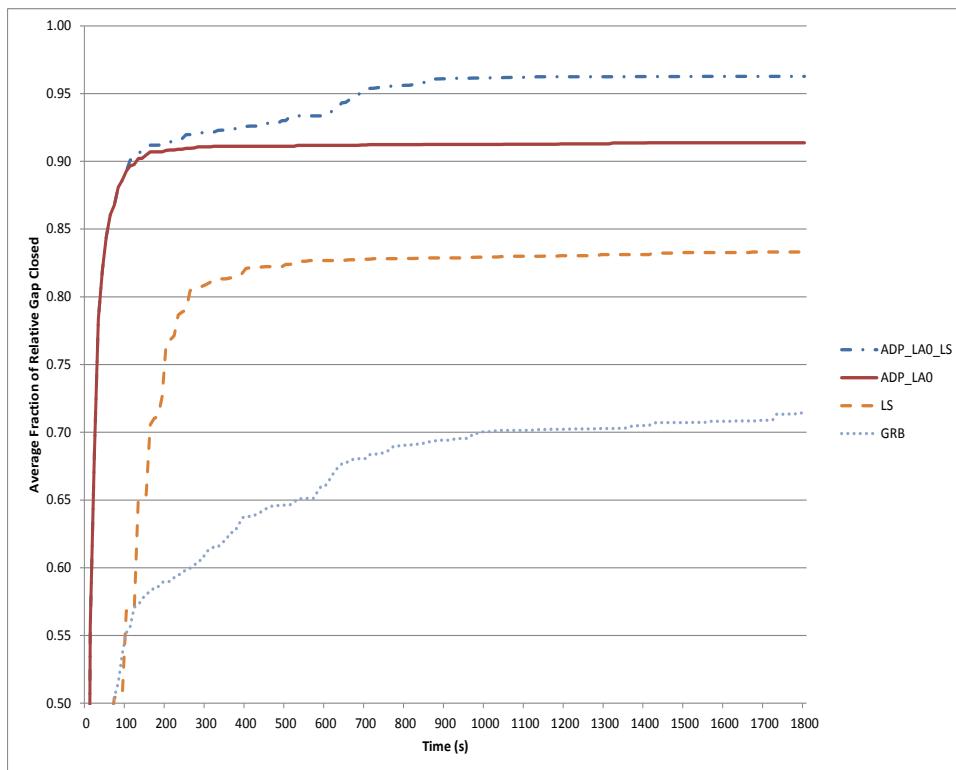


Figure 3: Comparison of our basic ADP with no lookahead (ADP\_LA0), our basic ADP followed by local search (ADP\_LA0\_LS), Gurobi (GRB), and Local Search (LS).

poor, we expect  $rgap_n$  to be closer to 1 and so the stepsize will be close to  $1/C$ , leading to a more drastic change in the VFA updates. On the other hand, if  $rgap_n$  is small, the stepsize will be closer to  $1/(C+n)$  so that the VFA updates are modest. In our experiments, these intensification and diversification strategies did not produce better results. Small improvements could be made on some instances, but typically at the expense of deteriorated performance on other instances.

Table 2 shows the time required for our ADP algorithm to find its best solution and the additional time that local search and Gurobi needed to find a solution of equal or better quality. A ‘>3600’ or ‘>18000’ means that local search or Gurobi could not find a better solution within a 1-hour or 5-hour time limit, respectively. A negative value means that a better solution was found in less time than in our ADP method. We see that ADP with no lookahead performs best on 21 of the 24 instances. In three instances, local search is able to find a better solution than ADP. In all but one instance, Gurobi requires more time to find a solution of equal or better quality.

Our convention for naming instances is based on the number of loading and discharging regions, the

Table 2: Instance-by-instance comparison: Additional time (sec) required by local search (LS) and Gurobi (GRB) to reach a solution of equal or better quality.

Instance	Time to Best		Additional Time	
	ADP	LA0	LS	GRB
LR1_DR02_VC01_V6a	0	>3600	>3600	456
LR1_DR02_VC02_V6a	15	-5	-5	-5
LR1_DR02_VC03_V7a	9	>3600	>18000	>18000
LR1_DR02_VC03_V8a	17	>3600	93	93
LR1_DR02_VC04_V8a	9	>3600	208	208
LR1_DR02_VC05_V8a	1	>3600	10823	10823
LR1_DR03_VC03_V10b	84	>3600	5016	5016
LR1_DR03_VC03_V13b	21	>3600	>18000	>18000
LR1_DR03_VC03_V16a	13	16	220	220
LR1_DR04_VC03_V15a	96	>3600	195	195
LR1_DR04_VC03_V15b	193	>3600	>18000	>18000
LR1_DR04_VC05_V17a	25	>3600	471	471
LR1_DR04_VC05_V17b	72	85	40	40
LR1_DR05_VC05_V25a	37	>3600	>18000	>18000
LR1_DR05_VC05_V25b	198	117	435	435
LR1_DR08_VC05_V38a	707	>3600	>18000	>18000
LR1_DR08_VC05_V40a	26	>3600	>18000	>18000
LR1_DR08_VC05_V40b	520	-302	1973	1973
LR1_DR08_VC10_V40a	46	>3600	>18000	>18000
LR1_DR08_VC10_V40b	277	>3600	>18000	>18000
LR1_DR12_VC05_V70a	81	>3600	>18000	>18000
LR1_DR12_VC05_V70b	1400	-1011	>18000	>18000
LR1_DR12_VC10_V70a	1180	>3600	>18000	>18000
LR1_DR12_VC10_V70b	1314	>3600	>18000	>18000

number of ports, the number of vessel classes, and the number of vessels. This convention is best understood with an example. Consider an instance named LR1\_DR03\_VC05\_V16b. LR1 means that there is one loading region/port. DR03 means that there are three discharging regions/ports. VC05 means that there are five vessel classes. V16 means that there are a total of 16 vessels (with at least one vessel belonging to each vessel class). Finally, if a letter is included at the end, this is to distinguish this instance from other instances.

As a final comment, we believe that our ADP method becomes more attractive as the number of vessel classes increases. This is because the time it takes a MIP solver to solve the MIP Model (1) directly or using local search should increase more rapidly with an increase in vessel classes than that of our ADP algorithm. Since some applications may not allow vessels to be aggregated by vessel class, our ADP approach may become more appealing if each individual vessel must be modeled.

## 5.2 Comparison of ADP Methods With and Without a Lookahead

We also attempted to improve our basic ADP method by including a lookahead of  $H = 1, 2,$  or  $3$  periods. As discussed in Section 4.3, our VFA ignores the availability of vessels at the loading port in future periods. By including a multiperiod lookahead of  $H$  periods, dispatching decisions for vessels available in time periods  $t, \dots, t + H$  are considered. As a consequence, we would hope that each subproblem would be less myopic as more information is at hand.

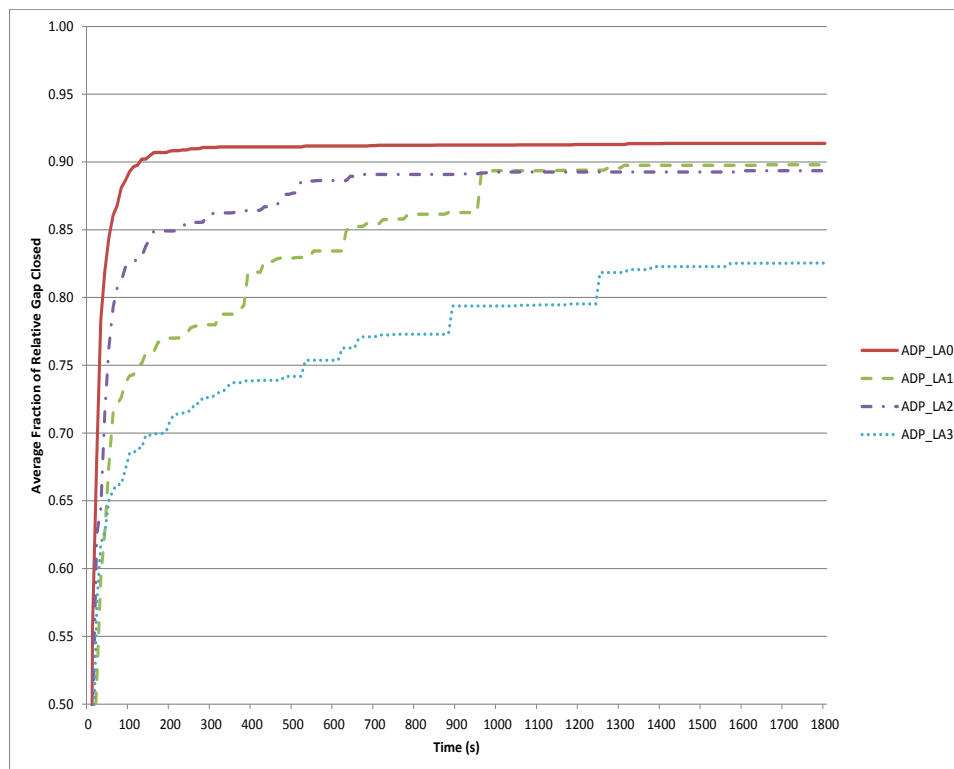


Figure 4: Comparison of our basic ADP with no lookahead (ADP\_LA0) with ADP methods using a 1-, 2-, and 3-period lookahead.

Our experiments reveal a counterintuitive result. As shown in Figure 4, given our choice of VFAs and slope updates, a multiperiod lookahead of one to three periods performs worse, *on average*, than having no lookahead. We observed that in a small number of instances a lookahead could outperform an ADP with no lookahead, both in terms of solution time and quality. One might ask if this inferior average performance is because each major iteration (Step 2 of Algorithm 1) of an ADP method with a lookahead requires more time since each MIP subproblem is more computationally demanding. This is not the case. Although we only show the first 30 minutes of computation time, in all but a few cases, the ADP methods with a lookahead (after hours of computing) still failed to achieve a comparable average solution quality as our basic ADP method with no lookahead.

One possible explanation for this lack of improvement is that a longer lookahead (i.e., setting  $H > 3$ ) is required to take advantage of future information for these instances. Because the median one-way travel time is typically between 15 and 18 periods (giving rise to round-trip travel times of 30 to 36 periods), a lookahead of  $H = 1, 2$ , or 3 periods may not capture any additional relevant information than having no lookahead at all. The problem with increasing the lookahead, however, is that each subproblem becomes even more time consuming, thus making it difficult to find good solutions quickly.

We also tried warm-starting a multiperiod lookahead with the best VFA found by our basic ADP with no lookahead. This strategy is similar to the intensification idea described above. Specifically, after applying our basic ADP method with no lookahead for  $N = 100$  iterations, we initialized the VFA of an ADP with a lookahead with the best VFAs, i.e., the set of slopes that produced the best solution over 100 iterations, found by the ADP with no lookahead. Even after insisting that smaller stepsizes take place, this approach still failed, on average, to generate solutions of equal or better quality than ADP\_LA0.

Table 3: Average percentage of time spent in each ADP-related function

<b>Function</b>	<b>% of Time</b>
MIP initialization	21.58
MIP solving	47.51
State updating	17.64
End effect polishing	6.02
Slope updating	0.84
Miscellaneous	6.41

### 5.3 Profiling the ADP Algorithm

Table 3 shows the percentage of time our basic ADP algorithm with no lookahead spends in each of its major functions, averaged over all instances and all time horizons considered. Almost 70% of the solution time is spent either initializing or solving all of the MIP subproblems. This large percentage is expected since MIP solving is costly compared to all of the other operations. However, a more intelligent implementation should be able to shrink the percentage of time spent on MIP initialization since this involves nothing more than several loops. End effect polishing is discussed at the end of Section 4.1 and refers to a simple procedure that we apply after obtaining a solution for the full horizon. Since our algorithm does not allow vessels to leave the system, needless trips at the end of the horizon may take place. End effect polishing seeks to remove

these needless trips that are an artifact of truncating what some might argue is an infinite horizon problem. In our approach, value function (or slope) updating requires virtually no time as convex combinations of information are used. This percentage of time would increase if one were to use more sophisticated schemes for updating the value function, e.g., regression.

## 6 Conclusions and Future Work

This paper introduced an approximate dynamic programming framework for generating good solutions quickly to a class of maritime inventory routing problems with a long planning horizon. The ADP approach appears to be one of the first of its kind in the maritime routing and scheduling domain and represents a significant departure from previous methods for this class of problems. Rather than putting the burden on a MIP solver to produce good solutions or improving solutions, our approach shifts this effort to identifying value function approximations that lead to good solutions. Computational experiments indicate that this framework is capable of obtaining better solutions than a commercial MIP solver and a popular local search method tasked with considering many periods simultaneously.

Regarding future research directions, Powell [26] and his associates have laid the groundwork for an algorithmic approach that can incorporate stochastic elements, e.g., stochastic demands, travel times, etc. Although we have not explored these extensions, the attractive feature of the proposed ADP framework is that it requires minor changes. In particular, it considers sample realizations of the uncertain elements to solve each time  $t$  subproblem, and then proceeds as normal. When the value function approximations are updated with a convex combination procedure as we have done, the stepsize may become dependent on the noise in the estimates obtained over the course of the algorithm.

Our framework concentrates on a setting involving a single loading port/region. Considering multiple loading ports/regions would be a useful extension and address an industrial setting that is likely to become more prevalent. At the same time, the presence of multiple loading ports introduces more complicated routing decisions. In addition, we would have to overcome what Godfrey and Powell [15] refer to as the “long-haul bias”, a phenomenon in which more costly decisions are often made to satisfy high value opportunities before less costly ones could be made to satisfy the same opportunities.

Another interesting experiment would be to assess the benefit from storing value function approximations when the model is re-optimized. For example, within the context of a general decision support tool that is called every month to obtain an updated long-term plan, it seems likely that warm-starting our ADP framework with the best known value function approximations would lead to better solutions faster.

## Acknowledgments

We wish to thank Warren Powell for references to recent ADP research as well as helpful “lessons from the field” that helped us avoid common pitfalls. We also thank Belgacem Bouzaiene-Ayari for instructive algorithmic suggestions related to ADP implementation. We are grateful to two anonymous referees for their perceptive comments that helped improve the quality of the paper.

## References

- [1] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536, 2010.
- [2] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
- [3] C. Blair and R. Jeroslow. The value function of a mixed integer program: I. *Discrete Mathematics*, 19(2):121–138, 1977.
- [4] C. Blair and R. Jeroslow. The value function of a mixed integer program: II. *Discrete Mathematics*, 25(1):7–19, 1979.
- [5] B. Bouzaiene-Ayari, C. Cheng, S. Das, R. Fiorillo, and W. B. Powell. From single commodity to multi-attribute models for locomotive optimization: A comparison of integer programming and approximate dynamic programming. *Submitted for publication*, 2013.
- [6] A. Campbell, L. Clarke, A. Kleywegt, and M. W. P. Savelsbergh. The inventory routing problem. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 95–113. Kluwer, 1998.
- [7] M. Christiansen and K. Fagerholt. Maritime inventory routing problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1947–1955. Springer-Verlag, 2nd edition, 2009.
- [8] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483, 2013.
- [9] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory-routing. *Transportation Science*, 48(1):1–19, 2014.
- [10] F. G. Engineer, K. C. Furman, G. L. Nemhauser, M. W. P. Savelsbergh, and J.-H. Song. A Branch-Price-And-Cut algorithm for single product maritime inventory routing. *Operations Research*, 60(1):106–122, 2012.
- [11] M. Fodstad, K. T. Uggen, F. Rømo, A. Lium, and G. Stremersch. LNGScheduler: a rich model for coordinating vessel routing, inventories and trade in the liquefied natural gas supply chain. *Journal of Energy Markets*, 3(4):31–64, 2010.
- [12] K. C. Furman, J.-H. Song, G. R. Kocis, M. K. McDonald, and P. H. Warrick. Feedstock routing in the ExxonMobil downstream sector. *Interfaces*, 41(2):149–163, 2011.
- [13] A. George and W. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine learning*, 65(1):167–198, 2006.
- [14] G. A. Godfrey and W. B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times. *Transportation Science*, 36(1):21–39, 2002.
- [15] G. A. Godfrey and W. B. Powell. An adaptive dynamic programming algorithm for dynamic fleet management, II: Multiperiod travel times. *Transportation Science*, 36(1):40–54, 2002.

- [16] V. Goel, K. C. Furman, J.-H. Song, and A. S. El-Bakry. Large neighborhood search for LNG inventory routing. *Journal of Heuristics*, 18(6):821–848, 2012.
- [17] V. Goel, M. Slusky, W.-J. van Hoesve, K. C. Furman, and Y. Shao. Constraint programming for LNG ship scheduling and inventory management. *Submitted for publication*, 2014.
- [18] R. Grønhaug and M. Christiansen. Supply chain optimization for the liquefied natural gas business. *Innovations in Distribution Logistics, Lecture Notes in Economics and Mathematical Systems*, 619:195–218, 2009.
- [19] R. Grønhaug, M. Christiansen, G. Desaulniers, and J. Desrosiers. A Branch-and-Price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415, 2010.
- [20] E. Halvorsen-Weare and K. Fagerholt. Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. *Annals of Operations Research*, 203(1):167–186, 2013.
- [21] M. Hewitt, G. L. Nemhauser, M. W. P. Savelsbergh, and J.-H. Song. A Branch-and-Price guided search approach to maritime inventory routing. *Computers & Operations Research*, 40(5):1410–1419, 2013.
- [22] J. M. Nascimento and W. B. Powell. An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Mathematics of Operations Research*, 34(1):210–237, 2009.
- [23] J. M. Nascimento and W. B. Powell. An optimal approximate dynamic programming algorithm for concave, scalar storage problems with vector-valued controls. *Automatic Control, IEEE Transactions on*, 58(12):2995–3010, 2013.
- [24] D. J. Papageorgiou, A. B. Keha, G. L. Nemhauser, and J. Sokol. Two-stage decomposition algorithms for single product maritime inventory routing. *To appear in INFORMS Journal on Computing*, 2014.
- [25] D. J. Papageorgiou, G. L. Nemhauser, J. Sokol, M.-S. Cheon, and A. B. Keha. MIRPLib – A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. *European Journal of Operational Research*, 235(2):350–366, 2014.
- [26] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley-Interscience, 2nd edition, 2011.
- [27] J. Rakke, H. Andersson, M. Christiansen, and G. Desaulniers. Branch-price-and-cut for creating an annual delivery program of multi-product liquefied natural gas. *To appear in Transportation Science*, 2014.
- [28] J. G. Rakke, M. Stålhane, C. R. Moe, M. Christiansen, H. Andersson, K. Fagerholt, and I. Norstad. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C: Emerging Technologies*, 19(5):896–911, 2011.
- [29] A. Ruszczyński. Post-decision states and separable approximations are powerful tools of approximate dynamic programming. *INFORMS Journal on Computing*, 22(1):20–22, 2010.
- [30] M. W. P. Savelsbergh and J. Song. An optimization algorithm for the inventory routing problem with continuous moves. *Computers & Operations Research*, 35(7):2266–2282, 2008.



- [31] Y. Shao, K. C. Furman, V. Goel, and S. Hoda. Bound improvement for LNG inventory routing. *Submitted for publication*, 2014.
- [32] H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application. *Transportation Science*, 43(2):178–197, 2009.
- [33] J.-H. Song and K. C. Furman. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 40(3):657–665, 2013.
- [34] M. Stålhane, J. G. Rakke, C. R. Moe, H. Andersson, M. Christiansen, and K. Fagerholt. A construction and improvement heuristic for a liquefied natural gas inventory routing problem. *Computers & Industrial Engineering*, 62(1):245–255, 2012.
- [35] R. Sutton and A. Barto. *Reinforcement learning: An introduction*. Cambridge University Press, 1998.
- [36] H. Topaloglu. A parallelizable dynamic fleet management model with random travel times. *European Journal of Operational Research*, 175(2):782–805, 2006.
- [37] H. Topaloglu. A parallelizable and approximate dynamic programming-based dynamic fleet management model with random travel times and multiple vehicle types. In V. S. Zeimpekis, G. M. Giaglis, C. D. Tarantilis, and I. Minis, editors, *Dynamic Fleet Management: Concepts, Systems, Algorithms and Case Studies*, pages 65–93. Springer, 2007.
- [38] H. Topaloglu and W. B. Powell. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1):31–42, 2006.
- [39] A. Toriello, G. L. Nemhauser, and M. W. P. Savelsbergh. Decomposing inventory routing problems with approximate value functions. *Naval Research Logistics*, 57(8):718–727, 2010.
- [40] K. Uggen, M. Fodstad, and V. Nørstebø. Using and extending fix-and-relax to solve maritime inventory routing problems. *TOP*, 21(2):355–377, 2013.