TIME DECOMPOSITION OF MULTI-PERIOD SUPPLY CHAIN MODELS

A Thesis
Presented to
The Academic Faculty

by

Alejandro Toriello

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2010

TIME DECOMPOSITION OF MULTI-PERIOD SUPPLY CHAIN MODELS

Approved by:

Professor George L. Nemhauser,
Advisor
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Professor Martin W.P. Savelsbergh
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Professor Shabbir Ahmed
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Professor Santanu Dey
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Dr. Ahmet Keha
Corporate Strategic Research
ExxonMobil Research & Engineering

Date Approved: July 29, 2010

To my grandparents,

Eugenia Bedard,

Eduardo Herrerías Estrada,

María Mercedes Nájera de Toriello,

and to the memory of my grandfather,

Lionel Toriello Saravia.

To my loving, patient wife, Catherine.

ACKNOWLEDGEMENTS

I owe many thanks to many people.

To Professor George Nemhauser, for his guidance and support as my advisor. To the other members of the ISyE faculty and of Georgia Tech in general who taught me, advised me and worked with me during my time here: Shabbir Ahmed, Bill Cook, Santanu Dey, Richard Duke, Özlem Ergun, Renato Monteiro, Arkadi Nemirovski, Gary Parker, Dana Randall, Martin Savelsbergh, Joel Sokol, Robin Thomas, Chen Zhou, and many others.

To my fellow graduate students, who studied with me, challenged me and banged heads with me: Doug Altner, Dan Dadush, Faram Engineer, Ricardo Fukasawa, Mike Hewitt, Helder Inacio, Fatma Kılınç, Dimitri Papageorgiou, Claudio Santiago, Sangho Shim, Dan Steffy, Kael Stilp, Steve Tyber, Juan Pablo Vielma, and many others I shouldn't have forgotten.

To ExxonMobil Research & Engineering, especially Kevin Furman, Ahmet Keha and Jin-Hwa Song.

To the following organizations, for their financial support: ARCS Foundation, Georgia Tech Office of the President, Roberto C. Goizueta Foundation, and the National Science Foundation.

To my wife and family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The methodologies of optimization and operations research have been applied since their inception to problems in supply chain management. The transportation, storage and management of goods is crucial to a business' bottom line, and problems inspired by supply chain management, logistics, transportation and inventory control have informed and nurtured research in optimization and operations research.

The discrete nature of decisions related to transportation and production with fixed setups makes these areas amenable to techniques from *mixed-integer linear programming* (MIP). Problems studied in MIP include a combination of discrete (integer) and continuous decision variables appearing in a linear objective function and linear constraints. MIP generalizes the well-known *linear program* (LP), which is similar but includes only continuous variables and is much easier to solve. Two famous problems studied by MIP researchers are the *traveling salesman problem* (TSP) [2] and the production *lot-sizing problem* (LSP) [53]. Both problems are inspired by applications in transportation and production, respectively, and their study has led to MIP-based decision support tools implemented in practice.

Many applications in inventory control include dynamic, periodic or recursive components; problems in this area are often studied in *dynamic programming* (DP). These problems include a set of possible states (e.g. inventory levels) and available actions from each state (e.g. order quantities). Canonical models in this field include the *economic order quantity* (EOQ) model for the continuous-review, single-item LSP, which was studied as far back as 1913 [26, 33].

1.1 Contribution

The main contribution of this thesis is the study and approximate solution of multi-period or infinite-horizon supply chain models using a combination of techniques from MIP and DP. As research has advanced solution techniques for different classical problems, practitioners and decision makers have introduced more complex variations and combinations of known models. In general, a holistic approach accounting simultaneously for different decisions is desirable, and research in recent decades has focused on these complex models. One well-known instance is the *inventory routing problem* (IRP) [16, 22, 49], which combines transportation and inventory decisions into a single multi-period or infinite horizon model.

The IRP is a prime example of the type of problem we are interested in. On one hand, it includes discrete decisions related to the dispatching and routing of a vehicle or fleet of vehicles; such decisions are readily modeled with integer and/or binary decision variables inside a MIP. On the other hand, the IRP also concerns the management of inventory at many points in a supply chain across time; the vector of initial inventories can thus be viewed as the state variable of a large and complex DP.

Our methodology seeks to quickly generate high-quality solutions for these complex supply chain models by decomposing them across time into single- or few-period subproblems, solving these smaller MIP's directly with known discrete optimization techniques, and linking the solution "fragments" together via the inventory state variable. To circumvent the *ending effect* which normally results in *myopic* solutions, we construct an approximate *inventory value function* that we include in the single- or few-period problem's objective. We obtain this value function with an *approximate dynamic programming* (ADP) algorithm that combines sampling, optimization, data fitting and DP techniques. Chapter 2 outlines our algorithmic framework using a generic fixed-charge network flow model, and Chapter 5 gives a computational case

study based on various practical model types.

The construction of the inventory value function within the ADP algorithm relies on data fitting models. Depending on the type of value function considered, the resulting fitting problem may be non-convex and difficult to solve. Chapter 3 outlines heuristic and exact optimization methods appropriate for different fitting scenarios, along with secondary implementation issues related to the function classes. Some of the exact optimization formulations require new MIP and *mixed-integer nonlinear programming* (MINLP) modeling techniques, and constitute a secondary contribution.

In order to efficiently embed the value function into a MIP model of reduced size, our algorithm makes simplifying assumptions about the function's structure. The main simplification is the replacement of superadditive, possibly discontinuous *piecewise linear* (PWL) functions with much simpler concave, continuous PWL functions. Although the structure of value functions for finite MIP models is well-studied, the corresponding questions for infinite-horizon models are not as clear. In Chapter 4, we study the value function of an infinite-horizon version of the canonical uncapacitated, single-item LSP, and compare our algorithm to its optimal policies as a proof of concept.

1.2 Literature Review

Since the introduction of seminal models like the TSP, many transportation problems have been studied using MIP. The current state of the art involves ever larger transportation networks sometimes spanning vast geographic areas and long stretches of time. One popular MIP technique especially suited to large, complex models is *column generation*; recent examples of transportation and IRP research using this technique include [19, 20, 21, 25, 47]. A general introduction to the concept is [24]. Many researchers have also developed MIP-based heuristic techniques for complex transportation problems; some examples are [17, 34, 50, 62, 63].

The supply chain and inventory control problems studied with DP have also vastly increased in size. Many problems of practical interest are immune to traditional DP methodology because of the famous *curse of dimensionality*: As the size and dimension of the problem grows, the time required to solve the problem exactly grows exponentially quickly. ADP is an umbrella term coined in recent years to refer to a variety of techniques that address the curse by approximating the solution in different ways; two recent texts that introduce many ADP concepts are [6, 55]. Several authors have attempted inventory value function approximations for the IRP and related problems, e.g. [1, 16, 40, 41, 48]. Specifically, the use of data fitting or regression and other functional approximation techniques goes at least as far back as 1959 [5]. More recent examples include the following: In [43], the authors use simulation and linear regression to approximately value different types of American options. Trick and Zin [68] use cubic splines to approximate the value function of a DP with a discretized state space; their methodology allows for refinement of the state space discretization to increase the approximate value function's accuracy. Finally, in [69] the authors study "feature-based" algorithms (including linear regression techniques) for finite-state and -action DP's and provide convergence results. Our algorithm generalizes the previously cited works by considering uncountable state and action spaces directly and by replacing classical least-squares linear regression with more general data fitting of continuous PWL concave functions.

Because multi-period or infinite-horizon problems are fundamental to supply chain management, many other methodologies exist beyond what is strictly DP. One important example is the *forecast horizon*, in which researchers determine the shortest multi-period problem whose solution's first period is identical to the infinite problem solution's first period. Variations of LSP are in particular studied under this paradigm; see [4, 23, 27, 31, 65]. Others have also studied the stability or accuracy of solutions generated by finite sub-problems for an infinite-horizon problem via a

rolling horizon, a recursive procedure where the solution to the finite problem is implemented, the model is updated, and a new problem of equal size is generated that reflects the previous solution's changes to the system. One example of such a study for various supply chain models is [39].

Although the mathematics are more complicated and the algorithms more incipient, there are also *infinite optimization* techniques available for infinite-horizon models. Research in this area seeks to characterize optimal solutions, approximate them with solutions to finite problems (e.g. using forecast horizons), or generate them with special algorithms. Specific examples of infinite optimization in supply chain management and related models are [59] (network flows and LSP), [60] (LP and LSP) and [64] (equipment replacement).

Data fitting and regression problems are fundamental and arise in many areas. [14, Chapter 6] surveys convex optimization techniques used in approximation and fitting, while [61] studies statistical techniques relevant to our fitting models. Specific examples of recent work in different fields that consider fitting problems related to PWL functions include [28] (neural networks), [35] (Bayesian regression), [54] (computer graphics and visualization), and [66] (statistical inference). Classification and clustering, which are related to PWL fitting, have previously been studied using mathematical programming in [7, 42, 52]; the first reference employs MIP techniques similar to some of the models studied here. Some of the PWL fitting problems we consider were introduced in [44]; the authors also developed a Gauss-Newton heuristic we adapt here for our own use.

The value function concept and its relation to cutting planes has been studied in MIP since the 1970's [38, 72], and received extensive attention in the work of Blair [9] and Blair and Jeroslow [10, 11, 12, 13]. However, we are not aware of any other work that explicitly links the MIP value function concept to value function approximation in ADP.

The single-item LSP is a seminal model studied as far back as 1958 [45, 71]. Two recent surveys of single-item LSP and related models are [15, 73]. The inventory value function of an infinite-horizon, average-cost, continuous-review, single-item LSP similar to the one we consider in Chapter 4 was studied in [29].

1.3 Technical Preliminaries

1.3.1 Mixed-Integer Programming and Superadditive Functions

A mixed-integer linear program is an optimization problem

$$V(s) = \max \ fx + cz \tag{1.1a}$$

$$\text{s.t. } Ax + Bz \leq s \tag{1.1b}$$

$$x \in \mathbb{Z}^m_+, \quad z \in \mathbb{R}^n_+, \tag{1.1c}$$

where $f \in \mathbb{R}^m, c \in \mathbb{R}^n, s \in \mathbb{R}^k, A \in \mathbb{R}^{k\times m}, B \in \mathbb{R}^{k\times n}$, and fx represents the inner product. (Here and elsewhere, we assume all models use rational numbers.) There is a rich theory and an abundance of applications for MIP models; [51] is a thorough treatment.

The explicit parametrization of (1.1) based on the right-hand side vector s is of particular interest to us. The optimal value of the problem as a function of the right-hand side vector, $V(s)$, is known as the MIP's *value function*. The following theorem summarizes well-known results. A function $V : S \subseteq \mathbb{R}^k \to \mathbb{R} \cup \{\pm\infty\}$ is *superadditive* if $V(s^1) + V(s^2) \leq V(s^1 + s^2), \forall\ s^1, s^2 \in S$ with $s^1 + s^2 \in S$. A *subadditive* function satisfies the reverse inequality.

Theorem 1.1 ([51])**.** *Let V be the value function of a MIP with maximization objective. Then V is PWL, superadditive, upper semi-continuous, and satisfies $V(0) \in \{0, \infty\}$. If $V(0) = 0$, then $V(s) < \infty, \forall\ s \in S$.*

For a MIP with minimization objective, the theorem's analogue states that the value function is subadditive and lower semi-continuous. Note that restrictions of

V are not guaranteed to be sub- or superadditive (unlike, say, convex and concave functions). A main part of this thesis' approach is the approximation of superadditive functions with concave ones. The next result motivates this approach.

Proposition 1.2 ([58]). *Let $V : \mathbb{R}^k \to \mathbb{R} \cup \{-\infty\}$ be superadditive and positively homogeneous. Then V is concave.*

A function V is *positively homogeneous* if $V(\lambda s) = \lambda V(s), \forall\ \lambda \geq 0$. In particular, a positively homogeneous function V has $V(0) = 0$. In addition, although MIP value functions are not necessarily positively homogeneous, their LP counterparts are. Rockafellar's text [58] is a standard reference for convex analysis and in particular studies the relation between subadditive (superadditive) and convex (concave) functions.

1.3.2 Dynamic Programming and Value Iteration

Let S, X be finite sets. Let $r : S \times X \to \mathbb{R} \cup \{-\infty\}$ be a *reward* function, let $f : S \times X \to S$ be a *transition* function, and let $\gamma \in [0, 1)$ be a discount factor. A discrete, deterministic, infinite-horizon, discounted dynamic program is given by

$$V(s) = \max\Bigg\{\sum_{t=1}^{\infty} \gamma^{t-1} r(s_{t-1}, x_t) : \\ s_t = f(s_{t-1}, x_t), x_t \in X, \forall\ t \geq 1; s_0 = s\Bigg\}, \forall\ s \in S. \tag{1.2}$$

The function $V \in \mathbb{R}^S$ is known as the DP's *value function*; in minimization contexts it is also called the *cost-to-go* function. Stochastic variants in which f is replaced with a set of probability distributions are known as *Markov decision processes* (MDP); a modern reference for DP and MDP is [56]. The following results summarize DP theory relevant to our work.

Theorem 1.3 ([56]). *V is the unique solution to the set of* Bellman equations*:*

$$V(s) = \max_{x \in X} \{r(s, x) + \gamma V(f(s, x))\}, \forall\ s \in S. \tag{1.3}$$

The *Bellman operator* $\mathcal{B} : \mathbb{R}^S \to \mathbb{R}^S$, closely related to the previous theorem, is given by

$$\mathcal{B}(V'(s)) = \max_{x \in X} \{r(s,x) + \gamma V'(f(s,x))\}, \forall\ s \in S, \forall\ V' \in \mathbb{R}^S. \tag{1.4}$$

In the equation above, any maximizing $x \in X$ is *greedy* with respect to V'.

Theorem 1.4 ([56])**.** *Let $V_0 \in \mathbb{R}^S$ be any vector, and let $V_n = \mathcal{B}(V_{n-1}), \forall\ n \in \mathbb{N}$. Then $\lim_{n\to\infty} V_n \to V$; moreover,*

$$\|V_n - V\|_\infty \leq \gamma^n \|V_0 - V\|_\infty,$$

where $\|\cdot\|_\infty$ is the maximum norm.

The algorithm that constructs the sequence V_n is known as *value iteration*.

CHAPTER II

APPROXIMATE INVENTORY VALUATION

This chapter introduces the algorithmic framework we use to construct approximate inventory value functions in multi-period or infinite-horizon supply chain models. To maintain an appropriate level of generality, we use a generic *fixed-charge network flow* (FCNF) model as a stand-in for any supply chain model of interest. The FCNF structure underlies virtually all transportation or logistics models, particularly those with fixed costs representing routing or production setup choices, and FCNF's are among the most difficult and thoroughly studied problems in MIP and discrete optimization [51].

We study the model with a combined MIP and ADP approach that incorporates the approximate value function into a MIP framework. By combining the techniques from mixed integer programming and approximate dynamic programming, we hope to circumvent difficulties typically encountered when solving multi-period models: The inventory value function allows us to shorten planning horizons, thus yielding a model size tractable to MIP methodology.

Within this context, the choice of a value function must balance two competing interests. The function must be simple enough to fit within a MIP, yet complex enough to capture the influence inventory has on the model. We have chosen to use PWL concave functions: From a modeling perspective, these functions can be implemented in a MIP with only a small number of auxiliary continuous variables, and therefore do not alter the model's difficulty. From a practical perspective, concavity captures the diminishing marginal value one expects in a model of this type, where, for instance, inventory at a consumer is more valuable the closer the consumer is to a stock-out.

Finally, from a theoretical perspective, piecewise linear concave functions are the "closest" continuous functions to MIP value functions, which are piecewise linear, superadditive and possibly discontinuous [9, 10, 11]. We must note, however, that the previous references study finite MIP's, while our current study extends to infinite-horizon models. Although value functions for infinite-horizon models are extensively studied in DP [56] as part of classical algorithms such as *value iteration*, the structure of infinite MIP's is less well understood. We revisit this question for a specific LSP model in Chapter 4.

Approximate piecewise linear concave value functions have already been studied for several resource allocation problems [55, Chapter 12] where the single-period model can usually be formulated as an LP. When the model is solved, the optimal dual solution is used as a proxy for a resource's value. After each solve, the dual solution is used to refine the approximate value function via an algorithm that preserves the function's concavity [67].

2.1 Generic Model and Examples

Let $G = (S \cup C, A)$ be a directed network, where the node set is composed of two finite, disjoint sets S and C that represent *suppliers* and *consumers*. Each supplier $i \in S$ has constant inventory bounds $[0, d_i]$ (with $d_i > 0$) and a constant production rate $0 < w_i \leq d_i$. Similarly, each consumer $j \in C$ has constant inventory bounds $[0, d_j]$ and a constant demand rate $0 < w_j \leq d_j$. Each node $i \in S \cup C$ has a starting inventory $s_i^0 \in [0, d_i]$.

The fixed cost of sending product on arc $a \in A$ is $f_a > 0$, and the variable, per-unit cost of sending product over the arc is $c_a > 0$. The total flow capacity over a period on arc $a \in A$ is $\kappa_a > 0$. The per-unit reward of picking up product from supplier $i \in S$ is r_i, and the per-unit reward of delivering product to consumer $j \in C$ is r_j. The holding cost per unit per period for inventory at node $i \in S \cup C$ is h_i. The per

period discount factor representing temporal preference is $\gamma \in [0, 1)$.

Let $\mathcal{T} = \{1, \ldots, T\}$ be the set of periods in the planning horizon, where $T \in \mathbb{N} \cup \{\infty\}$. Let $x_a^t \in \{0, 1\}$ indicate whether product flows on arc $a \in A$ during period $t \in \mathcal{T}$, and let z_a^t indicate the amount of product flow on the arc during the period. Let q_i^t denote the amount of product picked up from $i \in S$ during period $t \in \mathcal{T}$ and let q_j^t denote the amount of product delivered to consumer $j \in C$ during period $t \in \mathcal{T}$. Let $s_i^t \in [0, d_i]$ indicate the inventory amount at node $i \in S \cup C$ at the end of period $t \in \mathcal{T}$. We use the notation $\delta^+(i) = \{a \in A : i \text{ is the tail of } a\}$ and $\delta^-(i) = \{a \in A : i \text{ is the head of } a\}$.

The multi-period or infinite-horizon FCNF model is given by

$$\max \sum_{t \in \mathcal{T}} \gamma^{t-1} \left(\sum_{i \in S \cup C} (r_i q_i^t - h_i s_i^t) - \sum_{a \in A} (f_a x_a^t + c_a z_a^t) \right) \tag{2.1a}$$

$$\text{s.t. } s_i^t = s_i^{t-1} + w_i - q_i^t, \forall\ i \in S, t \in \mathcal{T} \tag{2.1b}$$

$$s_j^t = s_j^{t-1} - w_j + q_j^t, \forall\ j \in C, t \in \mathcal{T} \tag{2.1c}$$

$$\sum_{a \in \delta^+(i)} z_a^t - \sum_{a \in \delta^-(i)} z_a^t = q_i^t, \forall\ i \in S, t \in \mathcal{T} \tag{2.1d}$$

$$\sum_{a \in \delta^-(j)} z_a^t - \sum_{a \in \delta^+(j)} z_a^t = q_j^t, \forall\ j \in C, t \in \mathcal{T} \tag{2.1e}$$

$$z_a^t \leq \kappa_a x_a^t, \forall\ a \in A, t \in \mathcal{T} \tag{2.1f}$$

$$x_a^t \in \{0, 1\}, \forall\ a \in A, t \in \mathcal{T} \tag{2.1g}$$

$$z_a^t \geq 0, \forall\ a \in A, t \in \mathcal{T} \tag{2.1h}$$

$$q_i^t \geq 0, \forall\ i \in S \cup C, t \in \mathcal{T} \tag{2.1i}$$

$$s_i^t \in [0, d_i], \forall\ i \in S \cup C, t \in \mathcal{T}. \tag{2.1j}$$

In this model, the objective (2.1a) maximizes profit over the planning horizon; if $T = \infty$, we take the set of feasible solutions to be a subset of the sequences with well-defined and finite objective (cf. [60]). The supplier and consumer inventory balance constraints (2.1b) and (2.1c) track inventory levels from one period to the next. The

product balance constraints (2.1d) and (2.1e) track the amount of product flowing over the arcs and relate it to the pickup and delivery variables. The variable upper bound constraint (2.1f) ensures that product flow remains within arc capacity and is positive only if the arc's fixed cost is paid. Constraints (2.1g)–(2.1j) establish the decision variables' domain.

To simplify our subsequent analysis, we assume that the single-period model ((2.1) with $T = 1$) is feasible for any starting inventory vector $s^0 \in [0, d]$. (If the assumption does not hold, the network can be modified in one of several standard ways to create a new network that does satisfy the assumption.)

Example 2.1 (Single-Item Uncapacitated LSP)**.** One of the simplest examples of fixed-charge structure in a supply chain model is the single-item LSP. Suppose we need to manage the production schedule for a single item that experiences constant per-period demand $w > 0$. There is no production or inventory capacity, and all demand must be met each period, either with items produced that period, or items in inventory. Every period we produce, we incur a fixed cost $f > 0$ and a variable cost of $c > 0$ per unit produced. Items left over at the end of the period after demand is met incur a holding cost of $h > 0$ per unit. We can model this problem as

$$\min \sum_{t \in \mathcal{T}} \gamma^{t-1}(f x_t + c z_t + h s_t) \tag{2.2a}$$

$$\text{s.t. } z_t + s_{t-1} - s_t = w, \forall\ t \in \mathcal{T} \tag{2.2b}$$

$$M x_t - z_t \geq 0, \forall\ t \in \mathcal{T} \tag{2.2c}$$

$$x_t \in \{0, 1\}; z_t, s_t \geq 0, \forall\ t \in \mathcal{T}, \tag{2.2d}$$

where s_0 is the initial stock and $M > 0$ can be chosen large enough to guarantee optimality. Note that we adopt the minimization notation, since the model includes only costs. We study this problem and its value function in Chapter 4.

Example 2.2 (Simple-Time Maritime IRP)**.** Suppose S and C are sets of ports located far from each other, so that the distances between two ports in the same set

are much smaller than distances between two ports of different sets. Homogeneous capacitated vessels are available to begin a voyage from any supplier, can visit as many suppliers as necessary, and then travel to visit as many consumers as necessary, after which the voyage ends (with no return.) At most one vessel may begin a voyage in any period, and a vessel's voyage begins and ends in the same period. This last assumption is not as restrictive as it sounds, because if the travel time between S and C is assumed constant (say τ periods), then any voyage that begins on the supplier side in period t ends on the consumer side in period $t+\tau$, and therefore the inventories on the supplier side for period t may be identified with consumer inventories in period $t + \tau$.

As in the FCNF model, each supplier $i \in S$ has constant inventory bounds $[0, d_i]$ and a constant per period production rate $0 < w_i \leq d_i$, and each consumer $j \in C$ has constant inventory bounds $[0, d_j]$ and a constant per period consumption rate $0 < w_j \leq d_j$. Each supplier or consumer $i \in S \cup C$ has a starting inventory $s_i^0 \in [0, d_i]$. The fixed transportation costs between any two points $i, j \in S \cup C$ are $f_{ij} \geq 0$, and the per unit revenue obtained from delivering product from supplier $i \in S$ to consumer $j \in C$ is r_{ij}. Since vessels are homogeneous, we normalize product measurements so that vessels have unit capacity. There are no per-unit transportation and no inventory holding costs.

We model each period's transportation decision with a network $G = (N, A)$, where $N = S \cup C$ and $A = S^2 \cup (S \times C) \cup C^2$. For period t and $a \in A$, let $x_a^t \in \{0, 1\}$ denote whether a vessel travels on arc a. Let $y_i^t \in \{0, 1\}$ for $i \in S$ denote the indicator variable that is equal to one if and only if a voyage starts from i in period t, and let y_j^t be defined analogously for $j \in C$. Let q_{ij}^t denote the amount of product from supplier $i \in S$ delivered to consumer $j \in C$ in period t, and let z_a^t denote the amount of product in the vessel when it travels over arc $a \in A$ in period t. (Note that the definition of z does not differentiate by product.) Let $s_i^t \in [0, d_i]$ denote the inventory

at $i \in S \cup C$ at the end of period t.

The maritime IRP is modeled as

$$\max \sum_{t\in\mathcal{T}} \gamma^{t-1}\Bigg(\sum_{i\in S, j\in C} r_{ij}q_{ij}^t - \sum_{a\in A} f_a x_a^t\Bigg) \tag{2.3a}$$

$$\text{s.t.} \sum_{a\in\delta^+(i)} x_a^t - \sum_{a\in\delta^-(i)} x_a^t = y_i^t, \forall\ i \in S, t \in \mathcal{T} \tag{2.3b}$$

$$\sum_{a\in\delta^-(j)} x_a^t - \sum_{a\in\delta^+(j)} x_a^t = y_j^t, \forall\ j \in C, t \in \mathcal{T} \tag{2.3c}$$

$$\sum_{i\in S} y_i^t \leq 1, \forall\ t \in \mathcal{T} \tag{2.3d}$$

$$s_i^t = s_i^{t-1} + w_i - \sum_{j\in C} q_{ij}^t, \forall\ i \in S, t \in \mathcal{T} \tag{2.3e}$$

$$s_j^t = s_j^{t-1} - w_j + \sum_{i\in S} q_{ij}^t, \forall\ j \in C, t \in \mathcal{T} \tag{2.3f}$$

$$z_a^t \leq x_a^t, \forall\ a \in A, t \in \mathcal{T} \tag{2.3g}$$

$$\sum_{a\in\delta^+(i)} z_a^t - \sum_{a\in\delta^-(i)} z_a^t = \sum_{j\in C} q_{ij}^t, \forall\ i \in S, t \in \mathcal{T} \tag{2.3h}$$

$$\sum_{a\in\delta^-(j)} z_a^t - \sum_{a\in\delta^+(j)} z_a^t = \sum_{i\in S} q_{ij}^t, \forall\ j \in C, t \in \mathcal{T} \tag{2.3i}$$

$$y_i^t \in \{0,1\}, \forall\ i \in S \cup C, t \in \mathcal{T} \tag{2.3j}$$

$$x_a^t \in \{0,1\}, \forall\ a \in A, t \in \mathcal{T} \tag{2.3k}$$

$$q_{ij}^t \geq 0, \forall\ i \in S, j \in C, t \in \mathcal{T} \tag{2.3l}$$

$$z_a^t \geq 0, \forall\ a \in A, t \in \mathcal{T} \tag{2.3m}$$

$$s_i^t \in [0, d_i], \forall\ i \in S \cup C, t \in \mathcal{T}. \tag{2.3n}$$

The objective (2.3a) maximizes net profit over the planning horizon. The vessel flow balance constraints (2.3b) and (2.3c) require that a vessel exits a supplier or a consumer if it visits it. Constraint (2.3d) ensures that no more than one vessel starts a voyage every period. The supplier and consumer inventory balance constraints (2.3e) and (2.3f) track inventory levels from one period to the next. The variable upper

bound constraint (2.3g) ensures that product flows on an arc only if a vessel travels on it. The continuous product balance constraints (2.3h) and (2.3i) track the amount of product on a vessel and relate it to the delivery variables. (Because all pickups occur before any delivery, we can use the q_{ij}^t variables in both constraint classes and do not need to differentiate flow by product.) Note that the variable domain constraints (2.3j)–(2.3n) along with the variable upper bounds (2.3g) and product balances (2.3h) and (2.3i) ensure that the total amount delivered and the total on the vessel over any arc do not exceed the vessel's unit capacity.

The formulation detailed above does not explicitly include subtour elimination constraints of the form

$$\sum_{a \subseteq R^2} x_a^t \leq |R| - 1, \forall\ R \subseteq S \text{ or } R \subseteq C.$$

However, since $f \geq 0$, the product balance constraints (2.3h) and (2.3i) prevent any subtour from appearing in an optimal solution.

For T large enough, the model is infeasible if $\sum_{i \in S} w_i \neq \sum_{j \in C} w_j$. However, this requirement can be relaxed by adding a third-party supplier i_0 and a third-party consumer j_0, and modifying the model accordingly. One approach is to set $w_{i_0} = w_{j_0} = 0$, $d_{i_0} = d_{j_0} = \infty$, $h_{j_0}^0 = 0$ and $h_{i_0}^0 = M$, for some large enough $M > 0$. These parameters effectively mean that supply and consumption at the third-party points is unlimited, and delivery to and from these points can be treated as a slack in the model. Even with a third-party supplier and consumer, however, the model still requires $\sum_{i \in S} w_i \leq 1$ and $\sum_{j \in C} w_j \leq 1$; that is, the total supply per period and the total consumption per period must not exceed the vessel's capacity.

2.2 An Approximate Dynamic Programming Algorithm

Solution times for models such as (2.1) tend to grow exponentially as T increases. Moreover, in most practical settings only the decisions related to the first period (or first few periods) are immediately necessary; once these decisions are implemented,

the model can be updated to reflect them and the procedure can repeat recursively in a rolling horizon framework. However, if we solve (2.1) "as is" with a small T (say $T = 1$), the resulting solution is likely to be myopic and far from optimal in terms of a long or infinite horizon. To counteract this behavior, we can include a *value function* $V : [0, d] \to \mathbb{R}$ in the objective. Specifically, for (2.1) with finite T, we replace the objective (2.1a) with

$$\max \sum_{t \in \mathcal{T}} \gamma^{t-1} \left(\sum_{i \in S \cup C} (r_i q_i^t - h_i s_i^t) - \sum_{a \in A} (f_a x_a^t + c_a z_a^t) \right) + \gamma^T V(s^T). \tag{2.4}$$

For any V, define $P(s, V)$ as problem (2.1) with $T = 1$, $s^0 = s$ and (2.4) replacing (2.1a). Define $X(s)$, $\nu(s, V) \in \mathbb{R}$ and $\sigma(s, V) \in [0, d]$ respectively as the feasible region, optimal value and optimal ending inventory of $P(s, V)$; note that $X(s)$ is independent of V.

In discounted models, the actual value function V^* by definition satisfies the equation

$$V^*(s) = \max_{(x,z,q,s') \in X(s)} \sum_{i \in S \cup C} (r_i q_i - h_i s_i') - \sum_{a \in A} (f_a x_a + c_a z_a) + \gamma V^*(s'), \tag{2.5}$$

for any $s \in [0, d]$. Of course, if we knew V^* and could solve $P(s, V^*)$, we would have the optimal first-period decision. However, finding V^* essentially entails solving (2.1). There is also the additional difficulty of representing a function such as V^* inside of a MIP objective in order to solve $P(s, V^*)$.

We propose instead to compute a function that approximates V^* but can be easily represented within a MIP objective. Specifically, given a set $\mathcal{V}$ of PWL concave functions, we would like to compute a function $\tilde{V} \in \mathcal{V}$ that approximates V^*. In Chapter 3, we propose different function sets of interest and the details of implementation for each set; for now, assume a generic $\mathcal{V}$.

Algorithm 1 details the procedure for computing an approximation of V^*. Intuitively, it successively refines the incumbent value function V_n by sampling initial random inventory vectors s, observing their optimal value in the problem $P(s, V_n)$,

Algorithm 1 Fitting-Based Value Iteration

Require: $V_0 \in \mathcal{V}; K, L, N \in \mathbb{N}; \alpha_n \in (0, 1], n = 1, \ldots, N$
  for $n = 1, \ldots, N$ **do**
    for $k = 1, \ldots, K$ **do**
      $s^{n,k,1} \leftarrow \text{sample}([0, d])$
      for $\ell = 1, \ldots, L$ **do**
        $\nu^{n,k,\ell} \leftarrow \nu(s^{n,k,\ell}, V_{n-1})$
        $s^{n,k,\ell+1} \leftarrow \sigma(s^{n,k,\ell}, V_{n-1})$
      end for
    end for
    $\bar{V} \leftarrow \text{fit}(s^n, \nu^n)$
    $V_n \leftarrow \text{merge}(V_{n-1}, \bar{V}, \alpha_n)$
  end for
  return V_N

and using these observations to update V_n. It requires an initial value function V_0, which can be chosen based on prior knowledge or set to some default appropriate for $\mathcal{V}$. The parameters N, K, and L respectively denote the number of algorithm iterations, the number of inventory vector samples per iteration, and the length of each *sample path*, i.e. the number of solves starting from each sample. Although we don't consider it here, N can be replaced by an appropriate convergence test, where the algorithm runs until the change between successive iterates is small enough. The *step size* α_n is a weighted average that indicates how the current incumbent V_n and the best-fit function $\bar{V}$ are merged. The algorithm's three major tasks are explained in further detail below.

2.2.1 Sampling and observing

Unlike classical value iteration, since the set of feasible inventory vectors is uncountably infinite, we cannot hope to observe V_n at every point in $[0, d]$. Instead, we sample the set a number of times and use these samples as a proxy for the entire space. The sample function represents any generic sampling procedure that generates a random point from the set $[0, d]$; in our experiments we use a uniform distribution, but this

may be substituted for any other distribution that more accurately fits a particular application. (This may in fact be necessary in stochastic versions of (2.1) [55].) Another popular technique is to avoid sampling altogether and choose a set of representative inventory vectors which the algorithm uses at each iteration [69].

2.2.2 Fitting

After collecting a set of observations (s^n, ν^n), the next step is to find the function $\bar{V} \in \mathcal{V}$ that best fits this data according to some measure. The fit function solves an optimization problem of the form

$$\min_{V \in \mathcal{V}} \sum_{k=1}^{K} \sum_{\ell=1}^{L} |V(s^{n,k,\ell}) - \nu^{n,k,\ell}|^q, \tag{2.6}$$

where $q \in \mathbb{N}$. Depending on $\mathcal{V}$, (2.6) may be convex or non-convex and can be solved exactly with an optimization algorithm or approximately with a heuristic. Chapter 3 details solution procedures for various types of sets $\mathcal{V}$.

2.2.3 Merging

The merge function updates the incumbent value function V_n based on the best-fit obtained from the current iteration's observation. Formally, we have merge : $\mathcal{V} \times \mathcal{V} \times [0,1] \to \mathcal{V}$, where the following conditions are satisfied:

$$\text{merge}(V, V', 0) = V, \quad \forall\ V, V' \in \mathcal{V} \tag{2.7a}$$

$$\text{merge}(V, V', 1) = V', \quad \forall\ V, V' \in \mathcal{V} \tag{2.7b}$$

$$\text{merge}(V, V, \alpha) = V, \quad \forall\ V \in \mathcal{V}, \alpha \in [0,1]. \tag{2.7c}$$

In most cases, $\mathcal{V}$ can be represented as a polyhedral cone and the merge function is the convex combination of two elements in the set, with weights α and $(1-\alpha)$ respectively. We detail one exception of interest in Chapter 3.

The step size rule α_n indicates the weight given to the best-fit and incumbent functions at each iteration. In principle, we could simply replace each incumbent with

the new best-fit function at each iteration (equivalent to setting $\alpha_n = 1, \forall\ n$), but in practice this may result in numerically unstable behavior. Even in simple settings, theoretical convergence is not guaranteed unless $\sum_{n=1}^{\infty} \alpha_n = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$ [55], but these *harmonic* step sizes can lead to stalling behavior in practice, where the algorithm appears to converge because the step size becomes too small. In our experiments we have chosen to compromise the two extremes by using McClain's rule [46]:

$$\alpha_n = \begin{cases} 1, & n = 1 \\ \frac{\alpha_{n-1}}{1+\alpha_{n-1}-\beta}, & n \geq 2 \end{cases} \tag{2.8}$$

where $0 < \beta \ll 1$ is a parameter. Initially, the rule behaves like a harmonic series, giving the first few iterations large weight. However, as n grows the rule converges to $\beta > 0$, and thus in the long term later iterations outweigh early ones.

CHAPTER III

VALUE FUNCTION FITTING

In Chapter 2 we posed an infinite-horizon FCNF model and an ADP algorithm used to compute an approximate inventory value function for the model. In particular, the algorithm considers a set of PWL concave functions and chooses a function from the set. This chapter considers specific examples of value function sets, and explains the implementation details for each one, including modeling, fitting and, in one case, merging. We should note that the modeling of PWL concave functions is folklore in LP, and all modeling issues presented here are well-known in the optimization community. [70] is a recent study of more complex PWL modeling in MIP; [14, Chapter 6] is an extensive survey of convex optimization models for approximation and fitting.

The second major task of the algorithm, the fitting problem, is the main implementation question and this chapter's principal contribution. Given an inventory domain $[0, d] \in \mathbb{R}^m$, a set of PWL concave functions $\mathcal{V}$ with $V : [0, d] \rightarrow \mathbb{R}, \forall\ V \in \mathcal{V}$ and a finite set of *data points* $(s^k, \nu^k) \in [0, d] \times \mathbb{R}, \forall\ k = 1, \ldots, K$, we must solve

$$\min_{V \in \mathcal{V}} \sum_{k=1}^{K} |V(s^k) - \nu^k|^q, \tag{3.1}$$

where $q \geq 1$; in practical applications we usually have $q \in \{1, 2\}$. The minimax case ($q = \infty$, also known as the *Chebyshev* norm) is also of interest, and obtained simply by replacing the summation with a max over all absolute differences.

3.1 Separable Functions over a Fixed Grid

Let $p \in \mathbb{N}$, and suppose we have an a priori partition $0 = b_i^0 < b_i^1 < \cdots < b_i^p = d_i$ of each inventory dimension $i = 1, \ldots, m$. (We assume for simplicity of exposition that

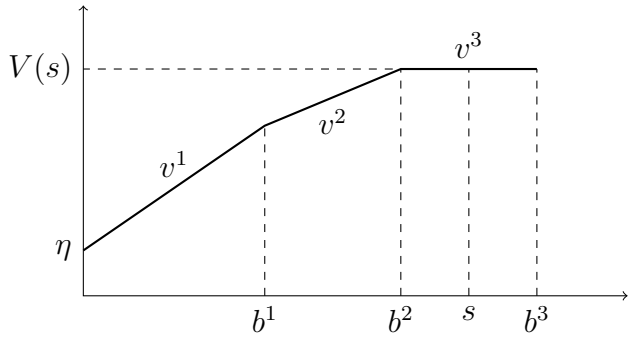

Figure 1: Univariate PWL concave function over fixed bucket lengths.

all dimensions are partitioned into p buckets, although this is not necessary.) We can define a separable PWL concave function as

$$V(s) = \eta + \sum_{i=1}^{m} v_i^j (\min\{s_i - b_i^{j-1}, b_i^j - b_i^{j-1}\})_+, \tag{3.2}$$

where $\eta \in \mathbb{R}$, $v_i^0 \geq \cdots \geq v_i^p, \forall\ i$ and $(a)_+ = \max\{a, 0\}$. The set of all separable PWL concave functions over grid b is then parametrized by

$$\mathcal{V}_{\text{SF}} = \{(v, \eta) \in \mathbb{R}^{m \times p} \times \mathbb{R} : v_i^0 \geq \cdots \geq v_i^p, \forall\ i = 1, \ldots, m\}. \tag{3.3}$$

Figure 1 shows a univariate example.

Example 3.1 (Example 2.2 continued)**.** Consider again the maritime IRP model introduced in Chapter 2. If we can assume that each port's inventory value is independent of the other ports, a separable value function is an appropriate choice. An intuitive partition of the inventory domain would then be $p = 2$, and $b_i^1 = d_i - w_i, \forall\ i \in S$, $b_j^1 = w_j, \forall\ j \in C$. In words, this partition assumes the value of each consumer's inventory is higher when inventory is below one period's consumption; i.e. a stock-out is imminent. Similarly, each supplier's inventory value is less when inventory is within one period's production of capacity, and a pickup must occur.

To model any $V = (v, \eta) \in \mathcal{V}_{\text{SF}}$, we define auxiliary variables $s_i^j, i = 1, \ldots, m, j =$

$1, \ldots, p$ and solve

$$V(s) = \eta + \max \sum_{i=1}^{m} \sum_{j=1}^{p} v_i^j s_i^j \tag{3.4a}$$

$$\text{s.t.} \sum_{j=1}^{p} s_i^j = s_i, \forall\ i = 1, \ldots, m \tag{3.4b}$$

$$0 \leq s_i^j \leq b_i^j - b_i^{j-1}. \tag{3.4c}$$

Because the slopes v are monotonically non-increasing in each dimension i, the optimal solution always increases the auxiliary variables in order from s_i^1 to s_i^p until their sum equals s_i, thus yielding (3.2). In the literature, this model is sometimes called the *incremental model* [70].

To fit a function from $\mathcal{V}_{\text{SF}}$ to the data points $\{(s^k, \nu^k)\}_{k=1}^K$, (3.1) becomes

$$\min \sum_{k=1}^{K} \left| \eta + \sum_{i=1}^{m} \sum_{j=1}^{p} v_i^j (\min\{s_i^k - b_i^{j-1}, b_i^j - b_i^{j-1}\})_+ - \nu^k \right|^q \tag{3.5a}$$

$$\text{s.t. } v_i^1 \geq \cdots \geq v_i^p, \forall\ i = 1, \ldots, m \tag{3.5b}$$

$$v \in \mathbb{R}^{m \times p}, \eta \in \mathbb{R}. \tag{3.5c}$$

Because the s^k are given, the quantity $(\min\{s_i^k - b_i^{j-1}, b_i^j - b_i^{j-1}\})_+$ can be precomputed and therefore (3.5) is convex; in particular, for $q \in \{1, 2\}$ it becomes respectively an LP or a *convex quadratic program* (QP). Both problems are efficiently solvable by commercial optimization code. The problem is also equivalent to constrained univariate linear spline regression, which is extensively studied in statistics [61].

When merging two functions $(v, \eta), (v', \eta') \in \mathcal{V}_{\text{SF}}$, we can equivalently consider them as two points in the polyhedral cone given by (3.3). By convexity, a weighted average of the two in the usual sense is also a member of V_{SF}:

$$\text{merge}((v, \eta), (v', \eta'), \alpha) = (1 - \alpha)(v, \eta) + \alpha(v', \eta'),$$

where the sum is considered component-wise. A similar parametrization applies to all function classes we consider except for the non-separable functions over variable

regions in Section 3.3; the convex combination serves as merge function in all cases but the latter.

3.2 Non-Separable Functions over Fixed Regions

We now suppose the inventory domain $[0, d]$ is partitioned into a finite set of polytopes $\mathcal{P}$ satisfying

i) each $P \in \mathcal{P}$ is full-dimensional,

ii) $\bigcup_{P\in\mathcal{P}} P = [0, d]$,

iii) $\text{int}(P) \cap \text{int}(Q) = \varnothing$, for every distinct $P, Q \in \mathcal{P}$,

iv) $P \cap Q$ is a face of P and Q for every $P, Q \in \mathcal{P}$.

We define a PWL concave function over this partition as

$$V(s) = \min_{P\in\mathcal{P}}\{v^P s + \eta_P\}. \tag{3.6}$$

In this definition, $(v^P, \eta_P) \in \mathbb{R}^m \times \mathbb{R}$, vs is the inner product of $v, s \in \mathbb{R}^m$, and the following condition is satisfied:

$$v^P s + \eta_P \leq v^Q s + \eta_Q, \forall\ Q \in \mathcal{P} \setminus \{P\}, s \in P, \forall\ P \in \mathcal{P}. \tag{3.7}$$

Figure 2 shows a two-dimensional example. The set of all non-separable PWL concave functions over the partition is then clearly $\mathcal{V}_{\text{NF}} = \{(v, \eta) \in (\mathbb{R}^m \times \mathbb{R})^{\mathcal{P}} : (3.7)\}$.

To model any $V = (v, \eta) \in \mathcal{V}_{\text{NF}}$ inside a maximization LP or MIP, we define an auxiliary variable ν and solve

$$V(s) = \max\ \nu \tag{3.8a}$$

$$\text{s.t. } \nu \leq v^P s + \eta_P, \forall\ P \in \mathcal{P} \tag{3.8b}$$

$$\nu \in \mathbb{R}. \tag{3.8c}$$

To fit these functions, we must first give an alternate characterization of $\mathcal{V}_{\text{NF}}$, for which we use the following result.

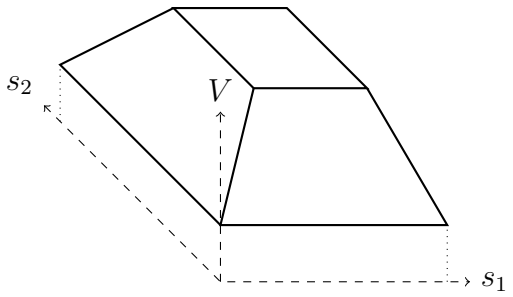

Figure 2: Bivariate non-separable PWL concave function.

Proposition 3.1 ([18])**.** *Any $(v, \eta) \in (\mathbb{R}^m \times \mathbb{R})^{\mathcal{P}}$ is concave if and only if it is continuous and its restriction to any two polytopes from $\mathcal{P}$ that share a facet is concave.*

Let $\Phi(P)$ be the set of vertices of P, let $\Phi(\mathcal{P}) = \bigcup_{P\in\mathcal{P}} \Phi(P)$, and for each $\varphi \in \Phi(\mathcal{P})$, let $\mathcal{P}_\varphi \subseteq \mathcal{P}$ be the set of polytopes that contain φ. Let $\Psi(P)$ be the set of facets of $P \in \mathcal{P}$ that are also facets of another polytope in $\mathcal{P}$, and let $\Psi(\mathcal{P}) = \bigcup_{P\in\mathcal{P}} \Psi(P)$. For any facet $\psi \in \Psi(\mathcal{P})$, let $P_\psi, Q_\psi \in \mathcal{P}$ be the unique pair of polytopes that satisfies $P_\psi \cap Q_\psi = \psi$. Choose $\varrho^\psi \in \psi$ and $\delta^\psi \in \mathbb{R}^m \setminus \{0\}$ to satisfy $\varrho^\psi + \delta^\psi \in P_\psi$ and $r^\psi - \delta^\psi \in Q_\psi$.

Corollary 3.2. *The set $\mathcal{V}_{NF}$ can alternately be characterized by the following two sets of inequalities:*

$$v^P\varphi + \eta_P = v^Q\varphi + \eta_Q, \forall\ P, Q \in \mathcal{P}_\varphi, \forall\ \varphi \in \Phi(\mathcal{P}) \tag{3.9a}$$

$$\frac{1}{2}(v^{P_\psi}(\varrho^\psi + \delta^\psi) + \eta_{P_\psi} + v^{Q_\psi}(\varrho^\psi - \delta^\psi) + \eta_{Q_\psi}) \le v^{P_\psi} r^\psi + \eta_{P_\psi}, \forall\ \psi \in \Psi(\mathcal{P}). \tag{3.9b}$$

$\mathcal{V}_{NF}$ is therefore a polyhedral cone.

Proof. The first set of inequalities enforces continuity at the vertices $\varphi \in \Phi(\mathcal{P})$, which implies continuity everywhere. The second set of inequalities imposes midpoint concavity along each triple $\varrho^\psi, \varrho^\psi \pm \delta^\psi$; because all functions (v^P, η_P) are affine, this translates into concavity across $\psi \in \Psi$, which gives concavity everywhere by Proposition 3.1. □

For each data point s^k, let $P_k \in \mathcal{P}$ be a polytope that contains s^k. The fitting formulation (3.1) becomes

$$\min \sum_{k=1}^{K} |v^{P_k} s^k + \eta_{P_k} - \nu^k|^q \tag{3.10a}$$

$$\text{s.t. } (3.9) \tag{3.10b}$$

$$(v, \eta) \in (\mathbb{R}^m \times \mathbb{R})^{\mathcal{P}}. \tag{3.10c}$$

For $q \in \{1, 2\}$, this problem is again an LP or QP, respectively. However, unlike in the separable case, the number of parameters (i.e. vertices and facets) to compute and hence the number of constraints may be significant.

The increase in computational effort is not the only issue we encounter when switching from separable to non-separable functions. Suppose $|\mathcal{P}| = p$; then the number of parameters defining a function in $\mathcal{V}_{\text{NF}}$, $\Theta(mp)$, is of the same order of magnitude as for $\mathcal{V}_{\text{SF}}$. However, in the separable case these parameters yield p^m hypercubes in the grid, whereas in the non-separable case we only get p polytopes. This drastic difference in the granularity of the partitions indicates that a non-separable function may only be appropriate in low dimensions.

As in the separable case, because the affine functions defining each $V \in \mathcal{V}_{\text{NF}}$ are indexed by region (polytope), the merging operation is a simple convex combination. However, this no longer applies when the regions defined by V are allowed to change.

3.3 Non-Separable Functions over Variable Regions

The obvious extension of the set of non-separable PWL concave functions over fixed regions defined by the polytopes in $\mathcal{P}$ is the similar set of functions which is not required to be defined over any fixed set of polytopes. Specifically, suppose we have $p \in \mathbb{N}$ affine functions $(v^j, \eta_j) \in \mathbb{R}^m \times \mathbb{R}, j = 1, \ldots, p$; then the function

$$V(s) = \min_{j=1,\ldots,p} \{v^j s + \eta_j\} \tag{3.11}$$

is PWL concave and V implies a partition of $[0, d]$ into (at most) p polytopes via

$$P_j = \{s \in [0, d] : v^j s + \eta_j \leq v^\ell s + \eta_\ell, \forall\ \ell \neq j\}. \tag{3.12}$$

This larger set of PWL concave functions defined by p affine functions over *variable* regions can be described simply as $\mathcal{V}_{\text{NV}} = (\mathbb{R}^m \times \mathbb{R})^p$; unfortunately, this parametrization is not one-to-one, since a permutation of the affine functions' indexing would leave the resulting PWL function unchanged. However, modeling remains exactly the same as before, and is accomplished with (3.8).

Example 3.2 (Example 2.2 continued)**.** Consider again the maritime IRP model (2.3). Unlike Example 3.1, suppose we cannot assume separability, perhaps because different groups of ports are geographically close to one another and thus influence each others' routing decisions. If we are unable to determine polyhedral regions within which inventory values are approximately affine, a variable region value function can determine the regions instead, while at the same time determining value in each region.

The fitting problem (3.1) becomes

$$\min_{(v,\eta)} \sum_{k=1}^{K} \Big| \min_{j=1,\ldots,p} \{v^j s^k + \eta_j\} - \nu^k \Big|^q, \tag{3.13}$$

and is non-convex because the min operator appears inside the absolute value. (Nonetheless, to the best of our knowledge there is no known proof of the problem's NP-hardness.)

For any $s \in [0, d]$, we can model $V(s)$ disjunctively as

$$\begin{gathered} V(s) \leq v^j s + \eta_j, \forall\ j = 1, \ldots, p \\ \bigvee_{j=1}^{p} V(s) \geq v^j s + \eta_j. \end{gathered}$$

This disjunction involves the union of unbounded polyhedra with different recession cones, so we cannot hope to model it in a MIP using traditional disjunctive programming techniques [3, 37]. Instead, we resort to a big-M method, where the disjunction

is replaced by

$$V(s) \geq v^j s + \eta_j - M(1 - y_j), \forall\ j = 1, \ldots, p$$
$$\sum_{j=1}^{p} y_j = 1, \quad y \in \{0,1\}^p.$$

For each s^k, define $\mu^k = V(s^k)$ and let $y^k \in \{0,1\}^p$. Then (3.1) is

$$\min \ \sum_{k=1}^{K} |\mu^k - \nu^k|^q \tag{3.14a}$$
$$\text{s.t. } \mu^k \leq v^j s^k + \eta_j, \forall\ j = 1, \ldots, p \tag{3.14b}$$
$$\mu^k \geq v^j s^k + \eta_j - M(1 - y_j^k), \forall\ j = 1, \ldots, p; k = 1, \ldots, K \tag{3.14c}$$
$$\sum_{j=1}^{p} y_j^k = 1, \forall\ k = 1, \ldots, K \tag{3.14d}$$
$$(v, \eta) \in (\mathbb{R}^m \times \mathbb{R})^p, \quad y \in \{0,1\}^{p \times K}, \quad \mu \in \mathbb{R}^K. \tag{3.14e}$$

Although technically correct, the preceding formulation suffers from symmetry, because a permutation of the j indices yields another feasible solution of equal objective. The following result addresses this issue.

Proposition 3.3. *An optimal solution of* (3.14) *satisfies*

$$v_1^1 \leq \ldots \leq v_1^p. \tag{3.15}$$

Proof. Let $(\tilde{v}, \tilde{\eta})$ be an optimal solution of (3.13), and let $\pi : \{1, \ldots, p\} \to \{1, \ldots, p\}$ be a permutation satisfying $\tilde{v}_1^{\pi(1)} \leq \cdots \leq \tilde{v}_1^{\pi(p)}$. Define

$$v_i^{j*} = \tilde{v}_i^{\pi(j)}, \forall\ i = 1, \ldots, m; j = 1, \ldots, p$$
$$\eta_j^* = \tilde{\eta}_{\pi(j)}, \forall\ j = 1, \ldots, p.$$

In words, (v^*, η^*) permutes the j-indices of $(\tilde{v}, \tilde{\eta})$ to order the resulting solution by the first coordinate of the v variables. Since the maximum operator is invariant under permutations, (v^*, η^*) is also optimal for (3.13). □

The addition of constraints (3.15) imposes an ordering of the feasible region and removes solution symmetry. Of course, in highly degenerate cases where the optimal solution has all first-coordinate slopes equal to one another, the symmetry would not be broken. However, the symmetry breaking constraints can be imposed in any dimension, so different variations of the problem can be implemented if degeneracy is encountered.

The LP relaxations of big-M formulations in MIP are notoriously weak. In order to partially mitigate this weakness in formulation (3.14), we consider an additional set of constraints, for which we first establish a technical result.

Lemma 3.4. *For each $j = 1, \ldots, p$, define the set*

$$S_j = \{\mu \in \mathbb{R}, a \in \mathbb{R}^p_+ : \mu = a_j; \mu \geq a_\ell, \forall\ \ell \neq j\},$$

and define also the set

$$S = \left\{\mu \in \mathbb{R}, a \in \mathbb{R}^p_+ : \mu \leq \sum_{j=1}^{p} a_j; \mu \geq a_j, \forall\ j = 1, \ldots, p\right\}.$$

Then $S = \operatorname{conv}\left(\bigcup_j S_j\right)$.

Proof. Both S and $S_j, \forall\ j$ are pointed polyhedral cones contained in the positive orthant. Let $e_j \in \mathbb{R}^p$ be the j-th unit vector. For each S_j, a complete set of extreme rays is

$$(\mu, a) = \left(1, e_j + \sum_{\ell \in T} e_\ell\right), \forall\ T \subseteq \{1, \ldots, p\} \setminus \{j\}.$$

This can be verified by noting that at any extreme ray, exactly one of the pair of constraints $\mu \geq a_\ell$ and $a_\ell \geq 0$ can be binding for each $\ell \neq j$. (Both are binding only at the origin.)

Similarly, a complete set of extreme rays of S is

$$(\mu, a) = \left(1, \sum_{j \in T} e_j\right), \forall\ T \subseteq \{1, \ldots, p\}, T \neq \varnothing.$$

As in the previous case, at any extreme ray exactly one of $\mu \geq a_j$ and $a_j \geq 0$ can be binding for each j. The constraint $\mu \leq \sum_j a_j$ ensures that at least one of the former is always binding.

The union over all j of the sets of extreme rays of S_j gives the set of rays for S, which proves the result. □

The preceding lemma gives a polyhedral description of the convex hull of a union of polyhedra with differing recession cones (see also [57, Theorem 1]). Letting $\mu = \mu^k$ and $a_j = -v^j s^k - \eta_j$ for each k, we apply the lemma as follows.

Corollary 3.5. *Suppose an optimal solution of* (3.13) *satisfies*

$$v^j s^k + \eta_j \leq 0, \forall\ j = 1, \ldots, p; k = 1, \ldots, K. \tag{3.16}$$

Then the following constraints are valid for (3.14)*:*

$$\mu^k \geq \sum_{j=1}^{p} (v^j s^k + \eta_j), \forall\ k = 1, \cdots K. \tag{3.17}$$

We can apply constraints (3.17) without loss of generality if we subtract a large enough number from all ν^k.

Model (3.14) has $\Theta(pK)$ binary variables, $\Theta(K + mp)$ continuous variables, and $\Theta(pK)$ constraints. Because of the relatively large number of binary variables, the model may only be computationally tractable for small-to-medium data sets. This computational difficulty would be especially apparent when $q = 2$ and (3.14) becomes a MINLP with convex quadratic objective. In these cases, heuristic methods may be used in tandem or instead of exact techniques. Magnani and Boyd [44] introduced a Gauss-Newton heuristic for (3.13); we reproduce it here in Algorithm 2 with our notation. Note that the linear fitting problem referred to in the algorithm is classical linear (affine) fitting, equivalent to (3.5) with $p = 1$ and without the concavity constraints.

Algorithm 2 Gauss-Newton heuristic for non-separable fitting over variable regions

randomly partition $\{s^k\}_{k=1}^K$ into p non-empty sets with pairwise non-intersecting convex hulls
repeat
  for $j = 1, \ldots, p$ **do**
    solve linear fitting problem for points in set j to obtain best-fit affine function (v^{j*}, η_j^*)
  end for
  set $V^*(s) \leftarrow \min_j \{v^{j*}s + \eta_j^*\}$
  partition $\{s^k\}_{k=1}^K$ into p sets according to (3.12), breaking ties arbitrarily
until partition is unchanged
return V^*

The invariance of functions in $\mathcal{V}_{\mathrm{NV}}$ under permutations of the j-indices creates a problem for the merge function. Specifically, for any $(v, \eta), (v', \eta') \in \mathcal{V}_{\mathrm{NV}}$, any permutation $\pi : \{1, \ldots, p\} \to \{1, \ldots, p\}$ yields a valid merge via

$$\big(\mathrm{merge}_\pi((v, \eta), (v', \eta'), \alpha)\big)^j = (1 - \alpha)(v^j, \eta_j) + \alpha((v')^{\pi(j)}, \eta'_{\pi(j)}).$$

The appropriate permutation may depend on the particular application; we present one tractable option.

One way to choose a permutation is by maximizing the "overlap" between partitions; i.e. solving

$$\max_\pi \sum_{j=1}^p \mathrm{vol}(P_j \cap P'_{\pi(j)}),$$

where $P \in \mathcal{P}$ and $P' \in \mathcal{P}'$ denote the partitions of $[0, d]$ implied by $(v, \eta), (v', \eta')$ according to (3.12). However, calculating the volumes involves high-dimensional integration over polytopes, which we cannot hope to solve efficiently. Instead, we propose the use of the ℓ_q norm in $\mathbb{R}^{m+1}$ as a proxy. This yields

$$\min_\pi \sum_{j=1}^p \|(v^j, \eta_j) - ((v')^{\pi(j)}, \eta'_{\pi(j)})\|_q, \tag{3.18}$$

where q may or may not be the same norm used in the fitting objective. These norms are readily calculated, and the resulting problem is a min-cost perfect matching.

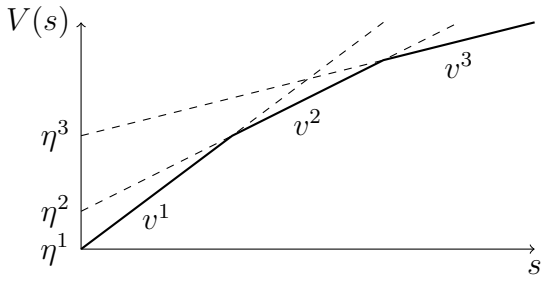

Figure 3: Univariate PWL concave function defined with the minimum operator.

3.4 Separable Functions over a Variable Grid

Whether the functions are defined over fixed or variable regions, the large number of parameters required for non-separable modeling and fitting in high dimensions remains an issue. We next present a compromise between the simplicity of separable functions and the versatility of non-separable functions over variable regions: We impose separability to maintain a manageable number of parameters, but allow variability in the bucket lengths that define the grid to increase versatility. In this case, we have

$$V(s) = \sum_{i=1}^{m} \min_{j=1,\dots,p} \{v_i^j s_i + \eta_i^j\}, \tag{3.19}$$

where $(v, \eta) \in \mathbb{R}^{m\times p} \times \mathbb{R}^{m\times p}$. We lose nothing by assuming that the slopes in each dimension are non-increasing, $v_i^1 \geq \cdots \geq v_i^p, \forall\ i$, and this in turn implies a non-decreasing order of intercepts, $\eta_i^1 \leq \cdots \leq \eta_i^p, \forall\ i$. Figure 3 shows a univariate example.

We use the following proposition to further simplify the set of functions.

Proposition 3.6. *For any function defined by* (3.19)*, we may assume*

$$\eta_i^1 = 0, \forall\ i = 2, \dots, m.$$

Proof. Let $V = (v, \eta)$ be defined as above, and let $\hat{V} = (v, \hat{\eta})$, where

$$\hat{\eta}_i^j = \begin{cases} \eta_i^j + \sum_{\ell=2}^{m} \eta_\ell^1, & i = 1 \\ \eta_i^j - \eta_i^1, & i \geq 2 \end{cases}, \forall\ j = 1, \dots, p.$$

The function $\hat{V}$ shifts intercepts in the first dimension up, and shifts all other intercepts down by the corresponding amount. For any $s \in [0, d]$, we then have $V(s) = \hat{V}(s)$. □

Corollary 3.7. *The set of separable functions defined over p variable buckets in each dimension has a parametrization given by*

$$\begin{aligned}\mathcal{V}_{SV} = \{(v, \eta) \in \mathbb{R}^{m\times p} \times \mathbb{R}^{m\times p} : v_i^1 \geq \cdots \geq v_i^p, \eta_i^1 \leq \cdots \leq \eta_i^p, \forall\ i = 1, \ldots, m; \\ \eta_i^1 = 0, \forall\ i = 2, \ldots, m\}.\end{aligned} \tag{3.20}$$

We can model these functions in one of two equivalent ways. The incremental model (3.4) still applies using

$$b_i^j = \frac{\eta_i^{j+1} - \eta_i^j}{v_i^j - v_i^{j+1}}, \forall\ j = 1, \ldots, p-1. \tag{3.21}$$

However, this equation is valid only if all univariate affine functions (v_i^j, η_i^j) are minimal for some $s_i \in [0, d_i]$. Otherwise, we have $b_i^{j+1} - b_i^j < 0$ for some j; however, we can eliminate this problem by re-parametrizing the set of functions as

$$\begin{aligned}\mathcal{V}_{\text{SV}} = \{(v, \eta, b) \in \mathbb{R}^{m\times p} \times \mathbb{R} \times \mathbb{R}^{m\times(p+1)} : v_i^1 \geq \cdots \geq v_i^p, \forall\ i = 1, \ldots, m \\ 0 = b_i^0 \leq \cdots \leq b_i^p = d_i, \forall\ i = 1, \ldots, m\}.\end{aligned} \tag{3.22}$$

The second modeling possibility mirrors (3.8). Define auxiliary variables $\nu \in \mathbb{R}^m$; any $V = (v, \eta) \in \mathcal{V}_{\text{SV}}$ is given by

$$V(s) = \max\ \sum_{i=1}^{m} \nu_i \tag{3.23a}$$

$$\text{s.t. } \nu_i \leq v_i^j s_i + \eta_i^j, \forall\ j = 1, \ldots, p \tag{3.23b}$$

$$\nu \in \mathbb{R}^m, \tag{3.23c}$$

where (v, η) follows the parametrization in (3.20).

We also use the second modeling formulation as a starting point for the fitting problem. Applying the same modeling techniques used in (3.14) yields

$$\min\ \sum_{k=1}^{K} \left| \sum_{i=1}^{m} \mu_i^k - \nu^k \right|^q \tag{3.24a}$$

$$\text{s.t. } \mu_i^k \leq v_i^j s_i^k + \eta_i^j, \forall\ i = 1, \ldots, m; j = 1, \ldots, p \tag{3.24b}$$

$$\mu_i^k \geq v_i^j s_i^k + \eta_i^j - M(1 - y_{ij}^k), \forall\ i = 1, \ldots, m; j = 1, \ldots, p; k = 1, \ldots, K \tag{3.24c}$$

$$\sum_{j=1}^{p} y_{ij}^k = 1, \forall\ i = 1, \ldots, m; k = 1, \ldots, K \tag{3.24d}$$

$$v_i^1 \geq \cdots \geq v_i^p, \forall\ i = 1, \ldots, m \tag{3.24e}$$

$$\eta_i^1 \leq \cdots \leq \eta_i^p, \forall\ i = 1, \ldots, m \tag{3.24f}$$

$$\eta_i^1 = 0, \forall\ i = 2, \ldots, m \tag{3.24g}$$

$$(v, \eta) \in (\mathbb{R}^m \times \mathbb{R}^m)^p, \quad y \in \{0, 1\}^{m \times p \times K}, \quad \mu \in \mathbb{R}^{m \times K}. \tag{3.24h}$$

Paradoxically, the separability of the functions, which simplifies modeling in various ways, complicates fitting by increasing the number of variables required in the formulation. In particular, the number of binary variables increases by an order of magnitude to $\Theta(mpK)$, which may again limit the model's scalability. Fortunately, the Gauss-Newton heuristic developed in [44] can be adapted for this setting as well.

To adapt the heuristic, we need a *first-order approximation* fitting problem. Given a partition b of $[0, d]$, define an assignment function $h(i, k) = \min\{j : s_i^k < b_i^j\}, \forall\ i = 1, \ldots, m, k = 1, \ldots, K$; this function indicates which bucket a point s^k belongs to in each dimension i. The first-order approximation fitting problem is

$$\min \sum_{k=1}^{K} \left| \sum_{i=1}^{m} v_i^{h(i,k)} s_i^k + \eta_i^{h(i,k)} - \nu_k \right|^q \tag{3.25a}$$

$$\text{s.t. } v_i^1 \geq \cdots \geq v_i^p, \forall\ i = 1, \ldots, m \tag{3.25b}$$

$$\eta_i^1 \leq \cdots \leq \eta_i^p, \forall\ i = 1, \ldots, m \tag{3.25c}$$

$$\eta_i^1 = 0, \forall\ i = 2, \ldots, m \tag{3.25d}$$

$$(v, \eta) \in \mathbb{R}^{m \times p} \times \mathbb{R}^{m \times p}. \tag{3.25e}$$

This problem is convex, and therefore efficiently solvable. Intuitively, for a separable PWL function defined over b, the affine function $\sum_i (v_i^{h(i,k)} s + \eta_i^{h(i,k)})$ gives the derivative, or first-order approximation, at point s^k. This formulation thus attempts

to fit a function from $\mathcal{V}_{\text{SV}}$ based on the derivative implied by grid b. If the resulting best-fit function changes the grid, we can reassign and repeat. Algorithm 3 details the heuristic.

Algorithm 3 Gauss-Newton heuristic adapted for separable fitting

pick a random grid b of the domain $[0, d]$ with p buckets in each dimension
repeat
  (re)define h according to b
  solve first-order approximation fitting problem (3.25) with h
  let (v^*, η^*) be the optimal solution
  update b according to (v^*, η^*) via (3.21)
until b is unchanged
return (v^*, η^*)

Either parametrization of $\mathcal{V}_{\text{SV}}$ is a polyhedral cone, and therefore merging is again simply a convex combination. However, we should note that the merge functions implied by each parametrization are not equivalent: For a fixed α, if we merge two functions parametrized by (3.20) and then compute the resulting grid, we obtain a different set of bucket lengths than if we convert each function to its equivalent parametrization according to (3.22) and merge the two resulting grids. The converse starting from a function parametrized via (3.22) also holds.

3.5 Hybrid Functions

Although we have presented the different function classes as separate entities, they may in fact be combined in a variety of ways according to the user's particular application. Since the minimum of two concave functions is again concave, we could define a hybrid function

$$V(s) = \min\{V_1(s), V_2(s)\},$$

where V_1 and V_2 belong to different classes of functions. Figure 4 gives a two-dimensional example. The modeling, fitting and merging can be adapted to address any "mix and match" scenario based on the principles we have outlined in this chapter.

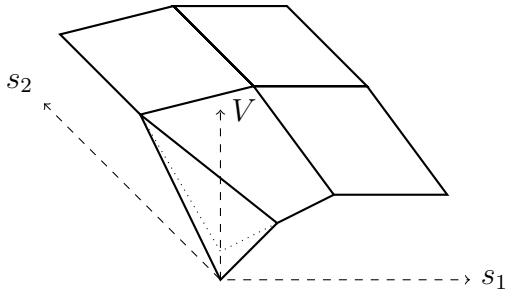

Figure 4: Bivariate hybrid PWL concave function.

Example 3.3 (Example 2.2 continued)**.** Consider once more the maritime IRP model introduced in Chapter 2. If the number of ports in the model is relatively large, it is computationally intractable to employ a non-separable value function. However, it may be that inventory value cannot be accurately approximated by a separable function over its entire domain. For instance, we may believe that two consumer ports, j_1 and j_2, have inventory value that is separable when the combined inventory is above a certain amount I. However, when $s_{j_1} + s_{j_2} \leq I$, a certain interaction between the two ports further reduces inventory value. In this case, a hybrid value function similar to Figure 4's example may be an appropriate compromise.

CHAPTER IV

INFINITE-HORIZON LOT SIZING

In Chapter 2 we presented an algorithm that approximates inventory value in a generic multi-period or infinite-horizon supply chain model with maximization objective. The algorithm uses DP and data fitting techniques to construct a PWL concave approximate inventory value function. The choice of PWL concave functions is motivated by true maximization MIP value functions, known to be PWL, superadditive and upper semi-continuous. However, though the aforementioned structure of value functions is well-known in finite MIP's, the infinite case is not as well understood. This chapter studies a particular infinite-horizon problem and its value function; we show that the structural characteristics of the finite case carry over to the infinite case with only minor adjustments. In addition, as a proof of concept we compare heuristic solutions generated with Algorithm 1's approximate value function to the optimal solutions.

The particular problem we study is a discounted, infinite-horizon version of the classical single-item uncapacitated LSP, already introduced in Example 2.1. Suppose we need to manage the production schedule for a single item that experiences constant per-period demand w. There is no production or inventory capacity, and all demand must be met each period, either with items produced that period, or items in inventory. Every period we produce, we incur a fixed cost $f > 0$ and a variable cost of $c > 0$ per unit produced. Items left over at the end of the period after demand is met incur a holding cost of $h > 0$ per unit. We can model this problem as

$$C(s, w) = \min \sum_{t=1}^{\infty} \gamma^{t-1}(fx_t + cz_t + hs_t) \tag{4.1a}$$

$$\text{s.t. } z_t + s_{t-1} - s_t = w, \forall\ t = 1, \ldots \tag{4.1b}$$

$$M_w x_t - z_t \geq 0, \forall\ t = 1, \ldots \tag{4.1c}$$

$$s_0 = s \tag{4.1d}$$

$$x_t \in \{0, 1\}; z_t, s_t \geq 0, \forall\ t = 1, \ldots, \tag{4.1e}$$

where $\gamma \in [0, 1)$ and $M_w > 0$ is a large number; we add the subscript to make its dependence on w explicit. We take the set of feasible solutions to be a subset of the sequences (x_t, z_t, s_t) with well-defined and finite objective (cf. [60]). Unlike our generic FCNF model (2.1), we employ a minimization objective more natural in the current case; this also motivates our use of cost notation instead of value notation in this chapter. We also explicitly parametrize both the initial inventory s and the demand rate w, and assume both are non-negative. We include the degenerate case $w = 0$ so the domain of C is the closed cone $\mathbb{R}^2_+$.

Problem (4.1) and its many variations have a long history in operations research. The structure of optimal solutions in the finite, dynamic case was studied in the seminal work of Wagner and Whitin [71], and many researchers have since attempted to generalize their results for more complex models. Most authors have given results pertaining to optimal solutions' structure, and related issues such as regeneration points and replenishment intervals (see Theorem 4.1 below.) For example, Graves and Orlin [32] study a cyclic generalization of (4.1) and its long-run average cost counterpart, proving various structural results and providing algorithmic implementations with worst-case running times.

However, the value function itself has received relatively little attention. The one important exception is [29], where the authors give a closed-form expression for the value function of a continuous-time, average-cost single-item lot-sizing problem. This chapter's theoretical results can be viewed as the analogue in the discrete-time, discounted case.

4.1 Optimal Solutions

The following theorem summarizes the structure of optimal solutions for (4.1). These results are well-known throughout the MIP, DP and inventory control community, and similar results exist for different variations of LSP. [71] has the original proofs for a finite variant of the problem; however, the arguments carry over to (4.1) with minor changes.

Theorem 4.1. *Suppose $w > 0$, and let $t^* = \lfloor \frac{s}{w} \rfloor + 1$. Any optimal solution to* (4.1) *satisfies the following statements.*

i) $z_t = 0, \forall\ t < t^*$.

ii) $z_{t^*} > 0$, *and* $s_{t^*-1} + z_{t^*} = k_{t^*} w$ *for some* $k_{t^*} \in \mathbb{N}$.

iii) $s_{t-1} z_t = 0, \forall\ t > t^*$, *and if* $z_t > 0$, *then* $z_t = k_t w$, *for some* $k_t \in \mathbb{N}$.

Proof. For $t < t^*$, if $z_t > 0$ we can always postpone production and decrease the objective. This also implies $z_{t^*} > 0$ by feasibility. For $t > t^*$, if we produce an amount that is not an integer multiple of w, we can always decrease production to the largest integer multiple of w and postpone the remaining production to the next period of positive production while improving the objective. This implies we only produce when incoming inventory is at zero. If $s - (t^* - 1)w > 0$, a similar argument shows that $s_{t^*-1} + z_{t^*}$ is an integer multiple of w. □

The theorem states that optimal solutions have a *replenishment interval* structure: Production is always equal to the cumulative demand for an interval of consecutive periods; for (4.1), we shall see that replenishment intervals are always equal because the data is stationary, except in degenerate cases when two interval lengths are optimal.

4.2 The Value Function

The most basic non-trivial case occurs when the initial inventory is zero.

Proposition 4.2. *Suppose $w > 0$. Then*

$$C(0, w) = \min_{k \in \mathbb{N}} \left\{ \frac{1}{1 - \gamma^k} \left(f + kcw + hw \sum_{\ell=1}^{k-1} \gamma^{\ell-1}(k - \ell) \right) \right\}.$$

For any w, at most two consecutive integers $k(w), k(w) + 1$ minimize the quantity inside the brackets.

Proof. This well-known proof uses standard DP techniques; see, e.g., [56]. By Theorem 4.1, we know that production must occur in integer multiples of w, and thus all future inventories will be integer multiples of w. We can thus consider a finite-state and -action DP with state space $S = \{0, \ldots, K\}$, where the integer K is chosen large enough. The set of feasible actions (i.e. actions with finite cost) are the "do nothing" action for all positive states, and the actions corresponding to a replenishment of length k, $k = 1, \ldots, K$ in the zero state. The quantity inside the minimization bracket is precisely the present value of replenishing inventory every k periods into perpetuity, and the minimum over all such quantities gives the optimal policy. Moreover, as a function of k the expression inside the brackets is strictly convex and eventually increasing, so the minimum over all natural numbers can be achieved by at most two consecutive numbers. □

This result also shows that we can take $M_w \geq (k(w) + 1)w$ in (4.1) to guarantee the feasibility of all optimal solutions, where $k(w)$ is an optimal replenishment length.

Corollary 4.3. *C is subadditive.*

Proof. This proof is almost identical to the finite case; see [51]. Let s, s' and w, w' be pairs of starting inventories and demands, with respective optimal solutions (x_t, z_t, s_t) and (x'_t, z'_t, s'_t). Then $\hat{x}_t = \max\{x_t, x'_t\}, (\hat{z}_t, \hat{s}_t) = (z_t, s_t) + (z'_t, s'_t)$ is feasible for $C(s +$

$s', w + w'$), where we take $M_{w+w'} = M_w + M_{w'} \geq (k(w + w') + 1)(w + w')$, and has an objective no greater than $C(s, w) + C(s', w')$. □

Corollary 4.4. $C(s, w) = C(0, w) - cs, \forall\ 0 \leq s < w, \forall\ w > 0.$

Proof. For simplicity, assume the optimal replenishment length is unique, and let $k(w) \in \mathbb{N}$ be the length. It suffices to prove that any optimal solution satisfies $x_1 = k(w)w - s$. Suppose not; by Theorem 4.1, $x_1 = k'w - s$, for some $k' \neq k(w)$. Then

$$f + c(k'w - s) + hw \sum_{\ell=1}^{k'-1} \gamma^{\ell-1}(k' - \ell) + \gamma^{k'} C(0, w) \leq$$
$$f + c(k(w)w - s) + hw \sum_{\ell=1}^{k(w)-1} \gamma^{\ell-1}(k(w) - \ell) + \gamma^{k(w)} C(0, w)$$

However, s can be eliminated from both sides, and the resulting relation implies that k' is an optimal replenishment lenght, contradicting our assumption. □

Corollary 4.5. $C(s, w) = h(s - w) + \gamma C(s - w, w), \forall\ s \geq w, \forall\ w.$

Proof. Follows directly from Theorem 4.1(i). □

The next theorem summarizes the preceding results into a single formula.

Theorem 4.6. *C is given by*

$$C(s, 0) = \frac{hs}{1 - \gamma}, \forall\ s \geq 0 \tag{4.2}$$

$$C(0, w) = \min_{k \in \mathbb{N}} \left\{ \frac{1}{1 - \gamma^k} \left(f + kcw + hw \sum_{\ell=1}^{k-1} \gamma^{\ell-1}(k - \ell) \right) \right\}, \forall\ w > 0 \tag{4.3}$$

$$C(s, w) = C(0, w) - cs, \forall\ 0 \leq s < w \tag{4.4}$$

$$\begin{aligned} C(s, w) = s \left(h \left(\frac{1 - \gamma^k}{1 - \gamma} \right) - \gamma^k c \right) - w \left(h \sum_{\ell=1}^{k} \ell \gamma^{\ell-1} - \gamma^k kc \right) \\ + \gamma^k C(0, w), \forall\ kw \leq s < (k + 1)w, \forall\ k \in \mathbb{N}. \end{aligned} \tag{4.5}$$

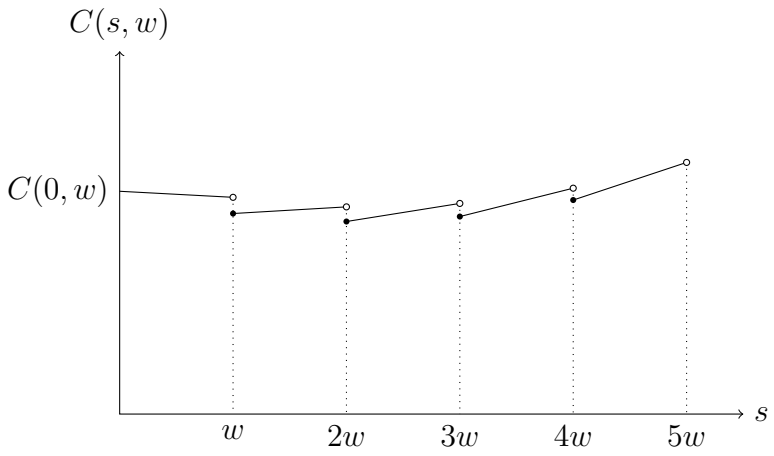

Figure 5: Single-item LSP value function for fixed $w > 0$ with $k(w) = 2$.

Proof. The first equation is directly obvious but also follows by setting $t^* = \infty$ in the proof of Theorem 4.1. The second and third equations are restatements previous results. The last equation follows by induction from Corollary 4.5. □

Figure 5 shows an example plot of $C(s, w)$ for a fixed $w > 0$ with optimal replenishment interval of two periods. The discontinuities in C occur along the lines $s = kw$, for each $k \in \mathbb{N}$. Within each region $\{(s, w) : kw \leq s < (k + 1)w\}$, $C(s, w)$ is PWL and continuous. Figure 6 shows a sample plot with $s = 0$. Figure 7 shows a three-dimensional sample plot of C.

Corollary 4.7. *C is PWL, with a countably infinite number of regions in which it is affine.*

This last result contrasts with the finite MIP case, in which value functions are PWL but defined by finitely many regions [13, Theorem 6.1]. However, any bounded subset of $\mathbb{R}^2_+$ that doesn't intersect the ray $w = 0$ contains finitely many regions in which C is affine.

Corollary 4.8. *C is lower semi-continuous.*

Proof. ($w > 0$) If w is positive, discontinuities occur when s is an integer multiple of

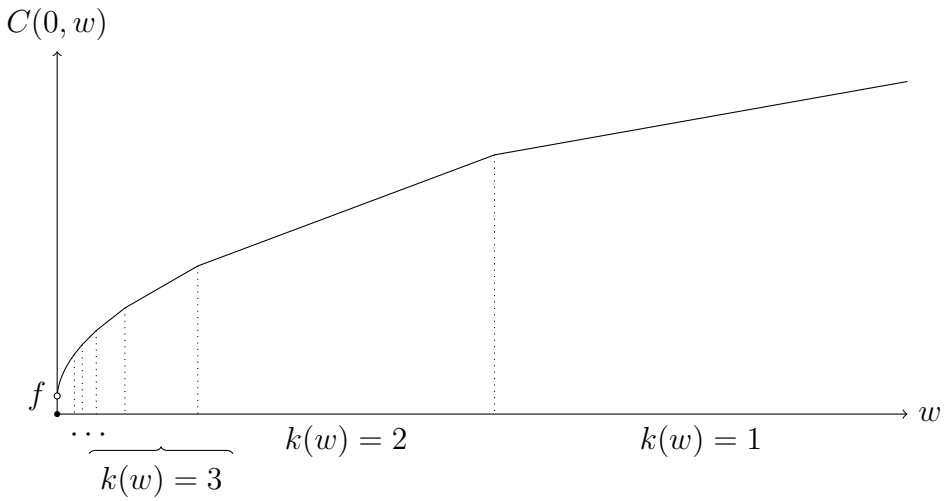

Figure 6: Single-item LSP value function for $s = 0$.

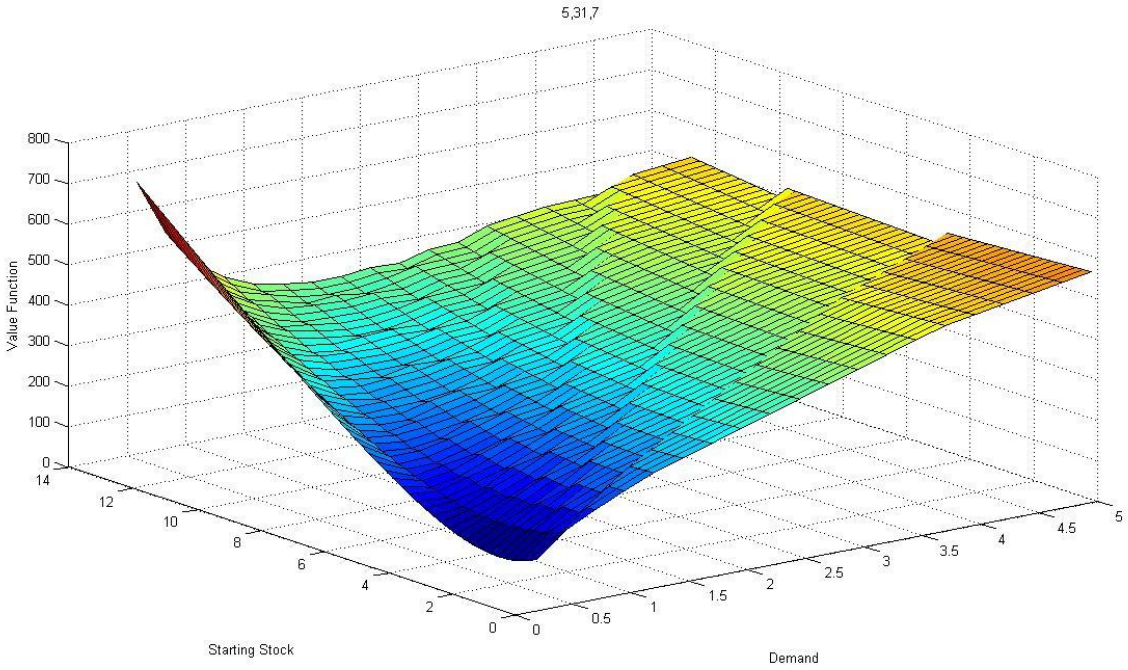

Figure 7: Three-dimensional rendering of sample single-item LSP value function.

w. First suppose that $s = w$; then using the basic fact $C(0, w) \geq \frac{cw}{1-\gamma}$, we have

$$C(0, w) - cw \geq \gamma C(0, w),$$

where the quantity on the left is $\lim_{s \uparrow w} C(s, w)$ and the right-hand side is $C(w, w)$. A similar argument, also using $C(0, w) \geq \frac{cw}{1-\gamma}$, establishes lower semi-continuity for $s = kw, k \geq 2$.

($w = 0$) The proof is trivial for $C(0, 0)$, since all other function values are positive. So suppose $s > 0$, and let $(\hat{s}, \hat{w})$ satisfy $\hat{s} > 0$, $0 < \hat{w} < \frac{\hat{s}}{2}$. Using the identity

$$\sum_{\ell=1}^{k} \ell \gamma^{\ell-1} = \frac{k\gamma^{k+1} - k\gamma^k + 1 - \gamma^k}{(1-\gamma)^2},$$

we have

$$\begin{aligned}
C(\hat{s}, \hat{w}) &= \hat{s}\left[h\left(\frac{1 - \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor}}{1-\gamma}\right) - \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} c\right] + \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} C(0, \hat{w}) \\
&\quad - \hat{w}\left[\frac{h}{(1-\gamma)^2}\left(\left\lfloor \frac{\hat{s}}{\hat{w}} \right\rfloor \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor + 1} - \left\lfloor \frac{\hat{s}}{\hat{w}} \right\rfloor \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} + 1 - \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor}\right) - \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} \left\lfloor \frac{\hat{s}}{\hat{w}} \right\rfloor c\right] \\
&= \frac{h\hat{s}}{1-\gamma} - h\hat{w}\left(\frac{\hat{s}}{\hat{w}} - \left\lfloor \frac{\hat{s}}{\hat{w}} \right\rfloor\right)\frac{\gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor}}{1-\gamma} + \hat{w}\left(\frac{\hat{s}}{\hat{w}} - \left\lfloor \frac{\hat{s}}{\hat{w}} \right\rfloor\right)\gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} c \\
&\quad - h\hat{w}\left(\frac{1 - \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor}}{1-\gamma}\right) + \gamma^{\lfloor \frac{\hat{s}}{\hat{w}} \rfloor} C(0, \hat{w}) \to \frac{hs}{1-\gamma} = C(s, 0),
\end{aligned}$$

as $(\hat{s}, \hat{w}) \to (s, 0)$. □

4.3 Approximate Value Function Comparison

In the remainder of the chapter, we report the results of a set of experiments designed to validate the algorithm presented in Chapter 2: We generate a value function for instances of (4.1) using Algorithm 1, and then compare solutions obtained by solving a single-period instance with the value function to optimal solutions as given by Theorems 4.1 and 4.6.

In our first experiment, we use an instance with the following parameters: $f = 3$, $c = 1$, $h = 2$, $\gamma = 0.99$, $w = 1$. The reader may verify that the optimal replenishment quantity is $2w = 2$. Our choice of value function class is a two-bucket PWL

convex function with fixed bucket points; see Section 3.1. (The LSP is formulated in minimization terms, and therefore we swap concave functions for convex ones.) As function breakpoint we use $b^1 = w = 1$, reasoning as in Example 3.1 that the most significant change in value occurs below one period's consumption. To run Algorithm 1, we set an upper bound $b^2 = 2w = 2$.

Table 1 shows the result of this experiment. After obtaining the value function, we solve instances from several different starting inventories, shown in the first column. The resulting solution's production quantity is shown in the second column, and the optimal production quantity given by Theorems 4.1 and 4.6 is shown for reference in the third. Clearly, the approximate value function in this case implies exactly the optimal production quantity for all the starting inventories tested.

Table 1: Lot-sizing experiment results for two-period optimal replenishment, using fixed-bucket value function.

Start. Inv.	Prod. Qty.	Opt. Prod. Qty.
0.0	2.00	2.00
0.2	1.80	1.80
0.4	1.60	1.60
0.6	1.40	1.40
0.8	1.20	1.20
1.0	0.00	0.00
1.2	0.00	0.00
1.4	0.00	0.00
1.6	0.00	0.00
1.8	0.00	0.00
2.0	0.00	0.00

The first experiment's results are optimal in part because of the choice of bucket breakpoint b^1, which is exactly w less than the optimal reorder quantity. In the next experiment, we use the same instance but instead generate a two-bucket PWL convex value function with a variable breakpoint (cf. Sections 3.3 and 3.4). The resulting value function has a shifted breakpoint $b^1 = 1.54$.

Table 2 outlines the second experiment's results. In this case, the solution's production quantity reflects the shifted breakpoint and is equal to 2.54 when there is

no initial inventory, an excess of 27%. In general, the order-up-to quantity implied by the value function is 2.54 instead of 2. To put this discrepancy into perspective, it is helpful to calculate the present value of each policy. Using Theorem 4.6, the reader may verify that $C(0,1) = 351.76$, whereas if the initial order is 2.54 and the subsequent ordering is optimal, the present value is

$$f + 2.54c + h(1.54 + \gamma 0.54) + \gamma^2(C(0,1) - 0.54c) = 353.92,$$

an increase of 0.61%. However, if the order-up-to quantity 2.54 is used into perpetuity, the present value is

$$f + 2.54c + h(1.54 + \gamma 0.54) + \gamma^2\left(C(0,1) + \frac{0.54h}{1-\gamma}\right) = 460.30,$$

an increase of 30.86%.

Table 2: Lot-sizing experiment results for two-period optimal replenishment, using variable-bucket value function.

Start. Inv.	Prod. Qty.	Opt. Prod. Qty.
0.0	2.54	2.00
0.2	2.34	1.80
0.4	2.14	1.60
0.6	1.94	1.40
0.8	1.74	1.20
1.0	0.00	0.00
1.2	0.00	0.00
1.4	0.00	0.00
1.6	0.00	0.00
1.8	0.00	0.00
2.0	0.00	0.00

As a final test, we repeated the second experiment for an instance with a much larger optimal replenishment quantity. The new instance's parameters that differ from the previous ones are $f = 60$, $c = 10$, $h = 0.25$, yielding an optimal replenishment quantity of $18w = 18$; we also increased the upper bound to $b^2 = 20w = 20$. In this case, the value function generated by Algorithm 1 has a breakpoint $b^1 = 18.24$,

Table 3: Lot-sizing experiment results for 18-period optimal replenishment, using variable-bucket value function.

Start. Inv.	Prod. Qty.	Opt. Prod. Qty.
0.0	19.24	18.00
0.2	19.04	17.80
0.4	18.84	17.60
0.6	18.64	17.40
0.8	18.44	17.20
1.0	0.00	0.00
1.2	0.00	0.00
1.4	0.00	0.00
1.6	0.00	0.00
1.8	0.00	0.00
2.0	0.00	0.00

yielding an order-up-to quantity of 19.24, as Table 3 shows. In this case, the order-up-to quantity exceeds the optimum by only 6.89%.

Using the same argument as before, we have $C(0,1) = 1,669.53$, whereas if the first order quantity is 19.24 and subsequent ordering is optimal, the present value is

$$f + 19.24c + h\sum_{\ell=1}^{18}\gamma^{\ell}(19-\ell) + \gamma^{19}(C(0,1) - 0.24c) = 1,670.15,$$

an increase of 0.037%. Similarly, if the order-up-to quantity 19.24 is used into perpetuity, the present value is

$$0.24c + \frac{1}{1-\gamma^{19}}\left(f + 19c + h\sum_{\ell=1}^{18}\gamma^{\ell-1}(19-\ell)\right) + \frac{0.24h}{1-\gamma} = 1,679.11,$$

an increase of 0.57%. The modest increase in the present value of the policy implied by the approximate value function in this second case suggests that the approximate approach is more effective in situations with longer replenishment intervals.

Why does the variable-bucket value function overshoot the optimal policy? Figure 8 plots the value function for the instance used in the first and second experiment. Along with the true value function, plotted in a solid line, we plot its best-fit two-bucket PWL convex functions obtained by solving two different problems: The best-fit function with fixed bucket at $b^1 = 1$ is plotted in a dotted line, while the best-fit

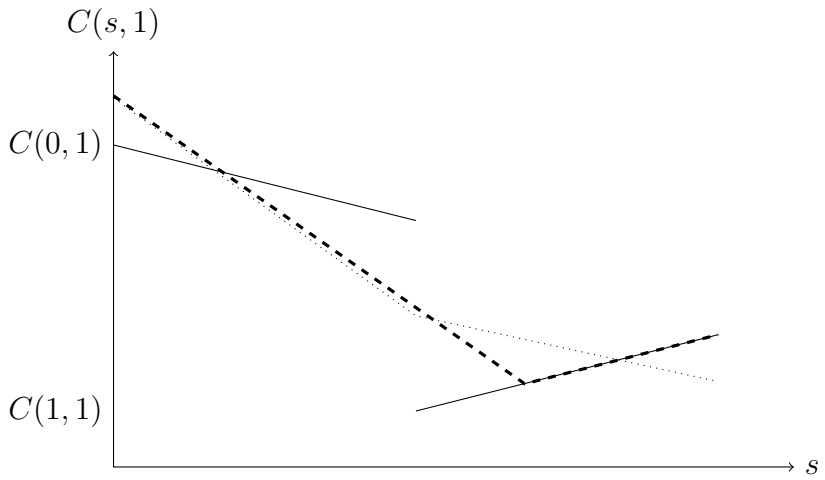

Figure 8: Plot of C with fixed- and variable-bucket PWL convex best-fit functions.

variable-bucket function is plotted with a dashed line. The variable-bucket best-fit function decreases the squared error by over 40%; however, in this example the resulting breakpoint implies a sub-optimal policy, while the fixed-bucket function with a worst fit implies the optimal policy. The example indicates that fixed-bucket or -region functions may be more appropriate when we have additional information about the problem and its value function. Nonetheless, all three experiments do show that we can use approximate value functions to generate solutions of high quality; the particular class of value function and the resulting solution's quality depend on our knowledge of the problem.

CHAPTER V

MODEL CASE STUDIES

This chapter details several experiments designed to test the approximate inventory valuation algorithm described in Chapter 2. The experiments use the approximate inventory value functions constructed by the algorithm to generate heuristic solutions for multi-period models, and we infer the accuracy of the value function by examining the quality of these solutions.

The specific models we use in the experiments are IRP's inspired by applications in maritime petrochemical transportation [30]. However, in its most general form the IRP has applications in virtually any transportation setting. We include experiments for two types of instances. The first type of instance includes a simplified travel time scheme, in which all voyages are assumed to take place within a single period. The more complex instances allow for different travel times depending on origin-destination pairs.

All computations were performed on a Xeon 2.66 GHz workstation with 8 Gb of RAM. CPLEX 11 was used as the optimization solver.

5.1 Simplified Travel Time

Our first set of experiments concerns the simple-time model (2.3) already described in Chapter 2. The instances used in the computational study were randomly generated based on problem characteristics found in real-world situations. For example, the transportation costs were randomly generated to satisfy the triangle inequality, with travel from suppliers to consumers about 15 times as costly as travel within either group. Similarly, production and consumption rates $w_i, i \in S \cup C$ were randomly generated so that the expected total production and the expected total consumption

per period were both approximately 85% of vessel capacity. However, a third-party supply port and a third-party consumption port were added to the model in case the two total rates were not equal. The number of suppliers and consumers in each instance was six and five respectively (plus third parties).

The value function class we chose for these experiments is the set of separable PWL concave functions over fixed grids, described in Section 3.1. Each port's inventory domain was divided into three buckets of equal length.

In the first experiment, we constructed five instances of (2.3) and randomly generated five starting inventories, for a total of 25 problem instances. Letting the planning horizon length be $T = 60$, we used two methods to generate solutions:

- 1P: Construct a solution one period at a time by solving the single-period subproblem (with approximate value function in the objective) to optimality and using its optimal ending inventory as the next starting inventory.
- 60P: Attempt to solve the entire 60-period instance (with approximate value function in the objective) using CPLEX, emphasizing feasibility, with a one-hour time limit.

We added the value function to the 60-period instances to ensure a fair comparison; i.e. both methods optimized with respect to the same objective. Table 4 presents the results of this experiment, with all times reported in seconds. Column Inst indexes the instance-starting inventory pair. Columns 1P Obj and 60P Obj respectively give the objective value of each solution. The Bound column has the best dual bound found by CPLEX within the time limit. Columns 1P Time and 60P Time show the computation time used to obtain the best solution via both methods.

We observe that the single-period heuristic generates competitive solutions, beating the 60P solution in all but three instances: 1-3, 3-3 and 5-1. However, the more notable result is the drastic time difference. The single-period heuristic is able to

Table 4: Simplified time instances, first experiment.

	Obj				Time (sec.)		
Inst	1P	2P	60P	Bound	1P	2P	60P
1-1	-810.19	-804.50	-827.59	-753.51	11	226	1963
1-2	-817.10	-823.54	-845.76	-757.60	14	345	1040
1-3	-892.84	-874.21	-847.39	-755.11	66	21362	3284
1-4	-813.53	-803.40	-828.75	-748.21	17	308	3246
1-5	-809.03	-807.93	-838.18	-754.80	13	261	2902
2-1	-745.49	-729.36	-750.48	-676.77	10	105	804
2-2	-715.36	-715.33	-745.97	-677.38	7	73	3253
2-3	-711.62	-708.11	-737.63	-663.68	9	141	1568
2-4	-708.93	-710.70	-733.13	-668.69	7	83	2953
2-5	-726.17	-721.23	-753.60	-676.24	8	58	1610
3-1	2794.40	2795.62	2771.41	2873.24	42	3060	235
3-2	2812.60	2818.29	2786.35	2902.84	52	2517	3309
3-3	2578.34	2667.05	2669.78	2841.47	193	48455	3147
3-4	2795.49	2798.76	2759.93	2875.42	43	1671	3313
3-5	2805.63	2808.43	2771.71	2880.39	38	1830	531
4-1	3342.36	3348.26	3300.65	3421.72	21	405	3424
4-2	3353.05	3363.90	3318.68	3424.61	19	245	3380
4-3	3287.23	3296.66	3266.03	3371.86	19	459	3266
4-4	3290.45	3296.85	3271.44	3363.51	22	324	3260
4-5	3335.52	3342.43	3305.05	3410.97	20	393	2092
5-1	-820.62	-812.62	-820.42	-741.29	13	188	3406
5-2	-803.15	-796.33	-834.64	-739.56	12	141	3347
5-3	-786.86	-782.59	-799.52	-721.43	16	216	2827
5-4	-797.96	-787.52	-819.95	-730.08	12	164	1895
5-5	-812.43	-812.41	-836.13	-738.71	11	173	3178

generate a solution in a few seconds (the average is 27.8,) while CPLEX does not find its best 60P solution until much later, after an average of 2,529 seconds. This drastic time difference suggests that one can generate a good solution quickly using the single-period heuristic.

The results of the first experiment suggested an additional experiment. Instead of constructing a solution by solving single-period instances, the same method could be applied with two-period instances linked by inventory levels:

- 2P: Construct a solution by solving the two-period subproblem (with approximate value function in the objective) to optimality and using its optimal ending inventory as the next starting inventory.

Table 4 shows the results of the two-period heuristic for the same instances, using

the same naming convention outlined earlier. Except for instances 1-2 and 2-4, the two-period heuristic solution has a better objective than the single-period heuristic solution and is only worse than the 60P solution in instances 1-3 and 3-3. The objective improvement, however, comes at the expense of a drastic increase in computation time that seems to depend on the particular instance. For example, the average time for all models derived from instances 2, 4 and 5 increases only by a factor of 15, from 14 seconds to 211 seconds. On the other hand, the two-period heuristic solution for instances 1-3 and 3-3 took many hours to complete, and indicates that the time required to solve even small problems of only a few periods depends heavily on the starting inventory vector.

In our second experiment, we wanted to investigate whether the solutions generated by the value function are myopic. Using the same parameters from the first experiment, we constructed five instances with planning horizon length $T = 10$ and used the five random starting inventories for a total of 25 problem instances. We generated solutions with two methods:

- 1P + 9P: Solve the single-period problem with approximate value function to optimality using CPLEX, and record the optimal solution. Use the optimal ending inventory as the starting inventory of a nine-period problem, solve this problem using CPLEX with optimality emphasis and a 24-hour limit. Concatenate the two solutions.

- 10P: Solve the entire ten-period problem using CPLEX, emphasizing optimality with a 24-hour limit.

Table 5 gives the results of the second experiment. As before, the naming convention for column INST pairs instances with starting inventories. The next two columns detail each solution's objective; 10P denotes the objective for the solution obtained

by solving the ten-period model, while (1P + 9P) denotes the objective of the solution obtained by solving the first period with a single-period model and then solving the remaining nine-period model. The BOUND column has the best bound for the ten-period model obtained by CPLEX as part of its optimization to generate the 10P solution.

Table 5: Simplified time instances, second experiment.

	OBJ (W/ VAL)		
INST	1P + 9P	10P	BOUND
1-1	-133.44	-134.55	-130.6
1-2	-138.17	-136.89	-134.29
1-3	-136.32	-135.3	-125.02
1-4	-128.16	-127.89	-124.75
1-5	-135.53	-135.29	-132.04
2-1	-118.18	-117.77	-113.97
2-2	-118.82	-118.61	-116.19
2-3	-108.16	-107.62	-104.04
2-4	-109.49	-109.08	-107.23
2-5	-117.85	-116.16	-114.84
3-1	462.63	462.68	467.07
3-2	480.46	480.2	487.15
3-3	380.83	385.82	393.72
3-4	446.18	463.5	467.14
3-5	465.69	467.41	472.05
4-1	590.29	590.55	598.82
4-2	595.63	597.39	601.12
4-3	532.89	531.75	546.46
4-4	532.22	534.67	540.14
4-5	577.41	580.57	587.33
5-1	-126.35	-125.89	-120.84
5-2	-125.24	-124.06	-120.74
5-3	-113.82	-112.83	-105.39
5-4	-116.22	-115.07	-111.44
5-5	-123.4	-123.5	-119.37

The results in Table 5 indicate that the solutions generated by the two methods are very similar. The (1P + 9P) solution's objective is on average only 0.55% less than the 10P objective, and is greater in four instances: 1-1, 3-2, 4-3 and 5-5. Moreover, the largest relative gap between objectives is only 3.74%, occurring on instance 3-4, and every other (1P + 9P) objective is within 1.5% of the 10P objective. Results are similar when we compare objectives to the CPLEX dual bound; the average relative gap for the (1P + 9P) solution is 3.01%, while the same average for the 10P solution

is 2.47%. Both measures indicate that implementing the 1P solutions results in an objective only about half a percentage point below what could be obtained by solving the entire ten-period model.

The preceding experiments do not compare an actual solution as it would be generated in practice. The typical approach to solving multi-period models involves solving an instance with a long planning horizon, e.g. $T = 60$, but then *implementing* only the decisions of the first few periods of the resulting solution. This process is then repeated inside of a rolling horizon framework. A more practical comparison therefore involves comparing the first few periods of solutions generated by our method against the traditional approach.

In our third experiment, we created three five-period instances and five starting inventories, for a total of 15 problem instances. We generated solutions via two methods:

- 5P: Solve the five-period instance (with approximate value function in the objective) using CPLEX, emphasizing feasibility and with an eight-hour time limit.
- TRAD: Solve a 60-period instance (without value function in the objective) using CPLEX, emphasizing feasibility with a 24-hour time limit. Truncate to the first five periods of the problem's solution.

However, comparing the two solutions is not a simple matter. One can consider the objective function value of each (without value function), but this does not take ending inventories into account, and is thus an incomplete comparison. On the other hand, if we use our approximate value function to measure the ending inventories' value, we are skewing the results in our favor (an optimal solution to the five-period problem with approximate value function included would always be the best solution according to this second measure.) We therefore decided on a compromise measure that attempts to capture the inherent value of inventory without using Algorithm

1's value function: If an instance has costs but not revenue in the objective of (2.3) ($r = 0$), then it is always better to have *less* inventory at the suppliers and *more* inventory at the consumers, because if a supplier has more inventory, its product must be picked up sooner (which entails a cost), and similarly, if a consumer has less inventory, its product must be dropped off sooner (which again entails a cost).

Table 6 details the results of the third experiment, using base instances with no revenue. The naming convention for column INST is the same as for Table 4. Columns 5P OBJ and TRAD OBJ respectively show the objective value of each method's solution (without value function). Columns 5P TIME and TRAD TIME show the time needed to find each solution, in minutes. The four columns under S INV and C INV show each solution's total ending inventory for suppliers and consumers, respectively.

Table 6: Simplified time instances, third experiment.

	OBJ		TIME (min.)		S INV		C INV	
INST	5P	TRAD	5P	TRAD	5P	TRAD	5P	TRAD
5-1	-75.6	-77.07	425	1394	1.72	2.91	2.34	2.09
5-2	-75.49	-77.39	9	1344	2.59	2.78	2.78	2.67
5-3	-63.29	-64.57	434	1417	3.68	3.68	3.56	2.57
5-4	-60.73	-78.66	0.5	1407	2.85	2.09	2.30	2.89
5-5	-76.15	-80.28	100	1242	1.93	2.66	2.54	2.54
6-1	-81.47	-84.02	95	1317	1.60	2.93	2.34	2.29
6-2	-82.55	-81.54	17	1213	2.59	3.47	2.78	2.50
6-3	-67.05	-69.38	4	1437	3.68	3.84	3.56	3.4
6-4	-66.25	-67.34	29	1365	2.68	3.67	2.3	2.18
6-5	-82.94	-83.5	0.1	1375	1.74	2.74	2.54	2.54
7-1	-81.49	-83.06	10	1432	1.71	3.19	2.39	2.38
7-2	-83.74	-83.78	14	1124	2.55	2.84	2.83	2.68
7-3	-67.96	-54.69	435	1315	3.65	4.63	3.6	2.48
7-4	-83.87	-68.25	3	1405	1.63	2.81	3.35	2.34
7-5	-83.31	-83.4	123	1428	1.82	2.43	2.59	2.58

We observe that the 5P solution method clearly outperforms the traditional approach. The average objective values are very similar (-75.46 for the 5P solution and -75.8 for the traditional solution,) with the 5P solution beating the traditional solution in all but three instances, 6-2, 7-3, and 7-4. However, the average ending inventories show a marked difference. The average total ending supplier inventory is

2.43 for 5P versus 3.11 for the traditional solution, and the analogous statistic for consumer inventory is 2.79 versus 2.54. So on average, the 5P solutions end with 22% less inventory at suppliers and 9.6% more inventory at the consumers, even though they spend less to achieve this. The last statement can be underlined by one additional fact: For all instances except 5-4, 6-2, 7-3 and 7-4, the 5P solution dominates the traditional solution, implying that the 5P solutions do more with less in most instances, and not just on the average.

As in the first experiment, the drastic difference in computation time is noteworthy. The 5P solutions are generated in an average of 113 minutes, as opposed to the traditional solutions, which take an average of 22.5 hours, close to the 24-hour limit. In fact, only for instances 5-1, 5-3 and 7-3 do the 5P times approach the eight-hour limit; in the rest of the cases the solution is found relatively quickly, at an average of only 31 minutes. As before, the shorter computation times suggest that the approximate value function can be used to generate a good solution quickly.

One of Algorithm 1's underlying goals is to compute an approximate value function that approximately satisfies the Bellman equation (2.5). Our fourth experiment tested how closely instance 5's value function satisfies the recursion by generating a set of 25 random inventories, directly calculating what their value is (as measured by the approximate value function) and then solving a single-period model with the value function. Figure 9 plots the results, with the predicted values along the x-axis and the observed values from the single-period solve in the y-axis.

As the plot shows, the approximate value function matches predicted and observed values quite closely. The geometric mean of the normalized absolute error between observed and predicted values is 7.33%, and only four points have an absolute error exceeding the average absolute error plus one standard deviation. This result indicates the PWL concave approximate value function approximates its true discontinuous analogue well for this instance, and gives further insight into our previous experiments'

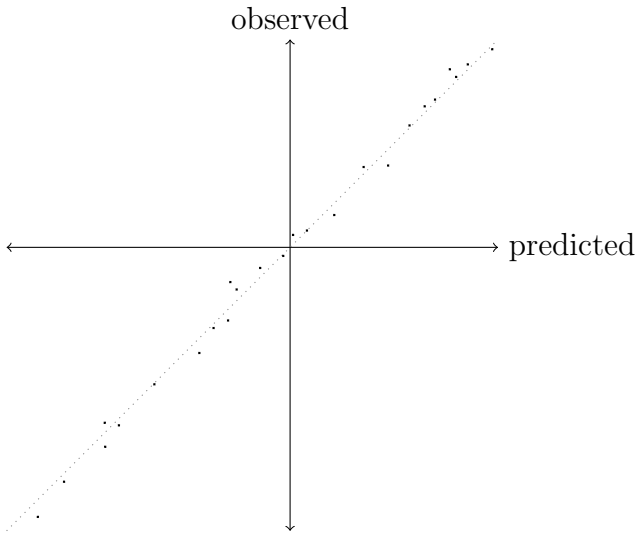

Figure 9: Plot of predicted and observed values for simple-time instance.

results.

An interesting question is whether we could gain anything by using the variable-bucket class of value functions introduced in Section 3.4. After obtaining the value function for instance 5 using fixed buckets, we reran Algorithm 1 allowing the buckets to vary. Our original intent was to compare solutions generated by each function; however, after several iterations we found the buckets almost unchanged, with none varying by more than a small fraction of the original bucket size. This outcome indicates that the original bucket choice of dividing each inventory domain into equal thirds was an appropriate one. It also suggests a second use for the variable-region version of Algorithm 1 as validation for a previously obtained approximate value function.

5.2 Variable Travel Time

In the next set of experiments, we add complexity to the previous model by allowing travel time to vary depending on the pair of ports involved. In addition to the parameters already defined, we have a travel time $\theta_a \in \mathbb{Z}_+$ for $a \in A$ indicating the

number of periods it takes to travel along a, and a maximum voyage length $\Theta \in \mathbb{Z}_+$. Vessels may now stay at a port for a period in the middle of a voyage, incurring a *demurrage* cost $f_d > 0$. We assume a ship cannot be redirected once a voyage is scheduled, and multiple ships may visit a port in the same period.

For a planning horizon $\mathcal{T}$, let $G_{\mathcal{T}}$ denote the time-expanded network constructed from G, Θ and T by creating $T + \Theta$ copies of each port $i \in S \cup C$, and for any $t \in \{1, \ldots, T + \Theta\}$ connecting node (i, t) with node $(j, t + \tau)$ if and only if

i) $(i, j) \in A$, $\tau = \theta_{ij}$ and $t + \tau \leq T + \Theta$, or

ii) $i = j$, $\tau = 1$ and $t \leq T + \Theta - 1$.

The first type of arcs corresponds to actual travel, while the second type corresponds to demurrage. Let $A_{\mathcal{T}}$ denote the arc set of this network, and $A_t \subseteq A_{\mathcal{T}}$ denote the subset of arcs that a vessel starting a voyage on period t can traverse, for $t \in \mathcal{T}$. Specifically,

$$A_t = \{((i, \tau_1), (j, \tau_2)) \in A_{\mathcal{T}} : \tau_1 \geq t, \tau_2 \leq t + \Theta\}.$$

For $t \in \mathcal{T}$ and $a \in A_t$, let x_a^t indicate whether the voyage starting in period t traverses arc $a \in A_t$. Let y_i^t indicate whether the voyage starting in t begins at supplier $i \in S$, and let $y_j^{t,\tau}$ denote whether the voyage starting in t ends at consumer $j \in C$ in period $t + \tau$. Let z_a^t denote the amount of product on board while traversing arc $a \in A_t$ on the vessel that started in t. Let $q_i^{t,\tau}$ be the amount of product picked up from supplier $i \in S$ in period $t + \tau$ by the vessel that started in t, and define $q_j^{t,\tau}$ analogously for $j \in C$. Let s_i^t be the amount of product at port $i \in S \cup C$ at the end of t. We use the notation $\delta_t^+(i, t')$ to denote the set of outgoing arcs of (i, t') contained in A_t, and a similar definition for δ_t^-.

The variable-time maritime IRP is given by

$$\max \sum_{t \in \mathcal{T}} \sum_{\tau=0}^{\Theta} \gamma^t \left(\sum_{i \in S \cup C} r_i q_i^{t,\tau} - \sum_{a \in A_t} f_a x_a^t \right) \tag{5.1a}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{a\in\delta_t^+(i,t)} x_a^t - \sum_{a\in\delta_t^-(i,t)} x_a^t = y_i^t, \quad i\in S, t\in\mathcal{T} \\
& \sum_{a\in\delta_t^+(i,t+\tau)} x_a^t - \sum_{a\in\delta_t^-(i,t+\tau)} x_a^t = 0, \quad i\in S, t\in\mathcal{T}, 0\le\tau\le\Theta \qquad (5.1b) \\
& \sum_{a\in\delta_t^-(j,t+\tau)} x_a^t - \sum_{a\in\delta_t^+(j,t+\tau)} x_a^t = y_j^{t,\tau}, \quad j\in C, t\in\mathcal{T}, 0\le\tau\le\Theta \\
& \sum_{i\in S} y_i^t \le 1, \quad t\in\mathcal{T} \\
& \sum_{j\in C}\sum_{\tau=0}^{\Theta} y_j^{t,\tau} \le 1, \quad t\in\mathcal{T} \qquad (5.1c) \\
& z_a^t \le x_a^t, \quad a\in A_t, t\in\mathcal{T} \qquad (5.1d) \\
& \sum_{a\in\delta_t^+(i,t+\tau)} z_a^t - \sum_{a\in\delta_t^-(i,t+\tau)} z_a^t = q_i^{t,\tau}, \quad i\in S, t\in\mathcal{T}, 0\le\tau\le\Theta \\
& \sum_{a\in\delta_t^-(j,t+\tau)} z_a^t - \sum_{a\in\delta_t^+(j,t+\tau)} z_a^t = q_j^{t,\tau}, \quad j\in C, t\in\mathcal{T}, 0\le\tau\le\Theta \qquad (5.1e) \\
& s_i^t = s_i^{t-1} + w_i - \sum_{\tau=\max\{0,t-T\}}^{\min\{\Theta,t\}} q_i^{t-\tau,\tau}, \quad i\in S, t\in\mathcal{T}\cup\{T+\tau\}_{\tau=1}^{\Theta} \\
& s_j^t = s_j^{t-1} - w_j + \sum_{\tau=\max\{0,t-T\}}^{\min\{\Theta,t\}} q_j^{t-\tau,\tau}, \quad j\in C, t\in\mathcal{T}\cup\{T+\tau\}_{\tau=1}^{\Theta} \qquad (5.1f) \\
& s_i^t \in [0, d_i], \quad i\in S\cup C, t\in\mathcal{T} \\
& s_i^t \ge 0, \quad i\in S, t\in\{T+\tau\}_{\tau=1}^{\Theta} \qquad (5.1g) \\
& s_j^t \le d_j, \quad j\in C, t\in\{T+\tau\}_{\tau=1}^{\Theta} \\
& x_a^t \in \{0,1\}, \quad a\in A_{\theta,t}, t\in\mathcal{T} \\
& y_i^t \in \{0,1\}, \quad i\in S, t\in\mathcal{T} \\
& y_j^{t,\tau} \in \{0,1\}, \quad j\in C, t\in\mathcal{T}, 0\le\tau\le\Theta \qquad (5.1h) \\
& z_a^t \ge 0, \quad a\in A_{\theta,t}, t\in\mathcal{T} \\
& q_i^{t,\tau} \ge 0, \quad i\in S, t\in\mathcal{T}, 0\le\tau\le\Theta \\
& q_j^{t,\tau} \ge 0, \quad j\in C, t\in\mathcal{T}, 0\le\tau\le\Theta
\end{aligned}
$$

In this formulation, the objective (5.1a) maximizes the difference between revenue and transportation cost, where we set $f_a = f_d$ for demurrage arcs. We assume that all transactions related to a voyage take place on the voyage's first period, implying all revenues and costs from a voyage starting in t are discounted by γ^t; however, this assumption may be altered if necessary. Constraints (5.1b) enforce flow balance on the routing variables. Constraints (5.1c) enforce a limit of at most one voyage per period; note that the second set of inequalities is redundant and implied by the first. The variable upper bounds (5.1d) ensure product flows on an arc only if a vessel traverses it. Constraints (5.1e) enforce product flow balance, while (5.1f) track inventory levels across periods. Finally, (5.1g) establishes bounds for the inventory state variables and (5.1h) establishes other variable bounds. Note that (5.1g) enforces only lower bounds for suppliers and upper bounds for consumers in periods after T.

In this new model, the state of the system is not only the inventory currently on hand, but also the previously pickups and dropoffs $\Theta - 1$ periods into the future. Specifically, in addition to the initial inventory s_i^0 at each port, we have the scheduled quantities $q_i^{0,\tau}$ for $\tau = 1, \ldots, \Theta - 1$. This implies that the dimension of the state space increases by a factor of Θ, which increases the computational challenge of these models.

The experimental instances were generated in a similar fashion to the previous section, with travel times θ also satisfying the triangle inequality. All test instances had three suppliers, four consumers, maximum voyage length $\Theta = 3$, and expensive slack variables were added to (5.1f) to ensure feasibility. The value function class we chose was an extension of the class used in the previous section: Each port's inventory domain was subdivided into three buckets of equal length, and future deliveries $q_i^{0,\tau}$ were assigned a single linear value coefficient (i.e. one bucket).

In the first experiment for the time-expanded instances, we repeated the first experiment from the previous section. We generated four instances of (5.1) and five

random starting inventory vectors, for a total of 20 problem instances. We generated solutions using methods 1P and 60P, explained above. Table 7 details the results, using a similar display convention. The INST column denotes the instance-inventory pair, with letters replacing numbers to emphasize the change to model (5.1). For each solution method, we display the solution's objective value, separated into contribution from the solution in (5.1a) (column OBJ. − EV) and contribution from the ending inventory's value (EV), and then totalled in a third column, OBJ. We report the time to generate solutions via method 60P in an additional column, but we do not report solution times for 1P as all solutions were generated within two seconds. The final BOUND column reports the best bound obtained by CPLEX during the computation for method 60P.

Table 7: Variable time instances, first experiment.

	1P			60P				
INST	OBJ. − EV	EV	OBJ.	OBJ. − EV	EV	OBJ.	TIME (sec.)	BOUND
A-1	-149.70	10.72	-138.98	-141.00	9.54	-131.46	3229	-95.85
A-2	-145.50	9.97	-135.53	-147.50	16.69	-130.81	3539	-96.96
A-3	-157.70	16.54	-141.16	-132.70	3.55	-129.15	3538	-99.40
A-4	-151.50	8.73	-142.77	-157.70	9.83	-147.87	1599	-96.84
A-5	-145.00	9.59	-135.41	-153.90	11.37	-142.53	3232	-98.11
B-1	-15.13	2.75	-12.38	-11.46	3.08	-8.38	3466	-6.91
B-2	-13.69	2.56	-11.13	-11.04	2.70	-8.34	2867	-6.72
B-3	-15.06	2.33	-12.73	-11.89	2.82	-9.07	3496	-7.09
B-4	-13.92	2.75	-11.17	-12.10	3.14	-8.96	3392	-7.08
B-5	-14.05	2.37	-11.68	-11.22	2.54	-8.68	3386	-7.12
C-1	-17.66	1.64	-16.02	-11.88	1.76	-10.12	2901	-7.99
C-2	-16.88	1.54	-15.34	-11.75	1.57	-10.18	3265	-8.06
C-3	-16.39	1.50	-14.89	-11.44	1.33	-10.11	3275	-8.13
C-4	-15.63	1.34	-14.29	-11.65	1.98	-9.67	3581	-8.15
C-5	-16.88	1.58	-15.30	-11.38	1.45	-9.93	2021	-8.34
D-1	-9.42	2.14	-7.28	-8.11	2.58	-5.53	417	-4.37
D-2	-9.81	2.03	-7.78	-8.11	2.59	-5.52	3527	-4.42
D-3	-9.47	1.83	-7.64	-7.94	2.19	-5.75	2818	-4.26
D-4	-9.60	1.62	-7.98	-8.17	2.39	-5.78	2976	-4.37
D-5	-10.42	1.68	-8.74	-8.21	2.45	-5.76	2303	-4.75

In this case, while the heuristic solutions remain competitive, the results are more mixed. Ending value is on average 6% higher than the benchmark solution's value, but the heuristic solution's "actual" objective value in (5.1a) is 23% more expensive.

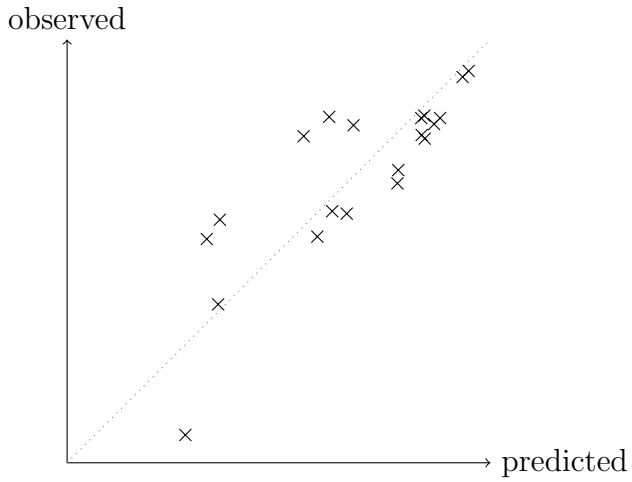

Figure 10: Plot of predicted and observed values for time-expanded instance.

In addition, only the solutions for instances A-4 and A-5 beat the benchmark in terms of the total objective (Obj). Nonetheless, the benchmark solutions still take an average of 49 minutes to generate (with all but one taking over 26 minutes), while the heuristic solutions can be generated instantly.

For our second experiment, we repeated the previous section's fourth experiment to investigate how closely the approximate value function satisfies Bellman's equation (2.5). Using instance A from the previous experiment, we generated 20 random inventory vectors. For each one, we calculated the approximate value as measured by the value function, and then solved a single-period instance of (5.1). Figure 10 plots the results, with the predicted values along the x-axis and the observed values from the single-period solve in the y-axis.

As the plot shows, the approximate value function again matches predicted and observed values quite closely. The geometric mean of the normalized absolute error between observed and predicted values is 10.8%, with only three points above 50% and five above 30%. In six of the cases, the observed value is larger than the predicted value, with the reverse holding in the remaining cases.

Notwithstanding the second experiment's results, there is clearly a difference in

quality between solutions for the time-expanded model (5.1) and those generated for the simpler model (2.3). Allowing the bucket lengths to vary in Algorithm 1 also did not help, as it once again validated the previously chosen buckets. We believe the quality decrease stems from an underlying structural issue: Not only do variable travel times increase the state space dimension by an order of magnitude, they also fundamentally alter how the states are related. For example, the value of inventory scheduled for delivery in one period should depend on the current inventory level. This complicated interaction suggests the use of more complex basis functions (cf. [55]) within the algorithmic framework. Since time-expanded networks are present in many practical applications, further research in this vein should prove valuable in practical as well as theoretical terms.

Another important issue is the premium in solution quality obtained by using PWL concave value functions instead of simpler affine functions. As a simple test, we computed an approximate value function for instance A with simple linear coefficients for each state variable $s_i^0, q_i^{0,\tau}$, and compared solutions generated by this function to the three-bucket inventory value function used in this section's first experiment. Specifically, we generated solutions for 60-period models with the same starting inventories as that experiment, and compared the solutions' objective value in (5.1a); we did not include ending inventory value in the measure to ensure a fair comparison. As a sanity check, we also computed myopic solutions (that is, with no value function) for the same instances. Table 8 details the results; column Inst identifies the instance, and columns 3B, 1B and No VF respectively give the objective value for the three-bucket, one-bucket (linear value function) and myopic solutions.

As the table shows, there is an average cost increase of 50% between the 3B and 1B solutions, while both solutions generated with value functions incur a small fraction of the myopic solution's cost. Of course, the solutions' cost difference stems from the three-bucket value function's concavity. It would be reasonable to expect smaller

Table 8: Variable time instances, third experiment.

Inst	3B	1B	No VF
A-1	-149.70	-220.80	-19675.50
A-2	-145.50	-229.00	-19302.20
A-3	-157.70	-224.40	-19618.00
A-4	-151.50	-219.00	-19607.00
A-5	-145.00	-229.00	-20045.40

differences as the value function "flattens out," becoming less concave and more linear. In addition, there may be situations in which solutions generated by value functions with fewer breakpoints have a better objective than solutions generated by value functions with more buckets; e.g. if the breakpoint placement varies in the two value functions. Issues related to sensitivity of solution quality with respect to a value function's number of buckets (or regions in the non-separable case) are certainly important and could shed further light on our methodology.

CHAPTER VI

CONCLUSIONS

This thesis details a time decomposition framework for multi-period supply chain planning models that uses approximate dynamic programming and data fitting methodologies to construct an approximate inventory value function. The framework is generic and versatile, amenable to quite complex models such as the maritime IRP problems presented in Chapter 5. The choice of PWL concave functions as approximate inventory value functions is justified by their close relationship to maximization MIP value functions, including the infinite-horizon LSP we study in Chapter 4. Although we have had empirical success with these techniques, we also believe this work opens many questions of theoretical and practical interest.

One important question for the ADP algorithm is theoretical convergence. For example, [69] prove convergence for a fitting-based DP scheme similar but simpler than our own. However, even their proof makes certain assumptions that are wholly impossible in our case, such as the evaluation of every possible state at each iteration. Nonetheless, convergence issues could be investigated further. Another issue is the extension of the framework for stochastic variants, where the stationary parameters are replaced by known distributions, or robust variants, where they are replaced by uncertainty sets. For the stochastic case, there is a fairly developed body of work, cf. [55]; the robust case is still in its infancy, although some DP issues have received attention, e.g. [36].

The choice of PWL concave functions as value functions, motivated also by their ease of implementation within maximization MIP objectives, has influenced our investigation and development of fitting models in Chapter 3. One obvious extension is

to upper semi-continuous PWL superadditive functions, which can also be modeled within a maximization MIP's objective, although at a higher computational cost. Fitting models for PWL sub/superadditive functions would require a better characterization of sub/superadditivity, a question of fundamental importance for MIP in general because of its relation to cutting planes. A secondary issue is the improvement of fitting models for function classes with variable regions. The use of the big-M method implies weak linear relaxations and possibly long solve times. Lower bounding procedures such as those presented in [8] could be useful.

We discuss another important issue related to value functions in Chapter 5. Specifically, the sensitivity of solution quality to a value function's number of regions is important, especially as the problem's dimension grows and computation times increase. Ideally, we would like the simplest function that produces solutions of a certain quality, perhaps with some guarantee. A related but more fundamental question is whether the number of regions can be updated dynamically within the ADP algorithm: As iterations progress, we observe the evolving value function, and subdivide or merge its domain's regions based on our observations. This dynamic updating would place our algorithm closer to constraint generation, Bender's decomposition and other techniques currently used in dynamic and stochastic programming.

At the end of Chapter 4, we note that value functions obtained over fixed regions or buckets may be superior to variable-region value functions. In particular, the example showed that an approximate value function with a breakpoint aligning with the true value function's discontinuity induced a better solution than the variable-bucket approximate value function, even though the latter's fit with respect to the true function was more accurate. An interesting follow-up question is whether we can discover discontinuities in a problem's true value function and use these to plan for the structure of the approximate value function.

In the same chapter we detail the value function of an simple infinite MIP, the

single-item LSP. The result has obvious extensions to general MIP and DP, in addition to variants such as the average cost optimality criterion, stochastic and robust MIP and DP, etc. We believe that a more general result in this vein, such as the characterization of infinite MIP value functions under various optimality criteria, would illuminate connections between MIP and DP that researchers have studied since the beginning of operations research. Moreover, if fully developed, any result on the connection of infinite MIP's and DP's could extend these fields' applicability to a new class of models.

REFERENCES

[1] Adelman, D., "A Price-Directed Approach to Stochastic Inventory/Routing," *Operations Research*, vol. 52, pp. 499–514, 2004.

[2] Applegate, D., Bixby, R., Chvátal, V., and Cook, W., *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[3] Balas, E., "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Applied Mathematics*, vol. 89, pp. 3–44, 1998.

[4] Bean, J., Smith, R., and Yano, C., "Forecast horizons for the discounted dynamic lot-size problem allowing speculative motive," *Naval Research Logistics*, vol. 34, pp. 761–774, 1987.

[5] Bellman, R. and Dreyfus, S., "Functional Approximation and Dynamic Programming," *Mathematical Tables and Other Aids to Computation*, vol. 13, pp. 247–251, 1959.

[6] Bertsekas, D. and Tsitsiklis, J., *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[7] Bertsimas, D. and Shioda, R., "Classification and Regression via Integer Optimization," *Operations Research*, vol. 55, pp. 252–271, 2007.

[8] Bienstock, D., "Eigenvalue techniques for proving bounds for convex objective, nonconvex programs." Working paper, 2009.

[9] Blair, C., "A closed-form representation of mixed-integer program value functions," *Mathematical Programming*, vol. 71, pp. 127–136, 1995.

[10] Blair, C. and Jeroslow, R., "The value function of a mixed integer program: I," *Discrete Mathematics*, vol. 19, pp. 121–138, 1977.

[11] Blair, C. and Jeroslow, R., "The value function of a mixed integer program: II," *Discrete Applied Mathematics*, vol. 25, pp. 7–19, 1979.

[12] Blair, C. and Jeroslow, R., "The value function of an integer program," *Mathematical Programming*, vol. 23, pp. 237–273, 1982.

[13] Blair, C. and Jeroslow, R., "Constructive characterizations of the value-function of a mixed-integer program I," *Discrete Applied Mathematics*, vol. 9, pp. 217–233, 1984.

[14] Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, 2004.

[15] Brahimi, N., Dauzere-Peres, S., Najid, N., and Nordli, A., "Single item lot sizing problems," *European Journal of Operational Research*, vol. 168, pp. 1–16, 2006.

[16] Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M., "The Inventory Routing Problem," in *Fleet Management and Logistics* (Crainic, T. G. and Laporte, G., eds.), ch. 4, pp. 95–114, Springer, 1998.

[17] Campbell, A. and Savelsbergh, M., "A Decomposition Approach for the Inventory-Routing Problem," *Transportation Science*, vol. 38, pp. 488–502, 2004.

[18] Carnicer, J. and Floater, M., "Piecewise linear interpolants to Lagrange and Hermite convex scattered data," *Numerical Algorithms*, vol. 13, pp. 345–364, 1996.

[19] Christiansen, M., "Decomposition of a Combined Inventory and Time Constrained Ship Routing Problem," *Transportation Science*, vol. 33, pp. 3–16, 1999.

[20] Christiansen, M. and Nygreen, B., "A method for solving ship routing problems with inventory constraints," *Annals of Operations Research*, vol. 81, pp. 357–378, 1998.

[21] Christiansen, M. and Nygreen, B., "Modelling path flows for a combined ship routing and inventory management problem," *Annals of Operations Research*, vol. 82, pp. 391–412, 1998.

[22] Cordeau, J.-F., Laporte, G., Savelsbergh, M., and Vigo, D., "Vehicle Routing," in *Handbook in Operations Research and Management Science: Transportation, Volume 14* (Barnhart, C. and Laporte, G., eds.), ch. 6, pp. 367–428, Elsevier, 2007.

[23] Dawande, M., Gavirneni, S., Naranpanawe, S., and Sethi, S., "Forecast Horizons for a Class of Dynamic Lot-Size Problems Under Discrete Future Demand," *Operations Research*, vol. 55, pp. 688–702, 2007.

[24] Desaulniers, G., Desrosiers, J., and Solomon, M., eds., *Column Generation*. Springer, 2005.

[25] Engineer, F., *Shortest path based column generation for integer programming*. PhD thesis, Georgia Institute of Technology, 2009.

[26] Erlenkotter, D., "Ford Whitman Harris and the Economic Order Quantity Model," *Operations Research*, vol. 38, pp. 937–946, 1990.

[27] Federgruen, A. and Tzur, M., "The dynamic lot-sizing model with backlogging: A simple $O(n \log n)$ algorithm and minimal forecast horizon procedure," *Naval Research Logistics*, vol. 40, pp. 459–478, 1993.

[28] Ferrari-Trecate, G., Muselli, M., Liberati, D., and Morari, M., "A learning algorithm for piecewise linear regression," in *Neural Nets: WIRN VIETRI-01, 12th Italian Workshop on Neural Nets* (Marinaro, M. and Tagliaferri, R., eds.), Springer, 2001.

[29] Fisher, M., Ramdas, K., and Zheng, Y.-S., "Ending Inventory Valuation in Multiperiod Production Scheduling," *Management Science*, vol. 47, pp. 679–692, 2001.

[30] Furman, K., Song, J.-H., Kocis, G., McDonald, M., and Warrick, P., "Feedstock Routing in the ExxonMobil Downstream Sector." To appear in *Interfaces*, 2010.

[31] Ghate, A. and Smith, R., "Optimal Backlogging Over an Infinite Horizon Under Time Varying Convex Production and Inventory Costs," *Manufacturing and Service Operations Management*, vol. 11, pp. 362–368, 2009.

[32] Graves, S. and Orlin, J., "The Infinite-Horizon Dynamic Lot-Size Problem with Cyclic Demand and Costs," Tech. Rep. OR 101-80, Operations Research Center, Massachusetts Institute of Technology, 1980. Available on-line at http://hdl.handle.net/1721.1/5365.

[33] Harris, F., "How many parts to make at once," *Operations Research*, vol. 38, pp. 947–950, 1990. Reprinted from *Factory, The Magazine of Management*, vol. 10, pp. 135–136, 1913.

[34] Hewitt, M., Nemhauser, G., and Savelsbergh, M., "Combining Exact and Heuristic Approaches for the Capacitated Fixed Charge Network Flow Problem," *INFORMS Journal on Computing*, vol. 22, pp. 314–325, 2010.

[35] Holmes, C. and Mallick, B., "Bayesian regression with multivariate linear splines," *Journal of the Royal Statistical Society, Series B*, vol. 63, pp. 3–17, 1999.

[36] Iyengar, G., "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30, pp. 1–21, 2005.

[37] Jeroslow, R. and Lowe, J., "Modeling with integer variables," *Mathematical Programming Study*, vol. 22, pp. 167–184, 1984.

[38] Johnson, E., "On the group problem for mixed integer programming," *Mathematical Programming Studies*, vol. 2, pp. 137–179.

[39] Kimms, A., "Stability Measures for Rolling Schedules with Applications to Capacity Expansion Planning, Master Production Scheduling, and Lot Sizing," *Omega*, vol. 26, pp. 355–366, 1998.

[40] Kleywegt, A., Nori, V., and Savelsbergh, M., "The Stochastic Inventory Routing Problem with Direct Deliveries," *Transportation Science*, vol. 36, pp. 94–118, 2002.

[41] Kleywegt, A., Nori, V., and Savelsbergh, M., "Dynamic Programming Approximations for a Stochastic Inventory Routing Problem," *Transportation Science*, vol. 38, pp. 42–70, 2004.

[42] Lau, K.-N., Leung, P.-L., and Tse, K.-K., "A mathematical programming approach to clusterwise regression model and its extensions," *European Journal of Operational Research*, vol. 116, pp. 640–652, 1999.

[43] Longstaff, F. and Schwartz, E., "Valuing American Options by Simulation: A Simple Least-Squares Approach," *The Review of Financial Studies*, vol. 14, pp. 113–147, 2001.

[44] Magnani, A. and Boyd, S., "Convex Piecewise-Linear Fitting," *Optimization and Engineering*, vol. 10, pp. 1–17, 2009.

[45] Manne, A., "Programming of economic lot sizes," *Management Science*, vol. 4, pp. 115–135, 1958.

[46] McClain, J., "Dynamics of exponential smoothing with trend and seasonal terms," *Management Science*, vol. 20, pp. 1300–1304, 1974.

[47] Michel, S. and Vanderbeck, F., "A Column Generation based Tactical Planning Method for Inventory Routing," Tech. Rep. 00169311, INRIA Bordeaux Sud-Ouest, team RealOpt, 2008. Available on-line at http://hal.inria.fr/docs/00/33/90/37/PDF/techRepR2.pdf.

[48] Minkoff, A., "A Markov decision model and decomposition heuristic for dynamic vehicle dispatching," *Operations Research*, vol. 41, pp. 77–90, 1993.

[49] Moin, N. and Salhi, S., "Inventory routing problems: a logistical overview," *Journal of the Operational Research Society*, vol. 58, pp. 1185–1194, 2007.

[50] Nananukul, N., *Lot-Sizing and Inventory Routing for a Production-Distribution Supply Chain*. PhD thesis, The University of Texas at Austin, 2008.

[51] Nemhauser, G. and Wolsey, L., *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1999.

[52] Pardalos, P. and Kundakcioglu, O., "Classification via Mathematical Programming (Survey)," *Applied and Computational Mathematics*, vol. 8, pp. 23–35, 2009.

[53] Pochet, Y. and Wolsey, L., *Production Planning by Mixed Integer Programming*. Springer, 2006.

[54] Pottmann, H., Krasauskas, R., Hamann, B., Joy, K., and Seibold, W., "On Piecewise Linear Approximation of Quadratic Functions," *Journal for Geometry and Graphics*, vol. 4, pp. 31–53, 2000.

[55] Powell, W., *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, Inc., 2007.

[56] Puterman, M., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 2005.

[57] Queyranne, M. and Wang, Y., "On the convex hull of feasible solutions to certain combinatorial problems," *Operations Research Letters*, vol. 11, pp. 1–11, 1992.

[58] Rockafellar, R., *Convex Analysis*. Princeton University Press, 1997.

[59] Romeijn, H., Sharma, D., and Smith, R., "Extreme Point Characterizations for Infinite Network Flow Problems," *Networks*, vol. 48, pp. 209–222, 2006.

[60] Romeijn, H. and Smith, R., "Shadow Prices in Infinite Dimensional Linear Programming," *Mathematics of Operations Research*, vol. 23, pp. 239–256, 1998.

[61] Ruppert, D., Wand, M., and Carroll, R., *Semiparametric Regression*. Cambridge University Press, 2003.

[62] Savelsbergh, M. and Song, J.-H., "Inventory Routing with Continuous Moves," *Computers and Operations Research*, vol. 34, pp. 1744–1763, 2007.

[63] Savelsbergh, M. and Song, J.-H., "An Optimization Algorithm for the Inventory Routing Problem with Continuous Moves," *Computers and Operations Research*, vol. 35, pp. 2266–2282, 2008.

[64] Schochetman, I. and Smith, R., "Infinite Horizon Optimality Criteria for Equipment Replacement under Technological Change," *Operations Research Letters*, vol. 35, pp. 485–492, 2007.

[65] Smith, R. and Zhang, R., "Infinite Horizon Production Planning in Time-Varying Systems with Convex Production and Inventory Costs," *Management Science*, vol. 44, pp. 1313–1320, 1998.

[66] Strikholm, B., "Determining the number of breaks in a piecewise linear regression model," tech. rep., Department of Economic Statistics and Decision Support, Stockholm School of Economics, 2006. SSE/EFI Working Paper Series in Economics and Finance, No. 648.

[67] Topaloglu, H. and Powell, W., "An Algorithm for Approximating Piecewise Linear Concave Functions from Sample Gradients," *Operations Research Letters*, vol. 31, pp. 66–76, 2003.

[68] Trick, M. and Zin, S., "Spline Approximations to Value Functions: A Linear Programming Approach," *Macroeconomic Dynamics*, vol. 1, pp. 255–277, 1997.

[69] Tsitsiklis, J. and van Roy, B., "Feature-Based Methods for Large Scale Dynamic Programming," *Machine Learning*, vol. 22, pp. 59–94, 1996.

[70] Vielma, J., Ahmed, S., and Nemhauser, G., "Mixed-Integer Models for Nonseparable Piecewise Linear Optimization: Unifying Framework and Extensions," *Operations Research*, vol. 58, pp. 303–315, 2010.

[71] Wagner, H. and Whitin, T., "Dynamic version of the economic lot size model," *Management Science*, vol. 5, pp. 89–96, 1958.

[72] Wolsey, L., "The b-hull of an integer program," *Discrete Applied Mathematics*, vol. 3, pp. 193–201, 1981.

[73] Wolsey, L., "Progress with single-item lot-sizing," *Journal of Operational Research*, vol. 86, pp. 395–401, 1995.