Proceedings of the 2007 IEEE Symposium on Approximate
Dynamic Programming and Reinforcement Learning (ADPRL 2007)

An Optimal ADP Algorithm for a High-Dimensional Stochastic Control Problem

Juliana Nascimento
Department of Operations Research
and Financial Engineering
Princeton University
Engineering Quadrangle
Princeton, NJ 08544
Phone: +1 609 258 6239
Email: jnascime@princeton.edu

Warren Powell
Department of Operations Research
and Financial Engineering
Princeton University
Engineering Quadrangle
Princeton, NJ 08544
Phone: +1 609 258 6239
Email: powell@princeton.edu

***Abstract*— We propose a provably optimal approximate dynamic programming algorithm for a class of multistage stochastic problems, taking into account that the probability distribution of the underlying stochastic process is not known and the state space is too large to be explored entirely. The algorithm and its proof of convergence rely on the fact that the optimal value functions of the problems within the problem class are concave and piecewise linear. The algorithm is a combination of Monte Carlo simulation, pure exploitation, stochastic approximation and a projection operation. Several applications, in areas like energy, control, inventory and finance, fall under the framework.**

I. Introduction

We consider a class of multistage stochastic problems called the *concave single asset acquisition class*. The objective is to act on a single type of asset in order to maximize expected contributions in discrete time over a finite horizon.

New information exogenous to the system becomes available at every time period and the probability distribution of the underlying stochastic process is not known, not even parametrically. It is only assumed to be Markov. The new exogenous information at time period t is a possibly correlated vector in $\mathbb{R}^m$, denoted by $\hat{W}_t$, that only depends on $W_{t-1} \in \mathbb{R}^n$, a vector containing all the relevant past information. We denote by R^x_{t-1} the asset level in the system after the decision at period $t-1$ is taken and let R_t be the asset level just before a decision is made at time t. The information $\hat{W}_t$ leads the system to the pre-decision state $S_t = (W_t, R_t) = (f_1(W_{t-1}, \hat{W}_t), f_2(R^x_{t-1}, \hat{W}_t))$, where f_1 and f_2 are deterministic functions.

After S_t is observed, a decision $x_t \in \mathbb{R}^l$, restricted to a convex constraint set $\mathcal{X}(S_t)$, is taken and a contribution $C_t(S_t, x_t)$, linear in x_t, is produced. The asset level changes after the decision and is given by $R^x_t = g(S_t) + A \cdot x_t$, where g is a deterministic scalar function, A is a $1 \times l$ input-output vector and $A \cdot x$ is an inner product operation that translates the effect the decision has on the asset level.

The most important feature of problems in this class is that their optimal value functions are concave and piecewise linear in the asset dimension.

This problem class covers a number of practical applications. We may have an energy company purchasing forward contracts for gas to satisfy an unknown demand for energy for a particular time period in the future. Another example is companies purchasing expensive equipment in advance expecting to lock in lower prices, without knowing the exact amount of equipment necessary in the future. It also falls under the framework a single asset inventory system where demand, selling and replenishing prices fluctuate over time.

The framework generalizes to a broad class of control problems that have a system evolution that depends on both an exogenous process and on a manageable process governed by a scalar such as temperature, flow or concentration. We require only that the contribution (or utility) function be concave, which would occur if there were diminishing returns from the controllable scalar variable.

Since the probability distribution of the underlying stochastic process is not known, standard methods that require the computation of expected values, such as classical backward dynamic programming and multistage stochastic programming, can not be applied, unless the distributions are estimated. Lately, solving stochastic optimization problems with a distribution free approach has been attracting considerable interest. Nonparametric methods for revenue management, multiproduct pricing and single-period newsvendor with its multi-period extension can be found in [1], [2] and [3], respectively. In the latter, bounds were established on the number of samples required to guarantee that the expected cost of the policies is arbitrarily close to the optimal one.

We encounter computational problems even if the distribution of $\hat{W}_t$ is known. W_t may be a vector which, combined with the scalar controllable state variable R_t, can produce extremely large state spaces which can make classical dynamic programming recursions impractical. In section VI, we report on experiments where the state space has as many as 16 million possible values. If we assume the probability distributions are known, exact solutions using classical methods require over 6 hours to compute, while policies that are within .001 of the optimal could be found in a matter of minutes. Q-learning

[4], which is often proposed for model-free applications, is even more difficult since the state space is enlarged with all possible actions.

We propose an approximate dynamic programming algorithm that exploits the structural properties of the optimal value functions of this problem class. It combines Monte Carlo simulation, a pure exploitation scheme, stochastic approximation and a projection operation. It is a generalization of the SPAR ([5]) algorithm, which is presented in the context of a two-stage problem. The proposed algorithm provably converges to an optimal policy and scales to large state spaces.

The optimal value function $V_t^*(W_t, \cdot)$, which is unknown, is piecewise linear, concave and can be represented by its decreasing slopes $v_t^*(W_t, R_{t1}), \ldots, v_t^*(W_t, R_{tN})$ and its corresponding break points $R_{t1}, \ldots, R_{tN}$. The main idea of the algorithm is to construct concave and piecewise linear function approximations $\bar{V}_t^n(W_t, \cdot)$, learning its slopes $\bar{v}_t^n(W_t, R_{t1}), \ldots, \bar{v}_t^n(W_t, R_{tN})$ over the iterations. The catch is that the algorithm does not try to learn the slopes for the whole state space, only for parts close to optimal asset levels, which are determined by the algorithm itself. Figure 1 illustrates the idea.

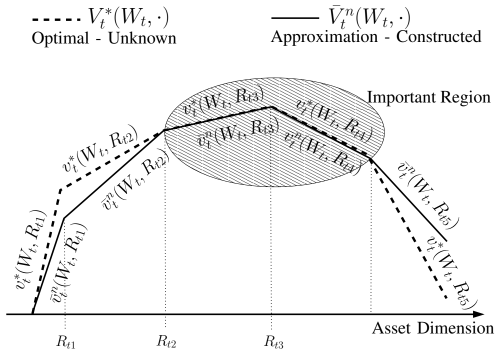


Fig. 1. Optimal value function and the constructed approximation

At each iteration n, the algorithm observes a sample realization of the stochastic process and at each time period t, it uses the sample realization and the current value function approximation to take the decision x_t^n, which is an optimal solution to the following optimization problem:

$$\max_{x \in \mathcal{X}(S_t^n)} \underbrace{C_t(S_t^n, x) + \gamma \bar{V}_t^{n-1}(W_t^n, g(S_t^n) + A \cdot x)}_{F_t(S_t^n, \bar{V}_t^{n-1}, x)},$$

where $S_t^n = (W_t^n, R_t^n)$ is the state at period t before the decision is taken, representing a sample realization of the information process and the asset level, respectively. Moreover, γ is a discount factor and $\bar{V}_t^{n-1}(S_t^n)$ is the current value function approximation. The slopes of $F_t(S_t^n, \bar{V}_t^{n-1}, \cdot)$ to the left and right of x_t^n are used to update the approximate slopes $\bar{v}_{t-1}^{n-1}$ and a projection operation is performed in case a violation of the concavity property occurs.

The proof of convergence of our algorithm to an optimal policy follows ideas from the field of approximate dynamic programming ([6]) as well as the proof of the SPAR algorithm in [5]. Proofs of convergence for Q-learning ([7], [8]) and optimistic policy iteration ([9]) require the sampling of all states and possibly all actions. A convergence proof for a Real Time Dynamic Programming ([10]) algorithm that considers a pure exploitation scheme is provided in [6], but it assumes that the distribution of the random variables is known.

The proposition of a provably optimal approximate dynamic programming algorithm for a class of multistage stochastic problems is the main contribution of this work. Our algorithm handles at the same time the unknown probability distribution of the underlying stochastic process and the curse of the dimensionality that may arise when trying to solve problems within this problem class. This is accomplished by exploiting the structural properties of the optimal value functions.

The algorithm also has applications in different areas like inventory management, finance, energy and control. We also demonstrate experimentally that it outperforms competing algorithms, and, in particular, dramatically outperforms standard backward dynamic programming when the distributions are assumed known.

This paper is organized as follows. Section II describes the problem class and the corresponding dynamic programming model, while section III presents some applications. Section IV describes the algorithm and section V gives a summary of the convergence proof. Section VI presents experimental results associated with one of the applications. Finally, the conclusions are presented in section VII.

II. The Problem Class

A single type of asset must be acquired to maximize the expected discounted contributions over the horizon $t = 0, \ldots, T$. The discount factor is denoted by γ.

During time period t, new information exogenous to the system becomes available. The exogenous process is denoted by $\hat{W} = (\hat{W}_0, \ldots, \hat{W}_T)$. Its underlying distribution is not known. However, each $\hat{W}_t \in R^m$ is a random variable with finite support that may depend only on W_{t-1}^x, where $W_{t-1}^x \in R^n$ is the information given by the exogenous process up until time $t-1$ that is relevant for $\hat{W}_t$.

The state of the system before a decision is taken is

$$S_t = (W_t, R_t) = (f_1(W_{t-1}^x, \hat{W}_t), f_2(R_{t-1}^x, \hat{W}_t)),$$

where W_t is the information relevant for $\hat{W}_{t+1}$ and for taking a decision at t, R_t is the asset level in the system just before a decision at t is taken and R_{t-1}^x is the asset level in the system right after a decision at $t-1$ was taken. The information process, is completely exogenous to the system, while the asset level is influenced by the decisions. Both f_1 and f_2 are deterministic functions that have a bounded and finite image set. The image set of f_2 is also integer and positive. The finite support set of W_t^x is denoted by $\mathcal{W}_t$.

The decision $x_t \in \mathbb{R}^l$ is integer, positive and bounded by a constant M_t. After the decision, the asset level is given by

$$R_t^x = g(S_t) + A \cdot x_t,$$

where g is an integer bounded function and $A \in \mathbb{Z}^l$ is an input-output vector. We thus obtain the post-decision state

$S_t^x = (W_t^x, R_t^x)$. The difference between W_t and W_t^x is that W_t may contain information relevant for taking the decision x_t in addition to the information relevant for $\hat{W}_{t+1}$, while W_t^x only contains the information relevant for $\hat{W}_{t+1}$. It should be clear that the decision x_t does not affect W_t^x.

Clearly, the asset level R_t^x is integer and bounded. In our problem class, it is also positive. Thus, $S_t^x \in \mathcal{S}_t = \mathcal{W}_t \times \{0, \ldots, B_t\}$, where B_t depends on g, A and $M_{t'}$ - the bound on the decision vector $x_{t'}$ for $t' \leq t$.

The decision must satisfy a set of constraints that may depend on the pre-decision state S_t. Formally, $x_t \in \mathcal{X}(S_t) \subset \{x \in \mathbb{R}^l : 0 \leq x \leq M_t\}$, where $\mathcal{X}(S_t)$ is a convex set.

The contribution in period t, for $t = 1, \ldots, T$, is given by

$$C_t(S_t, x_t) = c_{t0}(S_t) + \sum_{i=1}^{l} c_{ti}(S_t) x_{ti},$$

where, for $i = 0, \ldots, l$, $c_{ti}(S_t)$ is a bounded scalar function.

We construct the optimal value functions $V_t^* : \mathcal{S}_t \rightarrow \mathbb{R}$ around the post-decision state variables, taking advantage of the fact that S_t^x is a deterministic function of x_t. For $S_T^x \in \mathcal{S}_T$, we have that $V_T^*(S_T^x) = 0$. For $t = 1, \ldots, T$ and $S_{t-1}^x \in \mathcal{S}_{t-1}$ it is given by

$$V_{t-1}^*(S_{t-1}^x) = \mathbb{E}\left[\max_{x_t \in \mathcal{X}(S_t)} C_t(S_t, x_t) + \gamma V_t^*(S_t^x) | S_{t-1}^x\right].$$

We now address the key property of this problem class. The convex constraint set $\mathcal{X}(S_T)$ is such that $\mathbb{E}\left[\max_{x_T \in \mathcal{X}(S_T)} C_T(S_T, x_T) | S_{T-1}^x\right]$ is a concave function of R_{T-1}^x and the optimal decision x_T^* is integer without imposing integrality. Therefore, for $t = 0, \ldots, T$, $V_t^*(S_t^x)$ is concave and piecewise linear with integer break points in R_t^x.

Given the information W_{t-1}^x, if we disregard the values at asset level zero, the optimal value functions $V_{t-1}^*(S_{t-1}^x)$ can be uniquely identified by its decreasing slopes $\left(v_{t-1}^*(W_{t-1}^x, 1), \ldots, v_{t-1}^*(W_{t-1}^x, B_{t-1})\right)$, where

$$\begin{aligned} v_{t-1}^*(S_{t-1}^x) &= V_{t-1}^*(W_{t-1}^x, R_{t-1}^x) - V_{t-1}^*(W_{t-1}^x, R_t^x - 1) \\ &= \mathbb{E}\left[\hat{G}_t(S_t, v_t^*) | S_{t-1}^x\right]. \end{aligned} \quad (1)$$

The actual representation of the random variable $\hat{G}_t(S_t, v_t^*)$ is problem dependent, but based on our assumptions, we can infer that $v_{t-1}^*(S_{t-1}^x)$ is bounded. Therefore, given a bound B big enough, we have that, $v_{t-1}^*(W_{t-1}^x) = \left(v_{t-1}^*(W_{t-1}^x, 1), \ldots, v_{t-1}^*(W_{t-1}^x, B_{t-1})\right) \in \mathcal{C}_{t-1}$, where

$$\mathcal{C}_{t-1} = \left\{v \in \mathbb{R}^{B_{t-1}} : |v_i| \leq 3B, v_{i+1} \leq v_i \text{ for } i \geq 1\right\}.$$

We simplify notation letting $\bar{\mathcal{S}}_t = \mathcal{W}_t \times \{1, \ldots, B_t\}$, the post-decision state space minus the state pairs $(W_t^x, 0)$.

III. Applications

In order to make the previous section concrete, we describe two practical applications emphasizing their main elements, as the information process, the pre and post-state variables, the decision, its constraint set and the one period contribution. We also give the specific form of the random variable $\hat{G}_t(S_t, v_t^*)$, omitting its derivation.

A. Forward Contracts

At each period, it must be determined the amount of forward contracts of gas that should be purchased in order to meet a positive discrete integer random demand $\hat{D}$ for energy at time T. The energy price at T is also a random variable given by $\hat{P}^E$. The forward price for gas at t is denoted by P_t^G and its increment is denoted by $\hat{P}_t^G$. The demand is independent of the energy and gas prices. We have that $W_{t-1}^x = P_{t-1}^G$, while

$$W_t = (P_t^G, 0, 0) 1_{\{t<T\}} + (P_t^G, \hat{D}, \hat{P}^E) 1_{\{t=T\}}$$

The expression for $\hat{W}_t$ is obtained replacing P_t^G for $\hat{P}_t^G$ in the expression for W_t.

The decision x_t is a scalar denoting the number of forward contracts purchased at t. We have that R_t (R_t^x) is the amount of contracts purchased up until time t before (after) the decision at t is taken. Clearly, $R_t = R_{t-1}^x$ and $R_t^x = R_t + x_t$.

Function f_1 can take different forms, depending on the price process. In section VI, we will consider three different forms. It follows that $f_2(R, \hat{W}_t) = g(W_t, R) = R$ and A is a scalar equal to 1.

The states S_t and S_t^x are given by the current prices/demand and the amount of contracts purchased so far. The constraint set is given by $\mathcal{X}(S_t) = \{0 \leq x_t \leq M_t\}$.

The contribution in period t, for $t = 1, \ldots, T$, is given by

$$C_t(S_t, x_t) = -P_t^G x_t 1_{\{t<T\}} + \hat{P}^E \min(\hat{D}, R_T) 1_{\{t=T\}}.$$

Finally, the random variable $\hat{G}_t(S_t, v_t^*)$ is given by

$$\begin{aligned} \hat{G}_t^*(S_t, v_t^*) = \max\left(\min\left(P_t^G, v_t^*(P_t^G, R_t)\right), v_t^*(P_t^G, R_t + M_t)\right) \\ \times 1_{\{t<T\}} + \hat{P}^E 1_{\{\hat{D} \geq R_T\}} 1_{\{t=T\}}. \end{aligned}$$

B. Inventory System

At each period, the inventory manager must determine the number of assets to sell and the number of assets to be purchased in order to replenish the inventory, given the current inventory level. There are no holding costs and unsatisfied demand is lost. Moreover, assets not sold remain in the system. However, inventory items can be stolen (or otherwise lost from the system).

The exogenous process is given by $\hat{W}_t = (\hat{P}_t^s, \hat{P}_t^r, \hat{D}_t, \hat{R}_t)$, denoting selling and replenishing price increments, demand increments and missing items, respectively. The information variable W_t is given by $(P_t^s, P_t^r, D_t, \hat{R}_t)$, while W_t^x is given by (P_t^s, P_t^r, D_t), as the number of missing items is independent of any previous information. The demand D_t is assumed to be integer valued and the selling price is greater than the replenishing price. Like the former application, function f_1 can take different forms.

The decision $x_t = (x_{t1}, x_{t2})$ is the amount of sold assets and the amount of purchased assets to replenish the inventory, respectively. The state variable R_t (R_t^x) denotes the inventory level before (after) the decision. We have that

$$R_t = \max(0, R_{t-1}^x - \hat{R}_t) \text{ and } R_t^x = R_t - x_{t1} + x_{t2},$$

implying $f_2(R, \hat{W}_t) = \max(0, R - \hat{R}_t)$, $g(W_t, R) = R$ and the input-output vector $A = (-1, 1)$.

The constraint set is

$$\mathcal{X}(S_t) = \{0 \le x_{t1} \le \min(D_t, R_t), 0 \le x_{t2} \le M_t\}.$$

The contribution in each period is

$$C_t(S_t, x_t) = P^s_t x_{t1} - P^r_t x_{t2},$$

and the random variable $\hat{G}_t(S_t, v^*_t)$ is

$$\begin{aligned}\hat{G}_t(S_t, v^*_t) &= v^*_t(W^x_t, R_t + M_t)1_{\{v^*_t(W^x_t, R_t+M_t)>P^s_t\}}\\ &+ P^s_t 1_{\{v^*_t(W^x_t, R_t+M_t)\le P^s_t\}}\\ &\quad\times [1_{\{D_t \ge R_t\}} + 1_{\{D_t<R_t\}}1_{\{v^*_t(W^x_t, R_t-D_t+M_t)>P^s_t\}}]\\ &+ \min\left(P^r_t, v^*_t(W^x_t, R_t - D_t)\right)1_{\{D_t<R_t\}}1_{\{v^*_t(W^x_t, R_t-D_t+M_t)\le P^r_t\}}\\ &+ v^*_t(W^x_t, R_t - D_t + M_t)1_{\{D_t<R_t\}}1_{\{P^r_t<v^*_t(W^x_t, R_t-D_t+M_t)\le P^s_t\}}.\end{aligned}$$

IV. The SPAR-MultiPeriod Algorithm

We propose an algorithm that makes use of the properties of our problem class in order to learn an optimal policy. It combines Monte Carlo simulation in a pure exploitation scheme, in order to construct concave piecewise linear functions approximations. The function slopes are updated through stochastic approximation integrated with a projection operation.

The SPAR-MultiPeriod algorithm is presented in figure 2. As described in STEP 0, the algorithm inputs are piecewise linear value function approximations represented by their slopes $\bar{v}^0$ and a starting value for the pre-decision state at 0. The initial slopes must be decreasing in the asset dimension and bounded between $-B$ and B. A slope vector that is equal to zero for all states and time periods is a valid input. We refer interchangeably to the value function itself $\bar{V}^n_t(W^x_t, \cdot)$ and its slopes $\bar{v}^n_t(W^x_t)$. We assume that $\bar{v}^n_T(S^x_T) = 0$ for all iterations n and states $S^x_T \in \bar{\mathcal{S}}_T$.

At each iteration n the algorithm samples $\hat{W}^n_0, \ldots, \hat{W}^n_t$, as shown in STEP 1. Then, at each time period t, a decision x^n_t is taken. This decision is the optimal solution within the constraint set $\mathcal{X}(S^n_t)$, with respect to the current state S^n_t and value function approximation $\bar{V}^{n-1}_t(W^{x,n}_t, \cdot)$, as stated in STEP 2a. The decision x^n_t brings the system to the post-decision asset level $R^{x,n}_t = g(S^n_t) + A \cdot x^n_t$, as shown in STEP 2b. Then, the exogenous information for period $t+1$ becomes available and the next pre-decision state is computed following STEP 2c. Sample slopes are observed (see STEP 2d) and are used to update the approximation slopes $\bar{v}^{n-1}_t(W^{x,n}_t)$ (see STEP 2e). This step requires the use of a stepsize rule that is state dependent, denoted by $\bar{\alpha}^n_t(S^x_t)$. The updating procedure leads to a temporary slope vector z^n_t that may violate the concavity property. Thus, a projection operation is performed (STEP 2f). Then, the iteration counter is incremented and the algorithm is repeated.

A general post-decision state at t is denoted by S^x_t or (W^x_t, R^x_t). The two of them are used interchangeably. We use $S^{x,n}_t$ to denote the actual state visited by the algorithm at iteration n and time t. Furthermore, $\{S^{x,n}_t\}_{n\ge 0}$ is the sequence of states visited by the algorithm. The same notation convention holds for the pre-decision states S_t, S^n_t and $\{S^n_t\}_{n\ge 0}$ and the decisions x_t, x^n_t and $\{x^n_t\}_{n\ge 0}$. The sequences of slopes are

```
STEP 0: Initialization:
    STEP 0a: Initialize $\bar{v}^0_t(S^x_t)$ for all $t$ and $S^x_t$ to be
             monotone decreasing in $R^x_t$.
    STEP 0b: Initialize $S^n_0$, for $n \ge 0$.
    STEP 0c: Set $n = 1$.
STEP 1: Sample the exogenous process $\hat{W}^n_0, \ldots, \hat{W}^n_T$.
STEP 2: Do for $t = 0, \ldots, T$:
    STEP 2a: Find the optimal solution $x^n_t$ of
             $\max_{x \in \mathcal{X}(S^n_t)} C_t(S^n_t, x) + \gamma \bar{V}^{n-1}_t(W^{x,n}_t, R^x_t)$.
    If $t < T$ then
    STEP 2b: Find the post-decision state
             $(W^{x,n}_t, R^{x,n}_t) = (W^{x,n}_t, g(S^n_t) + A \cdot x^n_t)$.
    STEP 2c: Find the next pre-decision state
             $(W^n_{t+1}, R^n_{t+1}) = (f_1(W^{x,n}_t, \hat{W}^n_{t+1}), f_2(R^{x,n}_t, \hat{W}^n_{t+1}))$.
    STEP 2d: Observe $\hat{v}^n_{t+1}(R^{x,n}_t)$ and $\hat{v}^n_{t+1}(R^{x,n}_t + 1)$.
    STEP 2e: For $S^x_t \in \bar{\mathcal{S}}_t$,
             $z^n_t(S^x_t) = (1 - \bar{\alpha}^n_t(S^x_t))\bar{v}^{n-1}_t(S^x_t) + \bar{\alpha}^n_t(S^x_t)\hat{v}^n_{t+1}(R^x_t)$
    STEP 2f: $\bar{v}^n_t = \Pi_{\mathcal{C}}(z^n_t)$.
STEP 3: Increase $n$ by one and go to step 1.
```

Fig. 2. SPAR-MultiPeriod Algorithm

denoted by $\{\bar{v}^n_t(S^x_t)\}_{n\ge 0}$. It is straightforward to see that all these sequences generated by the algorithm have at least one accumulation point. We denote by $S^{x,*}_t$, S^*_t, x^*_t and $\bar{v}^*_t(S^x_t)$ the respective accumulation points.

We obtain sample slopes by replacing the expectation and the optimal slope v^*_t in (1) by the sample realization of the information W^n_t and the current approximation $\bar{v}^{n-1}_t$, respectively. Thus, for $t = 1 \ldots, T$, the sample slope is

$$\hat{v}^n_t(R) = \hat{G}_t\left(f_1(W^{x,n}_{t-1}, \hat{W}^n_t), f_2(R, \hat{W}^n_t)), \bar{v}^{n-1}_t\right). \tag{2}$$

The projection operator $\Pi_{\mathcal{C}}$ maps a vector z^n_t that may not be monotone decreasing in the asset dimension, into another vector $\bar{v}^n_t$, such that, for $W^x_t \in \mathcal{W}_t$, $(\bar{v}^n_t(W^x_t, 1), \ldots, \bar{v}^n_t(W^x_t, B_t)) \in \mathcal{C}_t$. A slight variation of the *Level* projection operator (introduced in [11]) is used. It imposes concavity by forcing the newly updated slope at $(W^n_t, R^{x,n}_t)$ to be greater or equal to the newly updated slope at $(W^n_t, R^{x,n}_t + 1)$ and then forcing the other violating slopes to be equal to the newly updated ones. Then, for $S^x_t \in \bar{\mathcal{S}}_t$, the projection is given by

$$\Pi_{\mathcal{C}}(z^n_t)(S^x_t) = \begin{cases} \frac{z^n_t(W^{x,n}_t, R^{x,n}_t) + z^n_t(W^{x,n}_t, R^{x,n}_t+1)}{2}, & \text{if C1} \\ \Pi_{\mathcal{C}}(z^n_t)(W^{x,n}_t, R^{x,n}_t), & \text{if C2} \\ \Pi_{\mathcal{C}}(z^n_t)(W^{x,n}_t, R^{x,n}_t + 1), & \text{if C3} \\ z^n_t(S^x_t), & \text{otherwise,} \end{cases} \tag{3}$$

where the conditions C1, C2 and C3 are

C1: $W^x_t = W^{x,n}_t, R^x_t = (R^{x,n}_t \text{ or } R^{x,n}_t + 1),$
$\qquad z^n_t(W^{x,n}_t, R^{x,n}_t) < z^n_t(W^{x,n}_t, R^{x,n}_t + 1);$

C2: $W^x_t = W^{x,n}_t, R^x_t < R^{x,n}_t, z^n_t(S^x_t) \le \Pi_{\mathcal{C}}(z^n_t)(W^{x,n}_t, R^{x,n}_t);$

C3: $W^x_t = W^{x,n}_t, R^x_t > R^{x,n}_t + 1, z^n_t(S^x_t) \ge \Pi_{\mathcal{C}}(z^n_t)(W^{x,n}_t, R^{x,n}_t + 1).$

We move on to the stepsizes $\bar{\alpha}_t^n(S_t^x)$. Let

$$\bar{\alpha}_t^n(S_t^x) = \alpha_t^n 1_{\{W_t^x = W_t^{x,n}\}}(1_{\{R_t^x = R_t^{x,n}\}} + 1_{\{R_t^x = R_t^{x,n}+1\}}),$$

where α_t^n is a scalar between 0 and 1, measurable with respect to $\mathcal{F}_t^n$ and has the following properties

$$\sum_{n=0}^{\infty}(\alpha_t^n)^2 \le \bar{B} < \infty \quad \text{and} \quad \sum_{n=0}^{\infty}\alpha_t^n 1_{\{S_t^{x,n}=S_t^{x,*}\}} = \infty \quad \text{a.s.,}$$

where $\bar{B}$ is a constant and $S_t^{x,*}$ is an accumulation point of the sequence $\{S_t^{x,n}\}_{n\ge 0}$. Clearly, the stepsize rule $\alpha_t^n = \frac{1}{N(S_t^{x,n})}$ satisfies all the conditions, where $N(S_t^{x,n})$ is the number of visits to state $S_t^{x,n}$ up until iteration n.

V. Convergence Analysis

The dynamic programming operator H maps a vector v into a new vector Hv. For $t = 0, \dots, T$ and $S_t^x \in \bar{\mathcal{S}}_t$,

$$(Hv)_t(S_t^x) = \begin{cases} \mathbb{E}\left[\hat{G}_{t+1}(S_{t+1}, v_{t+1})|S_t^x\right], & \text{if } t < T \\ 0, & \text{if } t = T. \end{cases}$$

Clearly, the vector of slopes of the optimal value functions v^* is the unique fixed point of H.

Using the operator H, we define two deterministic bounding sequences. For all $k \ge 0$ and states $S_T^x \in \bar{\mathcal{S}}_T$, $L_T^k(S_T^x) = U_T^k(S_T^x) = 0$. For $t = 0, \dots, T-1$ and states $S_t^x \in \bar{\mathcal{S}}_t$,

$$L_t^0(S_t^x) = v_t^*(S_t^x) - 2B \quad \text{and} \quad U_t^0(S_t^x) = v_t^*(S_t^x) + 2B,$$

and, for all $k \ge 0$,

$$L_t^{k+1}(S_t^x) = \frac{L_t^k(S_t^x) + (HL^k)_t(S_t^x)}{2}$$
$$U_t^{k+1}(S_t^x) = \frac{U_t^k(S_t^x) + (HU^k)_t(S_t^x)}{2}.$$

Under very mild assumptions on H, $\{L_t^k(S_t^x)\}_{k\ge 0}$ is a increasing sequence that bounds $v_t^*(S_t^x)$ from below while $\{U_t^k(S_t^x)\}_{k\ge 0}$ is a decreasing sequence that bounds $v_t^*(S_t^x)$ from above. Moreover, both sequences converge to $v_t^*(S_t^x)$.

Having defining these elements, we are ready to introduce our main theorem. We give a summary of its proof.

Theorem 1: For $k \ge 0$ and $t = 0, \dots, T$, there exists an integer $N_t^{*,k}$ such that, for all $n \ge N_t^{*,k}$ and $\bar{S}_t^{x,*} \in \bar{\mathcal{S}}_t^*$,

$$L_t^k(\bar{S}_t^{x,*}) \le \bar{v}_t^{n-1}(\bar{S}_t^{x,*}) \le U_t^k(\bar{S}_t^{x,*}) \quad \text{a.s.,} \tag{4}$$

where $\bar{\mathcal{S}}_t^*$ is the set of all states that are either equal to an accumulation point $(W_t^{x,*}, R_t^{x,*})$ of $\{S_t^{x,n}\}_{n\ge 0}$ or are equal to $(W_t^{x,*}, R_t^{x,*}+1)$ and $0 < R_t^{x,*} < B_t$. Therefore,

$$\bar{v}_t^n(\bar{S}_t^{x,*}) \to v_t^*(\bar{S}_t^{x,*}) \text{ a.s.}$$

Proof: The proof of the theorem is by backward induction on t. The base case is $t = T$. As $v_T^*(S_T^x) = U_T^k(S_T^x) = L_T^k(S_T^x) = \bar{v}_T^n(S_T^x) = 0$ for all states $S_T^x \in \bar{\mathcal{S}}_T$, integers $k \ge 0$ and iterations $n \ge 0$, the inequalities in (4) are trivial for $t = T$. In particular, we can take $N_T^{*,k} = \bar{N}$.

The backward induction proof is completed when we prove (4) for a general t, $t = 0, \dots, T-1$. Given the induction hypothesis for $t+1$, the proof for time period t is divided into two parts. In the first part, we prove for all $k \ge 0$ that there exists an integer N_t^k such that, for all $n \ge N_t^k$,

$$\bar{v}_t^{n-1}(\tilde{S}_t^{x,*}) \ge L_t^k(\tilde{S}_t^{x,*}), \quad \text{if } \tilde{S}_t^{x,*} \in \tilde{\mathcal{S}}_t^+ \tag{5}$$
$$\bar{v}_t^{n-1}(\tilde{S}_t^{x,*}) \le U_t^k(\tilde{S}_t^{x,*}), \quad \text{if } \tilde{S}_t^{x,*} \in \tilde{\mathcal{S}}_t^-. \tag{6}$$

Note that these inequalities only apply to states in the sets $\tilde{\mathcal{S}}_t^-$ and $\tilde{\mathcal{S}}_t^+$, which are proper subsets of $\bar{\mathcal{S}}_t^*$. A state S_t^x is in $\tilde{\mathcal{S}}_t^-$ ($\tilde{\mathcal{S}}_t^+$) if the projection operation decreased (increased) or kept the same the corresponding temporary slope infinitely often, i.e., $z_t^n(S_t^x) \le \bar{v}_t^n(S_t^x)$ for an infinite number of iterations n.

The proof of inequalities (5-6) is by induction on k and requires several intermediate results. We concentrate on the main elements of the proof of (5). The arguments for (6) are symmetrical.

We omit the base case, $k = 0$, and assume there exists N_t^k such that (5) holds for $n \ge N_t^k$. We shall show the existence of N_t^{k+1} such that $\bar{v}_t^{n-1}(\tilde{S}_t^{x,*}) \ge L_t^{k+1}(\tilde{S}_t^{x,*})$, for $n \ge N_t^{k+1}$.

We define a stochastic bounding sequence, namely, $\{\bar{l}_t^n(S_t^x)\}_{n\ge 0}$, for each $S_t^x \in \bar{\mathcal{S}}_t$. For $n \le N_t^k$, let $\bar{l}_t^n(S_t^x) = L_t^k(S_t^x)$, and, for $n \ge N_t^k$, let

$$\bar{l}_t^n(S_t^x) = (1 - \bar{\alpha}_t^n(S_t^x))\,\bar{l}_t^{n-1}(S_t^x) + \bar{\alpha}_t^n(S_t^x)(HL^k)_t(S_t^x).$$

A simple inductive argument proves that $\bar{l}_t^n(S_t^x)$ is a convex combination of $L_t^k(S_t^x)$ and $(HL^k)_t(S_t^x)$. Using this fact, it can be shown, for all n big enough, that

$$\bar{l}_t^{n-1}(\tilde{S}_t^{x,*}) \ge L_t^{k+1}(\tilde{S}_t^{x,*}) + \delta^k, \tag{7}$$

where δ^k is a positive number.

We also define a variable $\hat{s}_{t+1}^n$, that represents the error incurred by observing a sample slope instead of a true expectation. For $R = 1, \dots, B_t$,

$$\hat{s}_{t+1}^n(R) = (H\bar{v}^{n-1})_t(W_t^{x,n}, R) - \hat{v}_{t+1}^n(R).$$

For each $S_t^x \in \bar{\mathcal{S}}_t$, we use $\hat{s}_{t+1}^n$ to define a stochastic noise sequence represented by $\{\bar{s}_t^n(S_t^x)\}_{n\ge 0}$. For $n \le N_t^k$, let $\bar{s}_t^n(S_t^x) = 0$, and, for $n \ge N_t^k$, let

$$\bar{s}_t^n(S_t^x) = \max(0, (1 - \bar{\alpha}_t^n(S_t^x))\,\bar{s}_t^{n-1}(S_t^x) + \bar{\alpha}_t^n(S_t^x)$$
$$\times\, \hat{s}_{t+1}^n(R_t^{x,n} 1_{\{R \le R_t^{x,n}\}} + (R_t^{x,n}+1)1_{\{R > R_t^{x,n}\}})).$$

The sample slopes were defined in a way such that

$$\mathbb{E}\left[\hat{s}_{t+1}^n(R)|\mathcal{F}_t^n\right] = 0.$$

This last equation, the stepsize assumptions, the martingale convergence theorem and the boundedness of both the sample slopes and the approximate slopes are crucial for proving that the noise introduced by the observation of the sample slopes, which replace the observation of true expectations, go to zero as the number of iterations of the algorithm goes to infinity, i.e., for all $\bar{S}_t^{x,*} \in \bar{\mathcal{S}}_t^*$, $\{\bar{s}_t^n(\bar{S}_t^{x,*})\}_{n\ge 0} \to 0$ a.s.

Using the concavity of the value function approximations it can also be shown, for all n big enough, that

$$\bar{v}_t^{n-1}(\tilde{S}_t^{x,*}) \ge \bar{l}_t^{n-1}(\tilde{S}_t^{x,*}) - \bar{s}_t^{n-1}(\tilde{S}_t^{x,*}). \tag{8}$$

Combining (7), (8) and convergence to zero of the stochastic noise sequence, we obtain, for all n big enough that

$$\bar{v}_t^{n-1}(\tilde{S}_t^{x,*}) \ge L_t^{k+1}(\tilde{S}_t^{x,*}) + \delta^k - \delta^k$$
$$= L_t^{k+1}(\tilde{S}_t^{x,*}),$$

proving the existence of a integer N_t^{k+1} such that (5) is true for $k+1$.

Still considering time period t, we take on the second part, which proves the existence of an integer $N_t^{*,k}$ such that (4) is true for all states in $\bar{\mathcal{S}}_t^*$ and iterations $n \geq N_t^{*,k}$. In this part, we take care of the states $\bar{S}_t^{x,*} \in \bar{\mathcal{S}}_t^* \backslash (\tilde{\mathcal{S}}_t^+ \bigcap \tilde{\mathcal{S}}_t^-)$, In contrast to part 1, the proof technique here is not by forward induction on k and depends heavily on the projection operation.

An important property of the projection operator is that all the slopes to the left of $R_t^{x,n}$, changed by the projection operation are increased to be equal to the new slope at $R_t^{x,n}$. Similarly, all the slopes to the right of $R_t^{x,n}+1$, changed by the projection operation, are decreased to be equal to the slope at $R_t^{x,n}+1$. Another property, that is not hard to prove, is that for states $(\bar{W}_t^{x,*}, \bar{R}_t^{x,*}) \in \bar{\mathcal{S}}_t^* \setminus \tilde{\mathcal{S}}_t^+$, the state $(\bar{W}_t^{x,*}, \tilde{R}_+^{x,*})$ where $\tilde{R}_+^{x,*}$ is the maximum asset level smaller than $\bar{R}_t^{x,*}$ such that $(\bar{W}_t^{x,*}, \tilde{R}_+^{x,*}) \in \tilde{\mathcal{S}}_t^+$ is well defined. Moreover, the projection operation decreased the slopes of $(\bar{W}_t^{x,*}, R)$ infinitely many times, where $\tilde{R}_+^{x,*} \leq R \leq \bar{R}_t^{x,*}$.

These projection properties and the result of part 1 are then used to show the existence of a integer $N_t^{*,k}$ such that, for all $n \geq N_t^{*,k}$, $L_t^k(\bar{S}_t^{x,*}) \leq \bar{v}_t^{n-1}(\bar{S}_t^{x,*})$. The inequality for $U_t^k(\bar{S}_t^{x,*})$ is obtained using a symmetrical property of the projection operation. ■

We finish the convergence analysis proving that, with probability one, the algorithm learns an optimal decision for all states that can be reached by an optimal policy.

Theorem 2: For $t = 0, \ldots, T$, let $(S_t^*, \bar{v}_t^*, x_t^*, W_t^{x,*})$ be an accumulation point of the sequence $\{(S_t^n, \bar{v}_t^{n-1}, x_t^n, W_t^{x,n})\}_{n \geq 1}$ generated by the algorithm. With probability one, x_t^* is an optimal solution of

$$\max_{x \in \mathcal{X}(S_t^*)} \underbrace{C_t(S_t^*, x) + \gamma V_t^*(W_t^{x,*}, g(S_t^*) + A \cdot x)}_{F_t(S_t^*, V_t^*, x)}. \quad (9)$$

Proof: The solution x_t^n in STEP 2a of the algorithm, as it is optimal, satisfies

$$0 \in \partial F_t(S_t^n, \bar{V}_t^{n-1}, x_t^n) + \mathcal{X}^N(S_t^n, x_t^n),$$

where $\partial F_t(S_t^n, \bar{V}_t^{n-1}, x_t^n)$ is the subdifferential of $F_t(S_t^n, \bar{V}_t^{n-1}, \cdot)$ at x_t^n and $\mathcal{X}^N(S_t^n, x_t^n)$ is the normal cone of $\mathcal{X}(S_t^n)$ at x_t^n.

Then, by passing to the limit, we can conclude that each accumulation point $(S_t^*, \bar{v}_t^*, x_t^*)$ of the sequence $\{(S_t^n, \bar{v}_t^{n-1}, x_t^n)\}_{n \geq 1}$ satisfies the condition $0 \in \partial F_t(S_t^*, \bar{V}_t^*, x_t^*) + \mathcal{X}^N(S_t^*, x_t^*)$.

Let $R_t^{x,*} = g(S_t^*) + A \cdot x_t^*$. It is not hard to show that

$$\begin{aligned} &\partial F_t(S_t^*, \bar{V}_t^*, x_t^*) = \\ &\quad \{(c_{t1}(S_t^*) + \gamma A_1 y, \ldots, c_{tl}(S_t^*) + \gamma A_l y)^T : \\ &\quad y \in [\bar{v}_t^*(W_t^*, R_t^{x,*}+1), \bar{v}_t^*(W_t^*, R_t^{x,*})]\}. \end{aligned}$$

Since $(W_t^{x,*}, R_t^{x,*})$ is an accumulation point of $\{S_t^{x,n}\}_{n \geq 0}$, it follows from theorem 1 that

$$\begin{aligned} \bar{v}_t^*(W_t^{x,*}, R_t^{x,*}) &= v_t^*(W_t^{x,*}, R_t^{x,*}) \quad \text{a.s.} \\ \bar{v}_t^*(W_t^{x,*}, R_t^{x,*}+1) &= v_t^*(W_t^{x,*}, R_t^{x,*}+1) \quad \text{a.s.,} \end{aligned}$$

proving the theorem. ■

VI. Experimental Results

We want to establish the computational benefits of our algorithm relative to other Monte Carlo-based algorithms as well as classical backward dynamic programming. We provide a short description of each approach.

In a Batch-mode Monte-Carlo-based value iteration algorithm (Batch), at each iteration n, once a sample for the information process is gathered, sample slopes at all possible asset levels R_t^x are observed and used to update the corresponding slopes. That is, steps 2d and 2e are replaced by

STEP 2d: Observe $\hat{v}_{t+1}^n(R_t^x)$ for all $R_t^x \in \{1, \ldots, B_t\}$.
STEP 2e: For $S_t^x \in \mathcal{S}_t$,

$$z_t^n(S_t^x) = (1 - \alpha_t^n 1_{\{W_t^x = W_t^{x,n}\}}) \bar{v}_t^{n-1}(S_t^x) + \alpha_t^n 1_{\{W_t^x = W_t^{x,n}\}} \hat{v}_{t+1}^n(R_t^x).$$

Applying this method, which is synchronous in the sense that all the slopes for the observed $W_t^{x,n}$ are updated at once, we wish to see how it compares to an asynchronous approach (our algorithm).

A Real Time Dynamic Programming (RTDP) approach [10] assumes the knowledge of the probability distribution. Instead of using the sample slope given by (2), our RTDP algorithm uses

$$\hat{v}_{t+1}^n(R_t^x) = (H\bar{v}^n)_t(W_t^{x,n}, R_t^x).$$

When we compare this algorithm against ours, we are measuring the tradeoff between more information given by the expectation versus the time spent to do this operation.

A standard Q-learning algorithm [4] stores all possible State-Action pairs making this approach impossible to be implemented for our problem class. Therefore, instead of implementing a Q-learning approach, we propose and implement an algorithm that only stores the state after the decision is made and samples all possible actions infinitely often in a uniform way. This implies that all states are sampled infinitely often as well. This algorithm should be at least as good as standard Q-learning, due to a smaller state space. Step 2a of our algorithm thus is replaced by

STEP 2a: Sample x_t^n according to a discrete uniform distribution between 0 and M_t.

With this algorithm, we try to infer how a pure exploitation scheme (our approach) compares to a pure exploration one.

We show experimental results for some instances of the forward contract application described in section III. The considered instances are described in table I. We randomly generated problems using different distributions for the energy price $\hat{P}^E$ and for the initial prices of forward contracts P_0^G. Moreover, both discrete uniform (DiscU) and Poisson demand distributions with different parameters were used.

We also created different processes for the prices of the forward contracts, namely random walk (RW), mean reversion (MR) and geometric Brownian motion (GBM), all of which are described below. Even though all price processes are continuous, we use a discretization increment of 0.1 for all instances. We emphasize that when using our algorithm, the discretization only occurs when representing the value functions, the simulated paths are still continuous.

For all instances, the number of time periods is 10, implying that both the demand $\hat{D}$ and the energy price $\hat{P}^E$ are observed at $T = 10$. Moreover, the initial number of forward contracts is $R_0 = 0$ and the amount of contracts that can be purchased are bounded by 0 and 400, that is, $0 \leq x_t \leq M_t = 400$. The state space size of each instance is also shown in table I.

TABLE I
INSTANCES DESCRIPTION

Inst.	State Space	Initial Price	Energy Price	Demand	Price Proc.
1	580000	Constant 20	U(50,60)	DiscU(180,250)	RW
2	580000	Constant 20	U(50,60)	Poisson(200)	RW
3	4114800	1.7*U(1,12)	P_{T-1}*U(1.03,1.15)	Poisson(250)	MR
4	4114800	1.7*U(1,12)	P_{T-1}*U(1.03,1.15)	DiscU(180,220)	MR
5	1608000	Constant 40	Constant 25	Poisson(300)	GBM
6	1608000	Constant 45	Constant 15	DiscU(225,375)	GBM

The random walk price process is given by $P_t^G = P_{t-1}^G + \hat{P}_t^G$, where the price increment $\hat{P}_t^G$ has the normal distribution with mean $\mu = 0.02$ and standard deviation $\sigma = 1.5$.

The mean reversion price process is given by $P_t^G = P_{t-1}^G + \hat{P}_t^G + 0.5(B_t - P_{t-1}^G)$, where $\hat{P}_t^G$ is uniformly distributed between 0.9 and 1.2. Moreover, B_t is a deterministic process where $B_0 = 1.7\bar{U}(1, 12)$, $B_t = B_{t-1}\bar{U}(0.9, 1.2)$ and $\bar{U}$ is the mean of the corresponding uniform distribution.

Finally, the geometric Brownian motion process is given by $P_t^G = P_{t-1}^G e^{\hat{P}_t^G}$, where $\hat{P}_t^G$ is normally distributed with mean $\mu = 0.0125$ and standard deviation $\sigma = 0.087$.

It can be shown that when the random walk and the geometric Brownian motion are considered, the optimal slopes v_t^* are monotone increasing in P_t^G. Therefore, in order to improve the rate of convergence, the algorithms enforce this property while updating the approximating slopes when instances 1, 2, 5, 6 are considered.

Knowing the underlying distributions, as described in table I, we computed a policy using a classical backward dynamic programming (CDP) technique. A discretization increment of 0.01 was used. We also computed a policy using our algorithm (ADP) and the other approximate methods described in the beginning of this section. Except for the RTDP method, the distributions were assumed unknown.

Table II shows the time (in seconds) that took each method to be $10\%, 1\%, \ldots, 10^{-3}\%$ away from the solution given by the classical dynamic programming technique. It also shows the time to compute the CDP solution. All approximate methods were limited to 2 million iterations.

The error is measured according to

$$\eta^n = \frac{|\bar{F}^* - \bar{F}^n|}{\bar{F}^*} \times 100, \tag{10}$$

where $\bar{F}^*$ is the average value of following the CDP policy while $\bar{F}^n$ is the average value of following the policy obtained after n iterations of the given approximate algorithm. Both averages were taken considering 800 randomly generated sample paths.

The results for the method that samples uniformly the actions are not displayed in the table, since more than 2 million iterations of this method were required to be within 10% from the CDP solution, implying that using pure exploitation instead of pure exploration pays off.

TABLE II
TIME IN SECONDS TO REACH SOLUTION QUALITY LEVELS

Instance	Method	% away from CDP				
		10^1	10^0	10^{-1}	10^{-2}	10^{-3}
1	ADP	0.25	0.25	1.62	14.83	–
CDP Time	Batch	0.27	0.27	2.39	30.02	–
6.92×10^3	RTDP	0.81	0.81	2.65	–	–
2	ADP	0.26	0.26	0.26	6.66	–
CDP Time	Batch	0.21	0.21	0.21	–	–
6842.21 secs	RTDP	0.81	0.81	0.81	20.36	–
3	ADP	3.69	29.72	460.79	–	–
CDP Time	Batch	5921.77	–	–	–	–
7658.94 secs	RTDP	21.88	30.7	–	–	–
4	ADP	3.46	20.62	691.25	–	–
CDP Time	Batch	11816.34	–	–	–	–
7548.53 secs	RTDP	20.96	29.38	–	–	–
5	ADP	10.53	17.12	27.77	46.83	216.48
CDP Time	Batch	129	194.13	–	–	–
24149.52 secs	RTDP	406.34	677.7	812.85	812.85	948.04
6	ADP	0.34	4.9	9.52	206.47	236.63
CDP time	Batch	12.46	31.93	–	–	–
10766.61 secs	RTDP	9.18	376.77	376.77	564.38	–

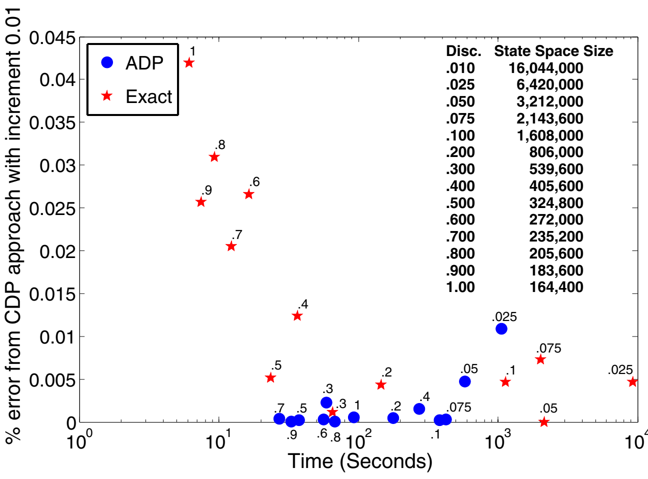
Instances 3 and 4 did not reach the .01% level. Since these instances use the mean reversion price process the algorithm could not impose the monotone increasing property of the slopes in the information dimension. Hence, we can infer that the more structure we impose, the more we speed up the rate of convergence.

We can also see that the computational time for the Batch approach is much higher than the computational time of our approach. Even though the Batch method makes better use of the information in each sample realization, since all the slopes associated with the given price realization are updated, it does not translate into better solutions in competitive time, showing that our asynchronous algorithm performs better than the synchronous one.

The same behavior was observed with the RTDP approach. More information given by the expectation instead of a sample realization does not result in an improvement in the solutions, when the same amount of time is considered for both our algorithm and the RTDP approach.

We can also infer from table II the computational advantage of using our method instead of a classical backward dynamic programm algorithm. For the instances considered in the experiment, even with a scalar decision and a scalar controllable state R_t^x, we can easily run into a curse of dimensionality. Thus, our algorithm is a reasonable method of choice even when the probability distribution of the underlying stochastic process is known, since we can obtain pretty good policies within very small computational time.

We now analyze the influence of the discretization of the forward prices when computing an exact policy and when computing a policy with our algorithm. Figure 3 shows the percentage away from the CDP solution (computed for increment 0.01), computed using 10, as a function of time for levels of discretization that go from 0.025 to 1. It also shows the corresponding state space size. The problem instance used in this experiment used the same settings as instance 5, except of course that the level of discretization varied. The discretization increment is shown next to the corresponding error. The results are for our method and for an exact dynamic programming algorithm that assumes the distributions are known.



Disc.	State Space Size
.010	16,044,000
.025	6,420,000
.050	3,212,000
.075	2,143,600
.100	1,608,000
.200	806,000
.300	539,600
.400	405,600
.500	324,800
.600	272,000
.700	235,200
.800	205,600
.900	183,600
1.00	164,400

Fig. 3. Quality versus time (in log scale) frontier for different levels of discretization.

We can see from this figure that for larger discretization increments, our method produced better results than the exact method with competitive computational times. This can be explained by the fact that, for our method, the discretization only occurs when representing the value functions. On the other hand, when computing the expectations for the exact method, the prices are indeed discrete and the probability mass of a whole interval is assigned to one point.

We can also observe that the smaller the increment, the more time it took the exact method to come up with a solution, since the expectation operation becomes more expensive (note that a log scale is used for time). For example, there is a substantial increase in CPU times when the discretization is reduced from $.05$ to $.025$. The computational time of our method does increase, but for a different reason. As the discretization becomes finer, our algorithm requires more iterations to produce better results, since we only update the value function approximation for the observed price. These results emphasize the previous conclusion that our method is a reasonable choice even for problems with known probability distributions. The only caveat is that the discretization increment should be wisely chosen, as there is a clear tradeoff between accuracy and time to obtain a good policy.

VII. Conclusions

We proposed an approximate dynamic programming algorithm for a class of multistage stochastic control problems. The state variable of the problems in this problem class is decomposed into a multi-dimensional information component and a controllable scalar component. Moreover, the optimal value functions are piecewise linear and concave in the controllable component, reflecting diminishing gains as the controllable scalar is increased. The distribution of the stochastic process generating the information component may be unknown.

Our algorithm is a combination of Monte Carlo simulation, stochastic approximation and a projection operation. It converges to an optimal policy and scales to high-dimensional problems. The main idea is to learn the slopes of the optimal value functions only for important regions of the state space, which are determined by the algorithm itself.

We demonstrated experimentally that our approach outperforms other provably convergent approximate approaches in the sense that less computational time is required in order to find a good policy. We also showed that our method is a good choice even when the distribution is known and classical backward dynamic programming could be applied. Finally, we also addressed the discretization of the state space issue and showed that our method handles it in a much better way than an exact method.

References

[1] G. van Ryzin and J. McGill, "Revenue management without forecasting or optimization: An adaptive algorithm for determining airline seat protection levels," *Management Science*, vol. 46, no. 6, pp. 760–775, June 2000.
[2] P. Rusmevichientong, B. Van Roy, and P. Glynn, "A non-parametric approach to multi-product pricing," *Operations Research*, vol. 54, no. 1, pp. 82–98, 2006.
[3] R. Levi, R. Roundy, and D. Shmoys, "Provably near-optimal sampling-based policies for stochastic inventory control models," 2006, submitted for publication.
[4] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
[5] W. B. Powell, A. Ruszczyński, and H. Topaloglu, "Learning algorithms for separable approximations of stochastic optimization problems," *Mathematics of Operations Research*, vol. 29, no. 4, pp. 814–836, 2004.
[6] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[7] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, pp. 185–202, 1994.
[8] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, Inc., 1994, pp. 703–710. [Online]. Available: citeseer.nj.nec.com/jaakkola94convergence.html
[9] J. Tsitsiklis, "On the convergence of optimistic policy iteration," *J. of Machine Learning Research*, vol. 3, pp. 59–72, 2002.
[10] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence, Special Volume on Computational Research on Interaction and Agency*, vol. 72, pp. 81–138, 1995.
[11] H. Topaloglu and W. B. Powell, "An algorithm for approximating piecewise linear concave functions from sample gradients," *Operations Research Letters*, vol. 31, no. 1, pp. 66–76, 2003.