

Approximate Dynamic Programs for Natural Gas Storage Valuation Based on Approximate Linear Programming Relaxations

Selvaprabu Nadarajah, François Margot, Nicola Secomandi

Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213

Tepper working paper 2011-E5

Initial version: April 5, 2011

Abstract

The real option valuation of natural gas storage contracts with a high dimensional model of the natural gas forward curve dynamics is an important practical problem that gives rise to an intractable Markov Decision Process (MDP). We propose a novel family of Approximate Dynamic Programs (ADPs) for this MDP based on partitioned surrogate relaxations of an approximate linear program. Used in conjunction with the information relaxation and duality approach for MDPs, our ADPs estimate lower and upper bounds to the value of storage within a Monte Carlo simulation of the natural gas forward curve. On a set of benchmark instances available in the literature, our estimated lower bounds essentially match the best known lower bound estimates, and our estimated upper bounds either match or improve those available in the literature, reducing the best estimated optimality gap to less than 2.3% on all these instances. Further, our approach subsumes the ADP available in the literature corresponding to these known estimated bounds, and sheds additional light on the performance of heuristics used in practice. Our research has broader relevance for the valuation of storage of other commodities, and the modeling and solution by approximate linear programming of intractable MDPs with state variables that include high-dimensional exogenous information, such as a forward curve.

1 Introduction

Storage plays an important role in matching the supply and demand for storable commodities, including natural gas. Natural gas merchants need quantitative methods to value and manage natural storage contracts (Maragos 2002). In theory, the value of such a storage contract is the risk neutral value of the cash flows of an optimal storage policy (Dixit and Pindyck 1994, Trigeorgis 1996). This value arises from the seasonality and volatility in the natural gas spot price and forward curve (the collection of prices of contracts for delivery of natural gas at different times in the future). The value of seasonality is not difficult to capture, as it can be computed by solving a deterministic linear program. In contrast, capturing the value due to price volatility, using high dimensional models of the natural gas forward curve, gives rise to an intractable Markov Decision Problem

(MDP). This problem can be approached by computing bounds on its optimal value function, as well as a suboptimal policy (Lai et al. 2010, abbreviated to LMS in the rest of this paper, and Maragos 2002).

LMS estimates lower bounds by evaluating a heuristic policy using Monte Carlo simulation, and upper bounds through the information relaxation and duality approach discussed by Brown et al. (2010, see references therein for earlier related work). LMS compute these upper and lower bound estimates based on a grid-based, that is, piecewise constant in the state, value function approximation obtained through an approximate dynamic programming approach. Their value function approximation includes a single price, the spot price. Their heuristic policies are near optimal for most of their benchmark instances, namely, their Spring, Summer, and Fall instances. However, their optimality gap in some of their Winter instances is relatively large, ranging from 3.3% to 5.1%.

In theory, one could attempt to close this optimality gap by directly extending the value function approximation of LMS to also include the prompt month futures price. However, this extension is computationally burdensome in the context of their model. We thus perform this extension using a different modeling approach. Our starting point is approximate linear programming for the heuristic solution of intractable MDPs (Schweitzer and Seidmann 1985), which is based on approximating the MDP optimal value function with a lower dimensional representation. Analogous to LMS, we use grid-based value function approximations, that is, look-up tables.

Each state in our MDP includes endogenous and exogenous information. The natural gas inventory level, which is controlled by the merchant, is our endogenous information. The natural gas forward curve, which is governed by external processes, is our exogenous information. Our grid-based value function approximations rely on a lower dimensional representation of this exogenous information; that is, at most they include the spot price and the prompt month futures price. Our observation is that using approximate linear programming in conjunction with such a grid-based value function approximation yields poor bounds on the value of storage. We observe that this behavior is caused by binding constraints in the Approximate Linear Program (ALP), which determine the value function approximation, corresponding to futures prices in the right tail of their distribution; that is, prices that are unlikely to occur.

We address this issue by proposing a constraint relaxation scheme that we label Partitioned

Surrogate Relaxation (PSR). PSR is our main methodological contribution. PSR partitions the ALP constraint set and replaces each set in this partition by a surrogate constraint, created by taking a positive linear combination of the constraints in this set. The multipliers used to derive each surrogate constraint assign different importance to different futures prices; that is, different elements of the exogenous information set. We propose an intuitively appealing selection of this partition and multipliers in the context of two grid-based value function approximations: One that includes only the spot price as part of its representation of the exogenous information, as in LMS, and one that extends it by also including the prompt month futures price.

We show that these selections of the PSR partition and multipliers give rise to ALPs that admit equivalent MDP representations. These MDPs can be efficiently solved by backward recursion (also exploiting the base-stock structure of their optimal policies; Secomandi 2009). In particular, this allows us to efficiently deal with the prompt month futures price extension of our single price value function approximation, and is also useful when sequentially reoptimizing these MDPs along a price sample path in the Monte Carlo simulation used for lower bound estimation (this reoptimization typically yields improved lower bounds).

Compared to the single price value function approximation of LMS, ours yields comparable upper bound estimates and marginally weaker lower bound estimates on their instances, but does so with a substantially reduced computational effort. Our two price value function approximation without reoptimization generates lower bound estimates that are comparable or better than those obtained by these authors, with improvements of up to 7.2% in the Winter instances. This indicates that this Approximate Dynamic Program (ADP) is a better approximation to the exact storage valuation MDP than are LMS and our single price ADP. More importantly, this approximation allows us to further improve by up to 2.8% some estimated upper bounds in their Winter instances. Further, reoptimization improves by a smaller amount the lower bound estimates obtained with our two price value function approximation than it improves the LMS and our estimated lower bounds based on a single price value function approximation (but, of course, reoptimizing our two price model is slower than reoptimizing a single price model).

Overall, by using reoptimized lower bounds and our improved upper bound estimates we reduce the optimality gap estimate in the Winter instances to within 2.3%. This result has managerial relevance. The rolling intrinsic policy is widespread in practice (LMS and references therein), and

LMS show that their reoptimized lower bound estimates either match or improve on those obtained with this policy. Our improved upper bound estimates confirm that this policy is also near optimal on the Winter instances, for which its estimated optimality gap is reduced to less than 5% from the previously known 7.5%.

We also explore the connection between our PSR approach and the ADP of LMS, and show that their ADP can be derived from a specific PSR. That is, our PSR approach subsumes their ADP. This sheds novel light on the LMS ADP, and illustrates the generality of our PSR approach. Indeed, other ADPs could be obtained by using alternative partitions and multipliers.

Our work is relevant for the valuation of the real option to store other commodities, as well as the solution by approximate linear programming of intractable MDPs whose state variables include high dimensional exogenous information.

The remainder of this paper is organized as follows. We review the extant literature in §2. We introduce the storage valuation model and the bounding approach that we use in §3. In §4, we present an ALP, our PSR approach based on this ALP, and alternative ADPs obtained from different PSRs of this ALP. We discuss numerical implementation and complexity issues in §5. We present our numerical results in §6 and conclude in §7.

2 Literature Review

Approximate dynamic programming has received substantial attention in the recent literature. Bertsekas and Tsitsiklis (1996), Trick and Zin (1997), Chang et al. (2007), Adelman (2006), Powell (2007), and Van Roy (2002) are excellent sources on this topic. Schweitzer and Seidmann (1985) introduce the approximate linear programming approach to approximate dynamic programming, and de Farias and Van Roy (2001, 2003, 2006) analyze it. Applications of this approach include Adelman (2004) and Adelman and Klabjan (2011) in inventory control; Adelman (2007), Farias and Van Roy (2007), and Zhang and Adelman (2009) in revenue management; and Morrison and Kumar (1999), de Farias and Van Roy (2001, 2003), Veatch (2005), and Moallemi et al. (2008) in queuing. The novelty of our work relative to this literature is two fold. The first is the presence of exogenous information in the state of the natural gas storage MDP that we consider, whereas this type of information is absent in most of the models studied in the extant approximate linear

programming literature. The second is our development and use of the PSR approach to deal with the difficulties brought about by using a lower dimensional representation of this information in an ALP. Although surrogate relaxation is not new, see, e.g., Glover (1968, 1975), its use in an approximate linear programming context is novel.

The use of constraint relaxations in approximate linear programming is relatively new and the literature is scant. Desai et al. (2009) and Petrik and Zilberstein (2009) use constraint relaxation to improve the value function approximation obtained by solving an ALP. Desai et al. (2009) consider a relaxation of an ALP by allowing budgeted violation of constraints, and successfully apply it to the game of Tetris. Petrik and Zilberstein (2009) propose a relaxation method similar to that of Desai et al. (2009) where violated constraints are penalized in the objective function. Our surrogate relaxation approach is different from those used in both of these papers.

As in LMS, we use the information relaxation and duality approach discussed by Brown et al. (2010), which generalizes earlier work by Haugh and Kogan (2004) and Rogers (2002), and consider grid based value function approximations. However, our approach is more general than that of LMS since it subsumes their ADP. We also introduce two new ADPs that essentially match or improve on the bounds obtained by LMS, thus adding to the literature on natural gas storage valuation (Chen and Forsyth 2007, Boogert and De Jong 2008, Thompson et al. 2009, Carmona and Ludkovski 2010, Secomandi 2009, Secomandi et al. 2010, Wu et al. 2010).

3 Storage Valuation Approach

This section, in part based on LMS, presents the storage valuation model and bounding approach used in this paper.

3.1 Valuation Model

A natural gas storage contract provides a merchant with the option to purchase and inject, store, and withdraw and sell natural gas during a predetermined finite time horizon, while respecting injection and withdrawal capacity limits, as well as inventory constraints. The merchant's goal is to maximize the value of the storage contract. We model this valuation problem as an MDP. Purchases and injections, and withdrawals and sales give rise to cash flows. The storage contract

has N possible dates with cash flows. The i -th cash flow occurs at time T_i , $i \in \mathcal{I} := \{0, \dots, N-1\}$. Each such time is also the maturity of a futures contract. We denote the futures price at time T_i of a contract maturing at time T_j , $j \geq i$, as $F_{i,j}$. The forward curve is the collection of futures prices $F_i := (F_{i,j}, i \in \mathcal{I}, j \geq i)$. We adopt the convention $F_N \equiv 0$. We also define $F'_i := (F_{i,j}, i \in \mathcal{I}, j > i)$, $\forall i \in \mathcal{I}$, for notational convenience. At a given maturity, the decision maker can purchase and inject or withdraw and sell natural gas, or do nothing.

The set of feasible inventory levels is $\mathcal{X} := [0, \bar{x}]$, where 0 and $\bar{x} \in \mathbb{R}_+$ represent the minimum and maximum inventory levels, respectively. The (absolute) values of the injection capacity C^I (< 0) and withdrawal capacity C^W (> 0) represent the maximum amounts that can be injected and withdrawn in between two successive trading times, respectively. An action a corresponds to an inventory change during this time period. A positive action represents a withdrawal and sell decision, a negative action a purchase and inject decision, and the zero action is the do nothing decision. The set of feasible injections, withdrawals, and overall actions are $\mathcal{A}^I(x) := [C^I \vee (x - \bar{x}), 0]$, $\mathcal{A}^W(x) := [0, x \wedge C^W]$, and $\mathcal{A}(x) := \mathcal{A}^I(x) \cup \mathcal{A}^W(x)$, respectively, where $\cdot \wedge \cdot \equiv \min\{\cdot, \cdot\}$ and $\cdot \vee \cdot \equiv \max\{\cdot, \cdot\}$.

The immediate reward from action a at time T_i is $r(a, s_i)$, where $s_i \equiv F_{i,i}$ is the spot price at this time. The coefficients $\alpha^W \in (0, 1]$ and $\alpha^I \geq 1$ model fuel losses associated with withdrawals and injections, respectively. The coefficients c^W and c^I represent withdrawal and injection marginal costs. The expression for the immediate reward is

$$r(a, s) := \begin{cases} (\alpha^I s + c^I)a, & \text{if } a \in \mathbb{R}_-, \\ 0, & \text{if } a = 0, \\ (\alpha^W s - c^W)a, & \text{if } a \in \mathbb{R}_+. \end{cases} \quad \forall s \in \mathbb{R}_+, \quad (1)$$

The value of storage can be obtained by solving the following MDP:

$$V_N(x_N, F_N) := 0, \quad \forall x_N \in \mathcal{X}, \quad (2)$$

$$V_i(x_i, F_i) = \max_{a \in \mathcal{A}(x_i)} r(a, s_i) + \delta \mathbb{E} \left[V_{i+1} \left(x_i - a, \tilde{F}_{i+1} \right) \mid F'_i \right], \quad \forall i \in \mathcal{I}, (x_i, F_i) \in \mathcal{X} \times \mathbb{R}_+^{N-i}, \quad (3)$$

where $V_i(x_i, F_i)$ is the optimal value function in stage i and state (x_i, F_i) ; δ is the per stage risk free discount factor; and \mathbb{E} is expectation with respect to the risk neutral distribution of the random

vector \tilde{F}_{i+1} conditional on F'_i . We refer to the model (2)-(3) as the Exact Dynamic Program (EDP).

Consistent with the practice-based literature (Eydeland and Wolyniec 2003, Chapter 5, and Gray and Khandelwal 2004) we use an N -factor model of the risk neutral dynamics of the natural gas forward curve. In this model, the time t futures price with maturity T_i , $F(t, T_i)$, evolves as a driftless Brownian motion with maturity specific constant volatility $\sigma_i > 0$. The instantaneous correlation between the standard Brownian motion increments $dZ_i(t)$ and $dZ_j(t)$ corresponding to the futures prices with maturities T_i and T_j , $i \neq j$, is $\rho_{ij} \in (-1, 1)$ ($\rho_{ii} = 1$). This model is

$$\frac{dF(t, T_i)}{F(t, F_i)} = \sigma_i dZ_i(t), \quad \forall i \in \mathcal{I}, \quad (4)$$

$$dZ_i(t)dZ_j(t) = \rho_{ij}dt, \quad \forall i, j \in \mathcal{I}, i \neq j. \quad (5)$$

Proposition 3.1 (see Secomandi et al. 2010, Proposition 1 and Lemma 1) contains structural properties of the optimal value function of EDP and its associated optimal policy. These properties serve as a reference for comparing the structural properties of the ADPs discussed in §4.

Proposition 3.1. (a) In every stage i , the value function $V_i(x_i, F_i)$ is concave in x_i for each given $F_i \in \mathbb{R}_+^{N-i}$; and (b) the optimal policy for EDP features two base-stock targets, $\underline{b}_i(F_i)$ and $\bar{b}_i(F_i) \in \mathcal{X}$, which depend on i and F_i and are such that $\underline{b}_i(F_i) \leq \bar{b}_i(F_i)$, and an optimal action $a_i^*(x_i, F_i)$ satisfies

$$a_i^*(x_i, F_i) = \begin{cases} C^I \vee [x_i - \underline{b}_i(F_i)], & \text{if } x_i \in [0, \underline{b}_i(F_i)), \\ 0, & \text{if } x_i \in [\underline{b}_i(F_i), \bar{b}_i(F_i)], \\ C^W \wedge [x_i - \bar{b}_i(F_i)], & \text{if } x_i \in (\bar{b}_i(F_i), \bar{x}]. \end{cases} \quad (6)$$

Moreover, in each stage $i \in \mathcal{I}$, if C^I , C^W and \bar{x} are integer multiples of some maximal number $Q \in \mathbb{R}_+$, then (c) $V_i(x_i, F_i)$ is piecewise linear and continuous in $x \in \mathcal{X}$ for each $F_i \in \mathbb{R}_+^{N-i}$; (d) $V_i(x_i, F_i)$ can change slope in inventory x only at inventory levels that are integer multiples of Q ; and (e) the optimal base-stock targets $\underline{b}_i(F_i)$ and $\bar{b}_i(F_i)$ can be chosen to be integer multiples of Q .

3.2 Bounding Approach

In general, computing an optimal policy for EDP under model (4)-(5) is computationally intractable. In this subsection, we describe a procedure based on Monte Carlo simulation for estimating lower and

upper bounds on the value of storage, as well as a suboptimal policy, given an approximation to the optimal EDP value function. We illustrate this procedure using the value function approximation $\hat{V}_i(x_i, s_i)$, which we assume is available. This approximate value function only includes the spot price s_i from the forward curve. Nevertheless, the same approach extends to other value function approximations that contain a larger subset of the forward curve in their states.

To estimate a lower bound, given a state (x_i, F_i) in stage i , we use $\hat{V}_i(x_i, s_i)$ as an approximation to $V_i(x_i, F_i)$, and compute a feasible action by solving the following greedy optimization problem:

$$\max_{a \in \mathcal{A}(x_i)} r(a, s_i) + \delta \mathbb{E} \left[\hat{V}_{i+1} \left(x_i - a, \tilde{F}_{i+1} \right) | F'_i \right]. \quad (7)$$

As discussed in §4.1, performing this optimization requires approximating the expectation term in 7. We apply this greedy action and sample the forward curve F_{i+1} to obtain the new state $(x - a_i(x_i, F_i), F_{i+1})$. We continue in this fashion until we reach time T_{N-1} . We then discount to time T_0 and cumulate the values of the rewards generated during this process. We repeat this process over multiple samples and average their discounted total cash flows to estimate the value of the greedy policy. This provides a lower bound estimate.

When a value function approximation is computed by an ADP, it is typically possible to generate improved lower bound estimates by sequentially reoptimizing this ADP, within the Monte Carlo simulation used to estimate lower bounds, in order to obtain updated value function approximations. Specifically, the value function approximation obtained by solving an ADP at time T_i spans times T_i through T_{N-1} . However, we only implement the greedy action induced by this value function at time T_i . At time T_{i+1} we account for this action and the newly available forward curve F_{i+1} , and re-solve the ADP. We repeat this procedure until time T_{N-1} . The sum of the rewards corresponding to the greedy actions performed at times T_0 through T_{N-1} discounted back to time T_0 provides a sample lower bound value. We estimate the reoptimized lower bound by averaging these sample values obtained over multiple sample paths.

For upper bound estimation, we use the information relaxation and duality approach for MDPs (see Brown et al. 2010, and references therein). Specifically, given a finite collection of forward curve sample paths, we solve a deterministic EDP version for each such sample path obtained by assuming perfect information of sampled forward curve evolution. That is, each such problem

corresponds to a sample path of the forward curve from time T_0 through time T_{N-1} . Moreover, the per stage rewards of each one of these problems are modified by penalties for using future forward curve information. These penalty terms are obtained from the given value function approximation. We estimate a dual upper bound as the average of the value functions of the stated collection of deterministic EDPs in the initial stage and state. This estimate can be done efficiently via Monte Carlo simulation because each such EDP is tractable.

4 A PSR Approach to Approximate Linear Programming

In this section, we present our PSR approach to approximate linear programming for EDP. We motivate our approach by applying approximate linear programming to a discretized version of EDP and by bringing to light a deficiency of the resulting ALP in §4.1. Our PSR approach, discussed in §4.2, addresses this deficiency. We propose two PSR models in §§4.3-4.4, and show that the model of LMS can be obtained as a specific PSR in §4.5. We discuss generalizations of our PSR approach in §4.6.

4.1 An Approximate Linear Programming Approach

A linear programming formulation of an MDP is well-known if the state and action spaces in each stage of this MDP are finite (Puterman 1994). EDP has infinite state and action spaces in every stage. Therefore, we discretize these spaces to obtain a linear program. By Proposition 3.1 we can optimally discretize the continuous inventory set \mathcal{X} into the finite set \mathcal{X}^D , and the feasible action set $\mathcal{A}(x)$ for inventory level $x \in \mathcal{X}^D$ into the finite set $\mathcal{A}^D(x)$. We assume that the set of feasible forward curves implied by (4)-(5) is discretized using a multidimensional extension of the Rubinstein (1994) binomial lattice method, which is a popular technique in computational finance (Broadie and Detemple 1996; details of this extension are available from the authors upon request). For now, let $\mathcal{F}_{i,j}^D$ denote the finite set of distinct values that the futures price $F_{i,j}$ can take on, and let \mathcal{F}_i^D represent the finite set of forward curve at time T_i . In addition to a discretization of the forward curves, the lattice method that we employ provides a probability mass function over their discretized values. In the rest of this paper we assume that the expectations in EDP and other ADPs where the state and action spaces are discrete, are expressed with respect to this and analogous probability

mass functions. Replacing the continuous sets that define EDP with discretized sets gives rise to an MDP that we refer to as the Discretized Dynamic Program (DDP). A linear program equivalent to DDP is

$$\mathbf{ELP:} \quad \min_{V^D} \sum_{i \in \mathcal{I}, x_i \in \mathcal{X}^D, F_i \in \prod_{j=i}^{N-1} \mathcal{F}_{i,j}^D} V_i^D(x_i, F_i) \quad (8)$$

$$V_i^D(x_i, F_i) \geq r(a, s_i) + \delta \mathbb{E} \left[V_{i+1}^D \left(x_i - a, \tilde{F}_{i+1} \right) | F_i' \right],$$

$$\forall i \in \mathcal{I}, (x_i, F_i) \in \mathcal{X}^D \times \prod_{j=i}^{N-1} \mathcal{F}_{i,j}^D, a \in \mathcal{A}^D(x_i), \quad (9)$$

$$V_N^D(x_N, F_N) = 0, \quad \forall x_N \in \mathcal{X}. \quad (10)$$

The decision variables of ELP are the $V_i^D(x_i, F_i)$ terms. The constraints of ELP are defined over stage-state-action triples. Solving ELP even for unrealistically small instances is computationally intractable, due to the curse of dimensionality.

To overcome this hurdle, we turn to approximate linear programming. The basic idea in approximate linear programming is to replace the value function $V_i^D(x_i, F_i)$ by a linear combination of basis functions. The variables in the resulting linear program are the basis weights in this linear combination. The grid-based value function approximations that we consider, denoted by ϕ_i , are look-up tables, which can also be defined using indicator basis functions. Specifically, we use the approximation $V_i^D(x_i, F_i) \approx \phi_i(x_i, s_i)$, where $\phi_i(x_i, s_i)$ is a look-up table that in stage i only depends on the inventory level $x_i \in \mathcal{X}^D$ and the spot price $s_i \in \mathcal{F}_{i,i}^D$. This choice of approximate value function reduces the dimensionality of the problem by considering only the spot price rather than the entire forward curve. For ease of exposition, in this subsection we restrict our development and analysis to the value function approximation $\phi_i(x_i, s_i)$, but extensions to value function approximations with two or more prices are straightforward.

Applying this value function approximation to ELP yields the ALP

$$\min_{\phi} \sum_{i \in \mathcal{I}, x_i \in \mathcal{X}^D, s_i \in \mathcal{F}_{i,i}^D} \phi_i(x_i, s_i) \quad (11)$$

$$\begin{aligned} \phi_i(x_i, s_i) &\geq r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_i'], \\ &\forall i \in \mathcal{I}, (x_i, F_i) \in \mathcal{X}^D \times \prod_{j=i}^{N-1} \mathcal{F}_{i,j}^D, a \in \mathcal{A}^D(x_i), \end{aligned} \quad (12)$$

$$\phi_N(x_N, s_N) = 0, \forall x_N \in \mathcal{X}^D. \quad (13)$$

The ALP (11)-(13) contains a much smaller number of decision variables, the $\phi_i(x_i, s_i)$ terms, than ELP but has the same number of constraints. However, in stage i , the only futures price relevant to the evolution of $F_{i,i+1}$ into s_{i+1} is $F_{i,i+1}$ (see (4)-(5); this also is true when this model is discretized using a binomial lattice). Therefore, we can equivalently rewrite (11)-(13) as

$$\mathbf{GALP:} \min_{\phi} \sum_{x_i \in \mathcal{X}^D, s_i \in \mathcal{F}_{i,i}^D} \phi_i(x_i, s_i) \quad (14)$$

$$\begin{aligned} \phi_i(x_i, s_i) &\geq r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}], \\ &\forall i \in \mathcal{I}, (x_i, s_i, F_{i,i+1}) \in \mathcal{X}^D \times \prod_{j=i}^{i+1} \mathcal{F}_{i,j}^D, a \in \mathcal{A}^D(x_i), \end{aligned} \quad (15)$$

$$\phi_N(x_N, s_N) = 0, \forall x_N \in \mathcal{X}^D. \quad (16)$$

GALP has fewer constraints than (11)-(13).

The value function approximation corresponding to an optimal solution to GALP, which we continue to denote by $\phi(x_i, s_i)$, is defined over the discrete set $\mathcal{X}^D \times \mathcal{F}_i^D$. In contrast, the value function approximation required for lower and upper bound computation (see §3.2) must be defined over the continuous sets used in the formulation of EDP. To bridge this gap, we define a mapping operator $\mathcal{M}_i^F(s_i)$ that maps $s_i \in \mathbb{R}_+$ onto \mathcal{F}_i^D , and allows us to use this value function approximation as $\phi(x_i, \mathcal{M}_i^F(s_i))$ for bound computation (it can be shown that only inventory levels in the set \mathcal{X}^D are needed in this computation). In later sections, continuous versions of other discrete value function approximations can be derived using similar mapping operators.

GALP can be solved to optimality using standard linear programming software. However, Proposition 4.1 shows that GALP can be equivalently expressed as the following MDP, which can

be solved by backward recursion:

$$\mathbf{GADP:} \quad \phi_i(x_i, s_i) = \max_{F_{i,i+1} \in \mathcal{F}_{i,i+1}^D} \left\{ \max_{a \in \mathcal{A}^D(x_i)} r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}] \right\}, \quad (17)$$

$\forall i \in \mathcal{I}$ and $(x_i, s_i) \in \mathcal{X}^D \times \mathcal{F}_{i,i}^D$, with $\phi_N(x_N, S_N) = 0, \forall x_N \in \mathcal{X}$. Given the inventory and spot price pair (x_i, s_i) , we assume without loss of generality that the maximizers of the inner and outer maximizations in (17) are unique.

Proposition 4.1. *Assume that in every stage $i \in \mathcal{I}$ the set $\mathcal{F}_{i,i}^D$ is bounded. The optimal value function of GADP optimally solves GALP.*

Proof. Our assumption that $\mathcal{F}_{i,i}^D$ is bounded implies that ELP is bounded. As a restriction of ELP, GALP has a bounded optimum or is infeasible. A feasible solution to GALP can be constructed by solving (17). Further, a bounded optimal solution to GALP must satisfy (17), since GALP has greater than or equal to constraints. Otherwise, if a bounded optimal solution does not satisfy (17), then at least one constraint of GALP does not bind and the optimal objective function value of GALP can be improved, which contradicts the assumed optimality of this solution. \square

GADP has two maximization operators: One over the action space $\mathcal{A}^D(x_i)$, the other over the futures price set $\mathcal{F}_{i,i+1}^D$. We have observed in computational experiments that the maximizer in the latter maximization is typically the largest value in the set $\mathcal{F}_{i,i+1}^D$. These “unlikely” prices, that is, prices in the right tail of the distribution of the random variable $\tilde{F}_{i,i+1}$ conditional on $F_{0,i+1}$, determine the value function approximation. Therefore, this is an unrealistic value function approximation that yields lower and upper bounds with poor quality when using the bounding approach discussed in §3.2. The letter “G” in the labels GALP and GADP was chosen to indicate the inherent “greediness” of these models over the set $\mathcal{F}_{i,i+1}^D$.

4.2 PSR

We introduce a PSR of GALP to attempt to correct the poor bounding performance of GALP. We also derive ADPs from different PSRs. Unlike GADP, these ADPs do not have an outer maximization over the set $\mathcal{F}_{i,i+1}^D$.

Our PSR procedure includes two steps: (1) Create a partition of the GALP constraint set into the K sets $\mathcal{G}_1, \mathcal{G}_2, \dots$, and \mathcal{G}_K , and (2) replace each constraint set \mathcal{G}_k , $k \in \{1, 2, \dots, K\}$, by a single surrogate constraint in the sense of Glover (1968, 1975); that is, the k -th such constraint is a non-negative linear combination of constraints in the set \mathcal{G}_k . More specifically, express the constraints of set \mathcal{G}_k as the the system of linear inequalities $A^k z^k \geq d_k$. We then choose a compatible vector of non-negative multipliers u^k , and replace \mathcal{G}_k by the single constraint $u^k A^k z^k \geq u^k d_k$. We repeat this for all the sets in the partition. Clearly, the resulting system of constraints is implied by the GALP constraints, and is a relaxation of GALP.

Two decisions are required when using our PSR approach: (1) How to select the partition of the constraint set; and (2) how to choose the surrogate multipliers used to derive the surrogate constraints. By making these decisions in three different ways, we derive three PSRs, two of which result in new ADPs, and one that yields the LMS ADP (in this case PSR is applied to a math program that is equivalent to GALP).

4.3 A Single Price PSR and ADP

In this subsection, we present a natural PSR of GALP. Each constraint of GALP is defined over the tuple $(i, x_i, s_i, F_{i,i+1}, a)$. We partition the constraints of GALP according to the values of (i, x_i, s_i) , that is, we have $|\mathcal{I}| \times |\mathcal{X}^D| \times |\mathcal{F}_{i,i}^D|$ sets in this partition, with all the constraints in each one of these sets defined for fixed values of (i, x_i, s_i) .

In §4.1, we attribute the poor bounding performance of GADP to its value function approximation being determined by $F_{i,i+1}^M(s_i) := \max\{F_{i,i+1} | F_{i,i+1} \in \mathcal{F}_{i,i+1}^D(s_i)\}$, where $\mathcal{F}_{i,i+1}^D(s_i)$ is the set of all prompt month futures prices in $\mathcal{F}_{i,i+1}^D$, conditioned on the event that the spot price s_i has occurred. Without loss of generality, we assume $F_{i,i+1}^M(s_i)$ to be unique. This implies that for a pair (x_i, s_i) , the constraint corresponding to $F_{i,i+1}^M(s_i)$ and the optimal action associated with this price, that is, the one corresponding to the optimal action in the inner maximization in (17)), holds as an equality in an optimal solution to GALP. An intuitively better choice for this binding constraint is the constraint corresponding to the optimal action in (17) when $F_{i,i+1}$ equals to the expected prompt month futures price given the spot price in stage i , s_i , and the maturity T_{i+1} futures price in stage 0, $F_{0,i+1}$. That is, $\bar{F}_{i,i+1}(s_i, F_{0,i+1}) := \mathbb{E}[\tilde{F}_{i,i+1} | s_i, F_{0,i+1}]$ is a better choice than $F_{i,i+1}^M(s_i)$ as it is more “probable” than $F_{i,i+1}^M(s_i)$.

To ensure that this constraint binds at optimality, we delete from each subset of constraints identified by (i, x_i, s_i, a) , that is, a subset obtained by fixing an action in the (i, x_i, s_i) partition set, all the constraints corresponding to $F_{i,i+1} \neq \bar{F}_{i,i+1}(s_i, F_{0,i+1})$. Therefore, the surrogate multipliers are equal to 1 when $F_{i,i+1} = \bar{F}_{i,i+1}(s_i, F_{0,i+1})$, and to 0 otherwise. If $\bar{F}_{i,i+1}(s_i, F_{0,i+1}) \notin \mathcal{F}_{i,i+1}^D(s_i)$, then we use the value closest to it in $\mathcal{F}_{i,i+1}^D(s_i)$ as a proxy.

Applying this PSR to GALP yields the following relaxation of its constraint set:

$$\phi_i(x_i, s_i) \geq r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | \bar{F}_{i,i+1}(s_i, F_{0,i+1})], \quad (18)$$

$\forall i \in \mathcal{I}, (x_i, s_i) \in \mathcal{X}^D \times \mathcal{F}_{i,i}^D$, and $a \in \mathcal{A}^D(x_i)$. Since this relaxed constraint set has a single constraint for each given tuple (i, x_i, s_i, a) , it is straightforward to notice that GALP with (15) relaxed by (18) is equivalent to the following ADP:

$$\mathbf{ADP1:} \quad \phi_i(x_i, s_i) = \max_{a \in \mathcal{A}^D(x_i)} r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | \bar{F}_{i,i+1}(s_i, F_{0,i+1})], \quad (19)$$

$\forall i \in \mathcal{I}$ and $(x_i, s_i) \in \mathcal{X}^D \times \mathcal{F}_{i,i}^D$.

It is not hard to show that the ADP1 optimal value function and policy share analogous properties to those of EDP (see Proposition 3.1). In particular, ADP1 has a base-stock optimal policy. This property provides theoretical support for ADP1 and allows us to compute its optimal value function more efficiently than using enumeration (see §5 in LMS).

4.4 A Two Price PSR and ADP

ADP1 computes a value function approximation that in every stage depends only on the spot price, in addition to inventory, a specification analogous to that of LMS. In this subsection, we discuss a richer value function approximation, which in each stage depends on the spot and prompt month futures prices, in addition to inventory. We denote this value function approximation in stage i by $\phi_i(x_i, s_i, F_{i,i+1})$.

We compute this value function approximation as a PSR of a version of GALP with decision variables $\phi_i(x_i, s_i, F_{i,i+1})$, and constraints expressed accordingly. Our PSR is analogous to that used in §4.3, with the obvious modification that $\bar{F}_{i,i+1}(s_i, F_{0,i+1})$ is replaced by $\bar{F}_{i,i+1}(s_i, F_{i,i+1}, F_{0,i+2}) :=$

$\mathbb{E}[F_{i,i+2}|s_i, F_{i,i+1}, F_{0,i+2}]$. This yields the following ADP:

ADP2:

$$\begin{aligned} \phi_i(x_i, s_i, F_{i,i+1}) &= \max_{a \in \mathcal{A}^D(x_i)} r(a, s_i) \\ &\quad + \delta \mathbb{E} \left[\phi_{i+1}(x_i - a, \tilde{s}_{i+1}, \tilde{F}_{i,i+1}) | \bar{F}_{i,i+1}(s_i, F_{i,i+1}, F_{0,i+2}) \right], \\ \forall i \in \mathcal{I} \setminus \{N-2, N-1\}, (x_i, s_i, F_{i,i+1}, F_{i,i+2}) &\in \mathcal{X}^D \times \prod_{j=i}^{i+2} \mathcal{F}_{i,j}^D, \end{aligned} \quad (20)$$

$$\begin{aligned} \phi_i(x_i, s_i, F_{i,i+1}) &= \max_{a \in \mathcal{A}^D(x_i)} r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}], \\ \forall i \in \{N-2, N-1\}, (x_i, s_i, F_{i,i+1}) &\in \mathcal{X}^D \times \prod_{j=i}^{i+1} \mathcal{F}_{i,j}^D, \end{aligned} \quad (21)$$

$$\phi_N(x_N, s_N) := 0, \forall x_N \in \mathcal{X}^D. \quad (22)$$

It is easy to show that ADP2 shares the same structural properties of ADP1 and EDP. As for ADP1, this provides theoretical support for ADP2 and facilitates the computation of its optimal value function.

4.5 A Single Price Nonlinear PSR

We now investigate the relationship between our PSR approach to generate ADPs and the ADP of LMS. Their ADP in our notation is

$$\mathbf{SADP:} \quad \phi_i(x_i, s_i) := \mathbb{E} \left[\max_{a \in \mathcal{A}(x_i)} r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}] | s_i, F_{0,i+1} \right], \quad (23)$$

$\forall i \in \mathcal{I}$ and $(x_i, s_i) \in \mathcal{X}^D \times \mathcal{F}_{i,i}^D$. LMS do not derive SADP using an approximate linear programming relaxation approach. Therefore, it is natural to ask if there are any connections between SADP and our PSR approach. Proposition 4.2 shows that SADP is a specific PSR of an equivalent reformulation of GALP.

Proposition 4.2. *SADP can be derived as a PSR of a mathematical program that is equivalent to GALP.*

Proof. Consider GALP. Its constraints can be equivalently rewritten as

$$\phi_i(x_i, s_i) \geq \max_{a \in \mathcal{A}^D(x_i)} \{r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}]\}, \quad (24)$$

$\forall i \in \mathcal{I}$ and $(x_i, s_i, F_{i,i+1}) \in \mathcal{X}^D \times \mathcal{F}_{i,i}^D \times \mathcal{F}_{i,i+1}^D$. Denote by $\mathbb{P}(F_{i,i+1} | s_i, F_{0,i+1})$, the probability that the random prompt month futures price $\tilde{F}_{i,i+1}$ equals $F_{i,i+1}$ conditional on the spot price being equal to s_i at time T_i and the futures price with maturity T_{i+1} being equal to $F_{0,i+1}$ at time T_0 . We can derive a single surrogate constraint for each triple (i, x_i, s_i) from the subset of constraints in (24) corresponding to $(i, x_i, s_i, F_{i,i+1})$ as

$$\begin{aligned} \phi_i(x_i, s_i) \geq & \sum_{F_{i,i+1} \in \mathcal{F}_{i,i+1}^D(s_i)} \mathbb{P}(F_{i,i+1} | s_i, F_{0,i+1}) \\ & \cdot \max_{a \in \mathcal{A}^D(x_i)} [r(a, s_i) + \delta \mathbb{E} [\phi_{i+1}(x_i - a, \tilde{s}_{i+1}) | F_{i,i+1}]], \end{aligned} \quad (25)$$

where we use the equality $\sum_{F_{i,i+1} \in \mathcal{F}_{i,i+1}^D(s_i)} \mathbb{P}(F_{i,i+1} | s_i, F_{0,i+1}) = 1$ to obtain $\phi_i(x_i, s_i)$ on the left hand side of (25).

Consider the math program SMP created by using constraint (25) for each triple $(i, x_i, s_i) \in \mathcal{I} \times \mathcal{X}^D \times \mathcal{F}_{i,i}^D$, and the GALP objective function. For each (i, x_i, s_i) triple, there is a single constraint (25). We claim that this constraint holds as an equality in an optimal solution to SALP. Suppose our claim is not true. Denote by $\phi_i^*(x_i, s_i)$ an optimal decision variable for SMP. Then there exists a triple (i, x_i, s_i) such that $\phi_i^*(x_i, s_i)$ is strictly greater than the right hand side of (25). Since the variable $\phi_i(x_i, s_i)$ only appears in one constraint in the left hand side of (25), and with a positive coefficient in the right hand side of constraints in stage $i - 1$, it is possible to reduce the value of this variable strictly below $\phi_i^*(x_i, s_i)$ while maintaining feasibility. However, this also reduces the value of the SMP objective function below the optimal value, since the decision variable $\phi_i(x_i, s_i)$ has a coefficient equal to 1 in this objective function, which contradicts the optimality of $\phi_i^*(x_i, s_i)$. Therefore, each constraint (25) holds as an equality in an optimal solution to SMP, and an optimal solution to SMP solves SADP. \square

Proposition 4.2 illustrates how a larger class of ADPs can be derived by first rewriting GALP in an equivalent form before applying the two step PSR procedure introduced in §4.2.

4.6 PSR Generalizations

Generalizations of our PSR approach are possible. Although the first step in our approach is restricted to considering *partitions* of the constraint set of GALP, our relaxation procedure easily extends to the case when the sets $\mathcal{G}_1, \mathcal{G}_2, \dots$, and \mathcal{G}_K do not form such a partition. However, for a general choice of these sets, the resulting relaxed linear program may not be representable as an MDP. Proposition 4.3 provides sufficient conditions for the choice of these sets to yield such an MDP. For ease of exposition, we state our conditions with reference to GALP, but extensions to ALPs with approximate value functions containing a larger subset of the forward curve in their states are straightforward. We omit the proof of Proposition 4.3 as it is similar to that of Propositions 4.1 and 3.1.

Proposition 4.3. *If each constraint in set \mathcal{G}_k , $k \in \{1, \dots, K\}$, shares the same triple (i, x_i, s_i) , then the linear program resulting from the PSR of GALP based on the sets \mathcal{G}_k , $k \in \{1, \dots, K\}$, has an equivalent MDP representation. Further, the resulting MDP shares the properties of Proposition 3.1.*

5 Computer Implementation and Complexity of Mappings

In this section we discuss the computer implementation of our ADPs, and the computational complexity of mapping futures prices encountered during simulation onto the discrete set of futures prices used in our implementation.

Discretization of futures prices and computation of conditional expectations are required to solve our ADPs. We use Rubinstein (1994) lattices for this purpose. Consider ADP1, which contains $s_i \equiv F_{i,i}$ in its state and requires $F_{i,i+1}$ for computing the conditional expectation in stage i . We discretize the spot price $F_{i,i}$ and the prompt month futures price $F_{i,i+1}$ by discretizing \mathbb{R}_+^2 . The required discretization and a probability mass function over its values can be created by evolving the time 0 futures prices $\{F_{0,i}, F_{0,i+1}\}$ using the Rubinstein three-dimensional binomial tree (see Rubinstein 1994). We call the resulting grid of price values and associated probabilities in a given stage a *price state grid* (Figure 1), as it approximates the exogenous part of the state space in that stage. This grid includes $(m_i + 1) \times (m_i + 1)$ values for the price pair $(s_i, F_{i,i+1})$, where m_i is the number of time steps used to discretize the time interval $[0, T_i]$. We denote by

$G(F_{i,i}, F_{i,i+1}|F_{0,i}, F_{0,i+1})$ the probability mass function of the pair $\{F_{i,i}, F_{i,i+1}\}$ in the price state grid at stage i .

Solving ADP1 requires computing the expectations $\bar{F}_{i,i+1}(s_i, F_{0,i+1}) \equiv \mathbb{E}[\tilde{F}_{i,i+1}|s_i, F_{0,i+1}]$ and $\mathbb{E}[\phi(\cdot, \tilde{F}_{i+1,i+1})|F_{i,i+1}]$. The first expectation can be computed using $G(F_{i,i}, F_{i,i+1}|F_{0,i}, F_{0,i+1})$. To compute the second expectation, we need transition probabilities from the stage i price state grid to values in the stage $i + 1$ price state grid.

First, we obtain the distribution of $\tilde{F}_{i+1,i+1}$ given $F_{i,i+1}$ by evolving a two-dimensional Rubinstein (1994) binomial tree starting from $F_{i,i+1}$ in the stage i price state grid (Figure 1). We call this distribution the stage i transition grid. This grid includes m values for s_{i+1} , where m is the number of time steps used to discretize the time interval $[T_i, T_{i+1}]$. We denote its corresponding probability mass function by $L(F_{i+1,i+1}|F_{i,i+1})$.

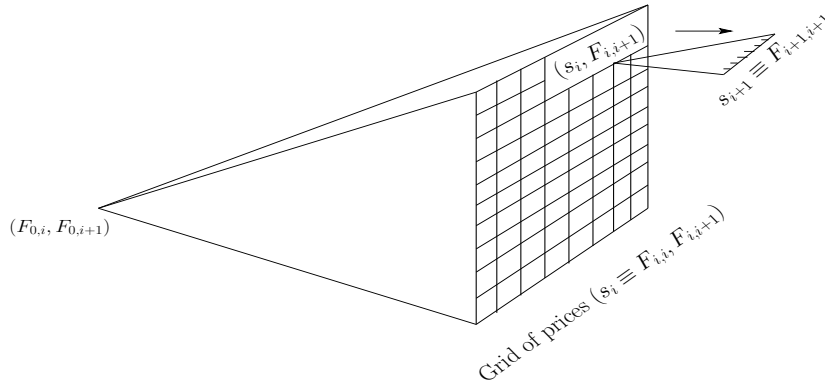


Figure 1: Illustration of the computation of the stage i state and transition grids

Second, we map the stage i transition grid to the stage $i + 1$ price state grid. We do this by mapping a price $F'_{i+1,i+1}$ in the stage i transition grid to the pair $\{F_{i+1,i+1}, F_{i+1,i+2}\}$ in the stage $i + 1$ price state grid that minimizes the quantity $|F'_{i+1,i+1} - F_{i+1,i+1}|$. The s_i values in the stage i price state grid and the s_{i+1} values in the stage i transition grid are sorted. Further, $\bar{F}_{i,i+1}(s_i, F_{0,i+1})$ increases with s_i , and the values of s_{i+1} in the stage i transition grid increase with $\bar{F}_{i,i+1}(s_i, F_{0,i+1})$. Therefore, for any $1 \leq p \leq m + 1$, the $m_i + 1$ sorted s_{i+1} values obtained by picking the p -th value from each of the $m_i + 1$ transition grids in stage i , in increasing order of $\bar{F}_{i,i+1}(s_i, F_{0,i+1})$, can be mapped onto the stage $i + 1$ price state grid in $O(m_{i+1})$ operations. Hence, the $m_i + 1$ stage i transition grids can be mapped onto the stage $i + 1$ price state grid using $O(m_{i+1}m)$ mapping steps.

This procedure can be easily extended to ADP2 using a four-dimensional Rubinstein (1994)

lattice. It can be shown that mapping the stage i transition grids to the stage $i + 1$ price state grid requires $O(m_i m_{i+1} m)$ operations.

6 Computational Results

In this section, we present our computational results. We describe our benchmarking instances from LMS and our computational setup in §6.1. We compare the performance of the ADPs discussed in §§4.3-4.5 in §6.2. We discuss our results with reoptimization in §6.3. We summarize our main findings in §6.4.

6.1 Instances and Computational Setup

We assess the performance of our methodology on the twelve 24-stage instances of LMS, which they created using data from the New York Mercantile Exchange (NYMEX) and the energy trading literature. Each instance has the unique identifier l_1 - l_2 - l_3 , containing the three labels l_1 , l_2 , and l_3 . The label l_1 corresponds to the number of stages, which in our case is 24. The label l_2 indicates the season, one of Spring (Sp), Summer (Su), Fall (Fa), and Winter (Wi). The label l_3 specifies one of three injection and withdrawal capacity restrictions, with 1, 2 and 3 denoting a heavy, intermediate, and mild restriction, respectively. The maximum inventory level \bar{x} is set to 1. The initial inventory is zero in all the instances. The inventory is discretized using 21 equally spaced points. We use 10,000 sample paths to obtain all the reported lower and upper bound estimates.

We use the same computational setup of LMS: A 64 bits Monarch Empro 4-Way Tower Server with four AMD Opteron 852 2.6GHz processors, of which we used only one, each with eight DDR-400 SDRAM of 2GB and running Linux Fedora 13, and the g++ 4.4.4 20100630 (Red Hat 4.4.4-10) compiler. The SADP results in this paper are those reported in LMS.

6.2 Performance Comparison of ADPs

Similar to LMS, we set m_i equal to 500 and m equal to 20 to obtain the results reported in this subsection (see §5 for a discussion of the parameters m_i and m). We denote by LBS and UBS, LB1 and UB1, and LB2 and UB2 the lower and upper bound estimates obtained when using SADP, ADP1, and ADP2, respectively. We start by comparing ADP1 and SADP using the figures in Table

1, where all the bound estimates are normalized by UB2, as it dominates both UBS and UB1. LBS and LB1 are within standard error of each other for the Spring, Summer, and Fall instances; the LB1 standard error is between 0.97% and 1.57% of UB2, and that of LBS is about 1% of UBS (LMS). LB1 is weaker than LBS by no more than 2.90% of UB2 in the Winter instances. UBS and UB1 match in all the instances after accounting for the UB1 standard error, which is between 0.36% and 0.73% of UB2, and the UBS standard error, which is about 0.6% of its value (LMS).

Solving ADP1 is at least 1000 times faster than solving SADP in all instances. Specifically, the run times for ADP1 range from 0.03 to 0.05 CPU seconds, while the run times for SADP are between 135.29 and 146.23 CPU seconds. To compute LB1 we determine feasible actions using (7). Instead, LMS use actions computed when solving SADP. The ADP1 overall run times is at least an order of magnitude shorter in all the instances.

The memory requirements of our ADP1 implementation is about 600 MB, while the SADP implementation in LMS requires 4.6 GB. In other words, ADP1 requires about 8 times less memory than SADP for the same discretization. This allows ADP1 to be run on most conventional laptops for $m_i = 500$.

We now consider the results pertaining to ADP2. ADP2 requires a price state grid with three prices. This makes the direct solution for a discretization of $m_i = 500$ too memory intensive. However, the price grid includes more improbable values as m_i increases. Therefore, we restrict the grid to spot prices in stage i that are less than or equal to 4 times the time zero futures price with maturity at time T_i (restricting the spot price range also restricts the range of the other two prices). This approach, standard in computational finance (Levy 2004), is effective in our application since the spot price, $F_{i,i}$, is highly correlated (over 90%) with the futures prices with the three nearest maturities, ($F_{i,i+1}$, $F_{i,i+2}$ and $F_{i,i+3}$). Table 4 in Online Appendix A, shows that the estimated lower and upper bounds change by less than 0.2% with this lattice restriction in place.

Table 1 reports the estimated upper and lower bound percent ratios for ADP2. As mentioned, UB2 improves upon UBS and UB1 in all the instances. It is remarkable that UB2 is smaller than both UBS and UB1 by an average of 2.62% in the Winter instances. The standard error of UB2 is between 0.32% and 0.66% of its value.

The first three columns of Table 1 clearly indicate the superiority of LB2 when compared with LBS and LB1. LB2 improves LBS by 1-3% in the Spring, Summer, and Fall instances, and by 6-8%

Table 1: LB1, LB2, LBS, UB1 and UBS normalized by UB2; $m_i = 500$ and $m = 20$.

Instance	LBS/UB2 %	LB1/UB2 %	LB2/UB2 %	UBS/UB2 %	UB1/UB2 %
24-Sp-1	94.06	93.52	96.64	100.95	100.95
24-Sp-2	95.08	94.81	98.37	101.14	101.14
24-Sp-3	95.99	95.92	98.83	100.87	100.87
24-Su-1	94.49	93.90	96.94	100.64	100.85
24-Su-2	96.66	96.62	98.96	100.64	100.48
24-Su-3	97.21	97.44	99.19	100.29	100.29
24-Fa-1	95.66	95.33	98.18	100.96	100.96
24-Fa-2	97.19	97.16	99.69	100.78	100.63
24-Fa-3	97.74	97.83	99.76	100.40	100.40
24-Wi-1	82.78	79.88	90.78	102.78	103.33
24-Wi-2	88.31	86.94	96.41	102.82	102.82
24-Wi-3	91.58	90.00	98.11	102.46	102.81

in the Winter instances. The improvements of LB2 on LB1 are similar, with the exception of the Winter instances, where the improvements are larger. The standard error of LB2 varies between 0.92% and 1.49% of UB2. These results indicate that ADP2 is a fundamentally better model than both SADP and ADP1.

Solving ADP2 with restricted lattices requires less CPU time than solving SADP. The ADP2 lower and upper bound simulation times for ADP2 increase considerably from their values when using ADP1. The average ADP2 total CPU time, that is, solving ADP2 and computing the lower and upper bounds, is shorter than that of SADP in 75% of the instances. For the remaining 25%, the difference is less than a minute (the average SADP CPU requirement is 140.01 CPU seconds).

Table 2: Optimality gaps.

Instance	LBS/UBS %	LB1/UB1 %	LB2/UB2 %
24-Sp-1	93.18	92.64	96.64
24-Sp-2	94.01	93.75	98.37
24-Sp-3	95.16	95.09	98.83
24-Su-1	93.89	93.11	96.94
24-Su-2	96.05	96.16	98.96
24-Su-3	96.93	97.15	99.19
24-Fa-1	94.75	94.42	98.18
24-Fa-2	96.43	96.56	99.69
24-Fa-3	97.35	97.44	99.76
24-Wi-1	80.54	77.30	90.78
24-Wi-2	85.88	84.56	96.41
24-Wi-3	89.38	87.55	98.11

Table 2 compares the optimality gaps of SADP, ADP1, and ADP2 computed using their respec-

tive lower and upper bound estimates. This provides an assessment of the stand alone valuation performance of each ADP (without reoptimization). Table 2 shows that ADP2 is the best stand alone ADP model, with optimality gap improvements of 8% to 13% on the Winter instances over SADP and ADP1. SADP performs better than ADP1 on the Winter instances, but the differences between these two ADPs in the other seasons is within standard error.

6.3 Reoptimized Lower Bounds

In this subsection, we discuss our reoptimization results (see §3.2). Reoptimization involves an increased computational burden as it requires the sequential reoptimization of ADPs, one for every stage and sample path in the Monte Carlo simulation used for lower bound estimation. To control these computational times we use a coarser discretization of the price state and transition grids when compared to the no reoptimization case.

With this coarser discretization, we use linear interpolation to evaluate the ADP1 value function at a given spot price encountered in simulation. We analyze the effect of interpolation using two settings: (1) $m_i = m = 5$ as in LMS, and (2) the finer discretization $m_i = m = 10$. Tables 5 and 6 in Online Appendix A contain the resulting lower bound estimates and CPU times compared to the case of no interpolation. Compared to this case, interpolation leads to an average increase of 1.6% in the estimated lower bounds for the Winter instances, but less than a 1% increase in these bounds for the remaining seasons when $m_i = m = 5$. The effect of interpolation is less pronounced for $m_i = m = 10$, where almost all the lower bound improvements are less than 0.5%. As expected, interpolation becomes less important as the discretization becomes finer. As the lower bound estimates with $m_i = m = 5$ and $m_i = m = 10$ are very similar, we discuss only the former estimates.

Table 3 reports the estimates of three reoptimized lower bounds relative to UB2 for $m_i = m = 5$. RLBS and RLB1 are the estimates of the reoptimized lower bounds obtained with SADP and ADP1, respectively. LBRI is the estimate of the lower bound of the rolling intrinsic policy, a common heuristic used in practice that reoptimizes the deterministic version of the MDP (2)-(3); that is, the version of this model that computes the intrinsic value of storage (see LMS and reference therein). We use the LBRI values reported by LMS.

RLB1 is within 1.2% of UB2 in the Spring, Summer, and Fall instances. Its standard errors are

spread between 0.94% and 1.86% of UB2. RLB1 matches RLBS once one accounts for this standard error, which is about 1% of UBS (LMS). Excluding the Winter instances, LBRI is comparable to both RLB1 and RLBS. In the Winter instances, LBRI is somewhat worse than these two lower bound estimates. It is interesting to point out that the resulting rolling intrinsic policy has an estimated optimality gap smaller than 5% in the Winter instances, compared to the 7.5% gap estimated by LMS in this case.

Table 3: Comparison of reoptimized optimality percentage gaps from SADP, ADP1, and the rolling intrinsic policy, using the upper bound from ADP2.

Instance	RLBS/UB2	RLB1/UB2	LBRI/UB2
24-Sp-1	99.31	98.93	99.22
24-Sp-2	99.25	98.82	99.32
24-Sp-3	99.38	99.06	99.37
24-Su-1	99.31	99.00	99.14
24-Su-2	99.87	99.79	99.87
24-Su-3	99.74	99.69	99.61
24-Fa-1	99.92	99.58	99.79
24-Fa-2	100.40	100.23	100.37
24-Fa-3	100.03	99.91	99.91
24-Wi-1	97.71	96.19	95.02
24-Wi-2	99.22	98.51	97.48
24-Wi-3	98.94	98.64	97.06

We also implemented reoptimization with ADP2. The resulting reoptimized lower bounds estimates are not substantially better than those obtained by reoptimizing ADP1, and are thus not discussed here. But the CPU times needed to reoptimize ADP2 are longer than those required to reoptimize both SADP and ADP1. This is because ADP2 requires a price state grid of higher dimensions compared to SADP and ADP1, and lattice restrictions are less effective with the coarse discretizations used for reoptimization. Further, LB2 is worse than RLBS by only a few percent points in all the instances, which is consistent with our observation that reoptimization is less critical for ADP2 than for both SADP and ADP1.

Finally, we compare the CPU times of the reoptimized SADP obtained by LMS against those required to reoptimize ADP1. The average run time in LMS for $m_i = m = 5$ with interpolation is 570 CPU seconds. For the same setting, the average run time for ADP1 is 94.36 CPU seconds, that is, 6 times shorter. Table 6 in Online Appendix A reports specific run times required to compute the reoptimized lower bounds.

6.4 Summary

ADP2 allows us to estimate better upper bounds than those that can be obtained with SADP and ADP1. Reoptimizing ADP1 and ADP2 yields lower bound estimates that are competitive with those generated by reoptimizing SADP. In particular, we are able to reduce the optimality gap on the Winter instances from 4.9% in LMS to 2.3% using our improved upper bounds from ADP2. These improved upper bounds also show that the rolling intrinsic policy has an optimality gap of less than 5% even in the Winter instances, which reduces the previously known gap of 7.5%. Moreover, the quality of the ADP2 lower bounds without reoptimization suggests that ADP2 is a fundamentally better model than both SADP and ADP1.

In terms of computational efficiency, estimating upper and lower bounds using ADP1 is at least an order of magnitude faster than doing this using SADP without reoptimization, and 6 times faster with reoptimization. ADP2 is comparable to SADP for bound estimation without reoptimization in most of the instances, but slower when used with reoptimization.

7 Conclusions

The valuation of natural gas storage is an important problem in practice. This problem gives rise to an intractable MDP when using high dimensional models of the natural gas forward curve evolution. An important feature of this MDP is the presence of this high-dimensional forward curve in its state. We propose a novel approximate linear programming relaxation scheme, based on partitioned surrogate relaxations, to obtain value function approximations to this MDP. We provide sufficient conditions under which the resulting relaxed ALP has an equivalent MDP, which allows for its efficient solution using backward recursion. Moreover, we demonstrate that the LMS ADP is one among many ADPs that can be derived using our approach.

We use our approach to introduce two new ADPs: One that provides lower bound estimates that are comparable to the best estimated lower bounds from LMS, with or without reoptimization, but at a substantially lower computational cost; and another that provides better estimated upper bounds and lower bounds without reoptimization at a similar computational cost for most of the instances considered, and comparable lower bound estimates with reoptimization at a larger computational effort. Using the best upper and lower bound estimates from our study and LMS, we are able to

reduce the average optimality gap on the harder Winter instances from the previously known 4.9% to 2.3%. Our improved estimated upper bounds have managerial relevance, as they imply that the rolling intrinsic policy, which is widespread in practice, is at most 5% suboptimal even on the harder Winter instances, rather than the previously known 7.5%.

Our proposed relaxation approach has potential relevance beyond our specific application. In particular, it may be relevant for valuing the real option to store other commodities and the solution of intractable MDPs with exogenous information variables in their states using approximate linear programming.

It would be interesting to investigate the efficacy of our surrogate relaxation procedure with functional, rather than look-up table value function approximations, in our, as well as other, domains. Another interesting research opportunity would be further strengthening the estimated bounds in the Winter instances in our application, for example by applying pathwise optimization techniques (Desai et al. 2010).

Acknowledgment

We thank Guoming Lai for useful discussions related to the implementation of grid-based ADPs.

References

- D. Adelman. A price-directed approach to stochastic inventory/routing. *Operations Research*, 52(4):499–514, 2004. 4
- D. Adelman. Math programming approaches to approximate dynamic programming. Tutorial, INFORMS Annual Meeting, Pittsburgh, PA, USA, 2006. 4
- D. Adelman. Dynamic bid prices in revenue management. *Operations Research*, 55(4):647–661, 2007. 4
- D. Adelman and D. Klabjan. Computing near optimal policies in generalized joint replenishment. *INFORMS Journal on Computing*, Forthcoming, 2011. 4
- D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, USA, 1996. 4
- A. Boogert and C. De Jong. Gas storage valuation using a Monte Carlo method. *The Journal of Derivatives*, 15(3):81–98, 2008. 5

- M. Broadie and J. Detemple. American option valuation: New bounds, approximations, and a comparison of existing methods. *The Review of Financial Studies*, 9(4):1211–1250, 1996. 9
- D. B. Brown, J. E. Smith, and P. Sun. Information relaxations and duality in stochastic dynamic programs. *Operations Research*, 58(4):1–17, 2010. 2, 5, 8
- R. Carmona and M. Ludkovski. Valuation of energy storage: An optimal switching approach. *Quantitative Finance*, 10(4):359–374, 2010. 5
- H.S. Chang, M.C. Fu, J. Hu, and S.I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Springer-Verlag, London, UK, 2007. 4
- Z. Chen and P.A. Forsyth. A semi-Lagrangian approach for natural gas storage valuation and optimal operation. *SIAM Journal on Scientific Computing*, 30(1):339–368, 2007. 5
- D. P. de Farias and B. Van Roy. On constraint sampling for the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2001. 4
- D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003. 4
- D. P. de Farias and B. Van Roy. A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. *Mathematics of Operations Research*, 31(3):597–620, 2006. 4
- V. V. Desai, V. F. Farias, and C. C. Moallemi. Approximate dynamic programming via a smoothed approximate linear program. Working paper, Columbia Univ., 2009. 5
- V. V. Desai, V. F. Farias, and C. C. Moallemi. Pathwise optimization for optimal stopping problems. Working paper, Columbia Univ., 2010. 25
- A. K. Dixit and R. S. Pindyck. *Investment Under Uncertainty*. Princeton University Press, Princeton, New Jersey, USA, 1994. 1
- A. Eydeland and K. Wolyniec. *Energy and Power Risk Management*. John Wiley and Sons Inc., Hoboken, NJ, USA, 2003. 7
- V. F. Farias and B. Van Roy. An approximate dynamic programming approach to network revenue management. Working paper, Stanford Univ., 2007. 4
- F. Glover. Surrogate constraints. *Operations Research*, 16(4):741–749, 1968. 5, 13
- F. Glover. Surrogate constraint duality in mathematical programming. *Operations Research*, 23(3):434–451, 1975. 5, 13
- J. Gray and P. Khandelwal. Towards a realistic gas storage model. *Commodities Now*, pages 1–4, June 2004.

- M. B. Haugh and L. Kogan. Pricing American options: A duality approach. *Operations Research*, 52(2): 258–270, 2004. 5
- G. Lai, F. Margot, and N. Secomandi. An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation. *Operations Research*, 58(3):564–582, 2010. 2
- G. Levy. *Computational Finance: Numerical Methods for Pricing Financial Instruments*. Butterworth-Heinemann, Oxford, UK, 2004. 20
- S. Maragos. *Valuation of Operational Flexibility of Natural Gas Reservoirs*, pages 431–456. Real Options and Energy Management Using Options Methodology to Enhance Capital Budgeting Decisions. Risk Publications, London, UK, 2002. 1, 2
- C. C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queuing networks. Working paper, Stanford Univ., 2008. 4
- J. R. Morrison and P. R. Kumar. New linear program performance bounds for queuing networks. *Journal of Optimization Theory and Applications*, 100(3):575–597, 1999. 4
- M. Petrik and S. Zilberstein. Constraint relaxation in approximate linear programs. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pages 809–816, Montreal, Canada, 2009. 5
- W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, Hoboken, New Jersey, USA, 2007. 4
- M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994. ISBN 0471619779. 9
- L. C. G. Rogers. Monte Carlo valuation of American options. *Mathematical Finance*, 12(3):271–286, 2002. 5
- M. Rubinstein. Return to OZ. *Risk Magazine*, 1994. 9, 17, 18
- P. J. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985. 2, 4
- N. Secomandi. Optimal commodity trading with a capacitated storage asset. *Management Science*, 56(3): 1090–1049, 2009. 3, 5
- N. Secomandi, G. Lai, F. Margot, A. Scheller-Wolf, and D. Seppi. The effect of model error on the valuation and hedging of natural gas storage. Working paper, Carnegie Mellon Univ., 2010. 5, 7
- M. Thompson, M. Davison, and H. Rasmussen. Natural gas storage valuation and optimization: A real options application. *Naval Research Logistics*, 56(3):226–238, 2009. 5

- M. A. Trick and S. E. Zin. Spline approximations to value functions. *Macroeconomic Dynamics*, 1(1):255–277, 1997. 4
- L. Trigeorgis. *Real Options: Managerial Flexibility and Strategy in Resource Allocation*. The MIT Press, Cambridge, MA, USA, 1996. 1
- B. Van Roy. *Neuro-Dynamic Programming: Overview and Recent Trends*. Handbook of Markov Decision Processes. Kluwer, Boston, MA, USA, 2002. 4
- M. H. Veatch. Approximate dynamic programming for networks: Fluid models and constraint reduction. Working paper, Gordon College, 2005. 4
- O.Q. Wu, D.D. Wang, and Z. Qin. Seasonal energy storage operations with limited flexibility. Working paper, Univ. of Michigan, 2010. 5
- D. Zhang and D. Adelman. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science*, 43(3):381–394, 2009. 4

Online Appendix

A Additional Computational Results

This section contains the following tables that supplement the numerical results discussed in §6:

1. Table 4 illustrates the effect of the lattice restriction on the lower and upper bound estimates obtained using ADP2 with $m_i = 100$ and $m = 20$.
2. Table 5 compares the effect of value function interpolation on the lower bound estimates obtained with ADP1 and reoptimization.
3. Table 6 contains the CPU times when reoptimizing ADP1.

Table 4: The ratio of the lower and upper bound estimates obtained with ADP2 without lattice restriction and those obtained with the lattice restriction ($m_i = 100$ and $m = 20$).

Instance	Estimated Ratio (%)	
	Lower Bound	Upper Bound
24-Sp-1	100.04	100.14
24-Sp-2	100.08	100.15
24-Sp-3	100.14	100.16
24-Su-1	100.04	100.16
24-Su-2	100.08	100.20
24-Su-3	100.05	100.23
24-Fa-1	100.01	100.03
24-Fa-2	100.03	100.06
24-Fa-3	100.04	100.07
24-Wi-1	100.07	100.04
24-Wi-2	100.09	100.07
24-Wi-3	100.11	100.06

Table 5: The effect of value function interpolation on the reoptimized lower bound estimates obtained with ADP1. I-RLB1 denotes the ADP1 lower bound with value function interpolation and reoptimization.

Instance	$m_i = m = 5$			$m_i = m = 10$		
	RLB1	I-RLB1	I-RLB1/RLB1 (%)	RLB1	I-RLB1	I-RLB1/RLB1 (%)
24-Sp-1	4.15	4.17	100.48	4.18	4.18	100.00
24-Sp-2	5.18	5.22	100.77	5.23	5.24	100.19
24-Sp-3	5.66	5.69	100.53	5.70	5.71	100.18
24-Su-1	4.66	4.67	100.21	4.68	4.68	100.00
24-Su-2	6.25	6.28	100.48	6.28	6.28	100.00
24-Su-3	6.78	6.80	100.29	6.80	6.80	100.00
24-Fa-1	4.09	4.13	100.98	4.14	4.15	100.24
24-Fa-2	6.35	6.41	100.94	6.42	6.42	100.00
24-Fa-3	7.47	7.52	100.67	7.53	7.53	100.00
24-Wi-1	1.70	1.73	101.76	1.71	1.72	100.58
24-Wi-2	2.40	2.44	101.67	2.41	2.42	100.41
24-Wi-3	2.77	2.81	101.44	2.77	2.77	100.00

Table 6: CPU seconds required to compute the ADP1 reoptimization based lower bound estimates without and with value function interpolation, that is, RLB1 and I-RLB1.

Instance	$m_i = m = 5$		$m_i = m = 10$	
	RLB1	I-RLB1	RLB1	I-RLB1
24-Sp-1	76.23	94.19	211.24	262.40
24-Sp-2	74.84	93.72	210.48	261.44
24-Sp-3	74.60	94.31	209.84	260.38
24-Su-1	75.49	94.59	211.26	261.25
24-Su-2	74.83	93.98	210.18	260.81
24-Su-3	74.43	94.88	210.77	260.49
24-Fa-1	76.43	94.83	211.32	262.34
24-Fa-2	75.54	93.96	210.33	260.39
24-Fa-3	75.02	94.62	209.95	259.79
24-Wi-1	76.17	95.59	211.30	262.30
24-Wi-2	75.46	93.86	209.84	260.36
24-Wi-3	75.42	93.84	209.45	259.26