Approximate Dynamic Programming for Storage Problems

Lauren A. Hannah LH140@DUKE.EDU
Duke University, Durham, NC 27708 USA

David B. Dunson DUNSON@STAT.DUKE.EDU
Duke University, Durham, NC 27708 USA

Abstract

Storage problems are an important subclass of stochastic control problems. This paper presents a new method, approximate dynamic programming for storage, to solve storage problems with continuous, convex decision sets. Unlike other solution procedures, ADPS allows math programming to be used to make decisions each time period, even in the presence of large state variables. We test ADPS on the day ahead wind commitment problem with storage.

1. Introduction

Storage problems are a special subclass of multistage stochastic optimization problems: a resource, such as energy, assets or stock options, is held so that it can be used either now for a certain gain or in the future with uncertain rewards. In each time period, the decision maker weighs the benefit of a here-and-now reward against the value of saving the resource for a later date. However, many storage problems, particularly those in energy, have action spaces that are much larger than those in traditional reinforcement learning problems. Math programming is an efficient way to search the action space, but it a convex, deterministic reward function. Rewards, however, are often dependent on a large "state of the world" variable. We exploit the special structure of storage—namely that time periods are tied only through the storage resource—to produce a method that accommodates convex optimization in the action space within a reinforcement learning setting. These methods are applied to the day ahead wind commitment problem, where a wind farm operator commits to production levels for the next 24 hours by placing a bid on the day ahead power market. The difficulties are that 1) the wind level is not known until is arrives and 2) the action is selecting production levels for the next 24 hours at once.

Sequential decision problems have been studied in both the reinforcement learning and operations research communities. Reinforcement learning typically solves multistage stochastic optimization problems by approximating a value to go function and choosing the action that maximizes the sum of current rewards and the estimated value to go. There are a variety of techniques to approximate value functions, including least squares regression with basis functions (Tsitsiklis & Van Roy, 2001; Lagoudakis & Parr, 2003), clustering and aggregation (Singh et al., 1995); these methods place no restrictions on the shape of the value function. However, in order to use math programming to search the action space, the value function needs to be convex or concave in the storage resource for every state of the world. This can be treated as shape-restricted functional regression, that is, regression where the response is a shape-restricted functional. Shape-restricted functional regression is largely unexplored, but there are many methods for univariate convex regression (Meyer, 2008; Seijo & Bodhisattva, 2011).

Likewise, the operations research community has approached these problems mainly through stochastic programming (SP). SP combines mathematical programming formulations with statistical approximations of the underlying outcome space, allowing a search over large decision spaces (Shapiro et al., 2009). SP estimates the value of future outcomes and decisions by simulating a few time stages into the future to produce a here-and-now decision, rather than an implementable policy. Most SP approaches rely on modeling the outcome space with a scenario tree: n_1 realizations from the outcome distribution of the first time period are sampled; for each of those, n_2 realiza-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

tions from the second time period are sampled from the conditional distribution and so on. While this sampling method gives desirable statistical properties, trees grow exponentially in the number of time periods, require a model for generation and often sparsely sample the outcome space.

We use approximate dynamic programming for storage (ADPS) to blend reinforcement learning and math programming. We simulate only T time periods in advance and nonparametrically approximate shape-restricted value functions. With the approximations, the problem is then solved via backward recursion, like traditional dynamic programming. Value functions are approximated by clustering states with Dirichlet process mixture models so that convex value functions can be fit within each cluster (Hannah et al., 2010). This scales well to large state spaces and avoids the difficulties that would be associated with shape-restricted functional regression. Scalability is demonstrated as ADPS is applied to the day ahead wind commitment problem, which has a complex state variable and requires the solution of a math program every time period to obtain commitment levels.

The rest of the paper is organized as follows. In Section 2, we review multistage stochastic optimization and apply it to the day ahead wind commitment problem. In Section 3, we present the ADPS algorithm. In Section 4, we apply ADPS to wind commitment in the day ahead electricity market with a storage resource. In Section 5, we give conclusions and future directions.

2. Multistage Stochastic Optimization and Storage

In this paper, we draw heavily from the stochastic programming representation of sequential decision problems, mainly to distinguish when information is available and when decisions are made. Notation is in the SP format. We begin by reviewing a single stage stochastic optimization problem,

$$\max_{x\in\mathcal{X}(\xi)} \mathbb{E}\left[f(x,\xi)\right].$$

Here $x \in \mathbb{R}^d$ is a continuous decision variable, f is a random utility function, $\xi : \Omega \to \Xi$ is a random variable, and $\mathcal{X}(\xi)$ is a random convex constraint set.

The single stage problem can be expanded to a $T-$stage stochastic optimization problem,

$$\max_{x_0\in\mathcal{X}_0} f_0(x_0) + \mathbb{E}\left[\max_{x_1\in\mathcal{X}_1(x_0,\xi_{[1]})} f_1(x_1,\xi_1) + \mathbb{E}\left[\cdots + \mathbb{E}\left[\max_{x_T\in\mathcal{X}_T(x_{[T-1]},\xi_{[T]})} f_T(x_T,\xi_T)\right]\right]\right], \tag{1}$$

where $x_{[t]} = (x_0, x_1, \ldots, x_t)$ is the set of all previous decisions, $\xi_1, \ldots, \xi_T$ is a random data process, $\xi_{[t]} = (\xi_1, \ldots, \xi_t)$ and $\mathcal{X}_t(x_{[t-1]}, \xi_{[t]})$ is a decision set that depends on the current state of the data process and previous outcomes. Note that when T is large, Equation (1) can be used to approximate an infinite horizon decision process.

The nested expectation format of Equation (1) is cumbersome. It can be described more compactly by a set of recursive, value-to-go functions. Let

$$V_t(x_{[t-1]}|\xi_{[t]}) = \max_{x_t\in\mathcal{X}_t(x_{[t-1]},\xi_{[t]})} f_t(x_t,\xi_t) + \mathbb{E}\left[V_{t+1}(x_{[t]}|\xi_{[t+1]})|\mathcal{F}_t\right] \tag{2}$$

for $t = 0, \ldots, T-1$ and $V_T(x_{[T-1]}|\xi_{[T]}) = 0$. Here $\mathcal{F}_t$ is the filtration generated by $(\xi_1, \ldots, \xi_t)$. We will use the value function representation throughout the rest of this paper.

2.1. Storage as Multistage Stochastic Optimization

We use structure to reduce the size of the value functions. Storage problems have the distinguishing characteristic that the value functions between times $t-1$ and t are tied together only by a storage variable, R_t, which denotes the amount of a resource in storage at time t. Therefore, the value functions need only capture the value of R_t conditioned on the random outcome. For storage problems, Equation (2) can be written as

$$\begin{aligned} V_t(R_t|\xi_{[t]}) = \max_{x_t} & f_t^K(x_t,\xi_t) + \mathbb{E}\left[f_t^U(x_t,\xi_{t+1}) \right. \\ & \left. + V_{t+1}(R_{t+1}|\xi_{[t+1]})|\mathcal{F}_t\right] \\ \text{s.t. } & x_t \in \mathcal{X}(R_t) \\ & R_{t+1} = f^R(R_t,x_t,\xi_{t+1}) \end{aligned} \tag{3}$$

for $t = 0, \ldots, T-1$ and $V_T(R_T|\xi_{[T]}) = 0$. Here, $\mathcal{X}(R_t)$ is a decision set that depends only upon the current storage resource and $f^R(R_t, x_t, \xi_{t+1})$ is a transition function between the current storage level and the storage level in the next time period. Note that we use R_{t+1} in the value function instead of $x_{[t]}$; it is a post-decision state variable and this change in time index will allow us to more easily approximate the expectation. The value function $V_{t+1}(x_{[t]} \,|\, \xi_{[t+1]})$ is broken up into two parts, the storage value function $V_{t+1}(R_{t+1}|\xi_{[t+1]})$, which is tied to the next time period, and a function $f_t^U(x_t, \xi_{[t+1]})$, that represents a here-and-now reward derived from the random information that arrives between time t and $t+1$, ξ_{t+1}.

2.2. Example: Day Ahead Wind Commitment

Throughout this paper we use the example of wind commitment on the day ahead market. Wind farms produce energy from a renewable but stochastic resource. The energy markets, however, are based around bids for future production, either 15 minutes or 1 hour before production in the regulating markets, or up to 24 hours before production in the day ahead market. Such markets work well for traditional energy producers, who make energy by burning a resource (coal and natural gas) or operate on a fixed, long-term schedule (nuclear), but can be problematic for renewable producers. For wind farms, determining the level of commitment (energy pledged) for a future production contract is a stochastic optimization problem. When storage is added, it becomes a multistage stochastic optimization problem (Ruiz et al., 2009). A slightly simplified model for the day ahead market is as follows. Parameters are given in Table 1.

1. Place a bid. At the end of day t, the wind farm manager sees electricity market prices $p_t = (p_{t,1}, \ldots, p_{t,24})$ for each of the next 24 hours in dollars per megawatt hour (MWh). Given p_t, she pledges how much electricity her farm will produce for each of the next 24 hours, $x_t = (x_{t,1}, \ldots, x_{t,24})$, in MWh.

2. Generate power. During day $t+1$, the farm receives wind $W_{t+1} = (W_{t+1,1}, \ldots, W_{t+1,24})$. This is converted into power output $L_{t+1} = (L_{t+1,1}, \ldots, L_{t+1,24})$. The wind-price pair (W_{t+1}, p_{t+1}) form ξ_{t+1}.

3. Distribute power to satisfy commitment. The power output L_{t+1} can be 1) sent to satisfy the commitment x_t, 2) placed into storage, or 3) dumped via discharging heat. These values are defined as the dummy decision variables, $y^x_{t+1,i}$, $y^{add}_{t+1,i}$ and $y^d_{t+1,i}$, respectively. The commitment can be comprised of electricity from 1) wind power L_{t+1}, 2) storage, or 3) unmet commitment (i.e. bought off the spot market); these are defined as $y^x_{t+1,i}$, $y^{rem}_{t+1,i}$ and $y^u_{t+1,i}$, respectively. The storage level when day $t+1$ starts is R_t; intermediate levels throughout the day are $z_{t+1} = (z_{t+1,0}, \ldots, z_{t+1,24})$; the final level $z_{t+1,24}$ determines the starting storage level for the next day. The maximum level of storage is R_{max}. The system has the following dynamics:

$$
\begin{aligned}
L_{t+1,i} &= y^x_{t+1,i} + y^{add}_{t+1,i} + y^d_{t+1,i},\\
x_{t,i} &= y^x_{t+1,i} + y^{rem}_{t+1,i} + y^u_{t+1,i},\\
z_{t+1,i} &= z_{t+1,i-1} + y^{add}_{t+1,i} - y^{rem}_{t+1,i},\\
0 &\le z_{t+1,i} \le R_{max}.
\end{aligned}
$$

Table 1. Parameters and variables for the day ahead wind commitment problem at time t.

		Decision Variables
x_t	=	COMMITMENT DECISION
		Storage Variable
R_t	=	STORAGE LEVEL AT TIME t
		Random Variables
W_{t+1}	=	WIND FROM TIME t TO $t+1$
L_{t+1}	=	POWER OUTPUT FROM TIME t TO $t+1$
		Power Flow Variables
y^x_{t+1}	=	POWER FROM WIND TO COMMITMENT
y^{add}_{t+1}	=	POWER FROM WIND TO STORAGE
y^{rem}_{t+1}	=	POWER FROM STORAGE TO COMMITMENT
y^d_{t+1}	=	POWER FROM WIND THAT IS DUMPED
y^u_{t+1}	=	UNMET COMMITMENT
z_{t+1}	=	INTRA-DAY STORAGE LEVELS
		Other Parameters
R_{max}	=	MAXIMUM STORAGE LEVEL
p_t	=	DAY AHEAD MARKET PRICES

4. Get revenue. Revenue is generated from the commitment level multiplied by the price, $\sum_{i=1}^{24} x_{t,i}p_{t,i}$. Revenue is lost by dumping power, $\sum_{i=1}^{24} f^d(y^d_{t+1,i})$, or by not meeting the commitment level, $\sum_{i=1}^{24} f^u(y^u_{t+1,i})$. For a horizon of T days, the entire problem is defined by the following recursive equations,

$$
\begin{aligned}
V_t(R_t|\xi_{[t]}) = \max_{x_t} &\sum_{i=1}^{24}(x_{t,i}p_{t,i} - f^d(y^d_{t+1,i}) - f^u(y^u_{t+1,i}))\\
&+ \mathbb{E}\left[V_{t+1}(R_{t+1}|\xi_{[t+1]})|\mathcal{F}_t\right]
\end{aligned}
$$

subject to

$$
\begin{aligned}
z_{t+1,0} &= R_t,\\
L_{t+1,i} &= y^x_{t+1,i} + y^{add}_{t+1,i} + y^d_{t+1,i}, \quad i = 1, \ldots, 24,\\
x_{t,i} &= y^x_{t+1,i} + y^{rem}_{t+1,i} + y^u_{t+1,i}, \quad i = 1, \ldots, 24,\\
z_{t+1,i} &= z_{t+1,i-1} + y^{add}_{t+1,i} - y^{rem}_{t+1,i}, \quad i = 1, \ldots, 24,\\
0 &\le z_{t+1,i}, \quad i = 1, \ldots, 24,\\
R_{max} &\ge z_{t+1,i}, \quad i = 1, \ldots, 24,\\
R_{t+1} &= z_{t+1,24},
\end{aligned}
$$

for $t = 0, \ldots, T-1$, where $V_T(R_T) = 0$.

3. Solution via ADPS

The storage problem in Equation (3) can be solved as a stochastic dynamic program (SDP). That is, since $V_T(R_T|\xi_{[T]})$ can be found, it is found for all values of R_T and all sample paths $(\xi_1, \ldots, \xi_T)$. The values

for time T are then plugged into the subproblems for time $T-1$ and the problem is solved recursively. This method is computationally infeasible in most cases.

Approximate dynamic programming models stochastic dynamic programs by approximating value functions and representing dynamics in a simplified form. Equation (3) is conditioned on $\xi_{[t]}$, the *entire history* of random variables. Instead of conditioning on everything, let S_t be a *state variable*. It is the smallest collection of information needed to make a decision. State variables reduce the domain of the value functions and makes the transition from S_t to S_{t+1} Markov.

The state variable gives information that is associated with both the current objective function and the information needed to take the expectation $\mathbb{E}[V_{t+1}(R_{t+1}|S_{t+1})|S_t]$. It is often well approximated by a smaller subset of information; in an abuse of notation, we use these values interchangeably. For example, in the wind commitment problem, we define S_t as the set of currently observed market prices, p_t, and the wind from the previous day, W_t. The prices are used directly in the optimization problem and provide information about future prices. The wind from the previous day provides information about future wind. The state could be expanded to include information like price and wind forecasts, but we use only the smaller variable due to the unavailability of data.

When $\mathcal{F}_t$ is replaced by S_t, the value functions in Equation (3) are still impossible to compute if S_t is a large discrete or continuous space. Therefore, the value function $V_t(R_t|S_t)$ is replaced with an approximation, $\bar{V}_t(R_t|S_t)$. The new problem faces two main difficulties: 1) computing the expectation $\mathbb{E}\left[\bar{V}_{t+1}(R_{t+1}|S_{t+1})|S_t\right]$, and 2) maximizing over x_t. The expectation is difficult because it is defined over a high-dimensional space; the maximization is difficult because the decision is included in the expectation and may require math programming.

We propose using approximate dynamic programming for storage (ADPS) to solve these problems. First, because we cannot solve an infinite number of recursive equations, we collect a finite number of samples from the state process, $(S_0(\omega_k),\dots,S_T(\omega_k))_{k=1}^n$, and random information process, $(\xi_1(\omega_k),\dots,\xi_T(\omega_k))_{k=1}^n$. These samples are used to approximate the infinite dimensional state and outcome spaces. The storage variable is approximated by a finite sampling. For each sample and storage pair, the value function is approximated by a weighted average of the outcomes given by the other samples. Dirichlet process mixture models over the state space provide an appealing way to weight the samples by clustering. Solving these problems for the storage level samples gives a concave approximate value function conditioned on the state $S_t(\omega_k)$; this is done for all k. The value functions are then passed on to time $t-1$. We assure that the maximization over x_t is easily computable by maintaining concavity in the approximate value functions and approximating the expectation with a deterministic weighted average.

3.1. Single Stage Stochastic Optimization

To solve Equation (3), we begin by approximating the infinite-dimensional state and outcome spaces by n sample paths, $(S_0(\omega_k),\dots,S_T(\omega_k))_{k=1}^n$ and $(\xi_1(\omega_k),\dots,\xi_T(\omega_k))_{k=1}^n$. Unlike scenario trees,these samples are not conditionally independent (but neither do they grow exponentially with the number of time periods). However, the distribution of (S_{t+1},ξ_{t+1}) is implicitly assumed to be Markov conditioned on S_t. Therefore, if we know $\bar{V}_{t+1}(R_{t+1}|S_{t+1})$, we can treat Equation (3) as a single stage stochastic optimization problem for a fixed state $S_t = s_t$. Hannah et al. (2010) develops methods to solve single stage stochastic optimization problems with large, continuous decision spaces and non-i.i.d. observations that include a state variable. We use their "function-based" optimization method, which takes an average of the outcomes weighted by the state variable,

$$\begin{aligned}\bar{V}_t(r_t|s_t) = \max_{x_t} f_t^K(x_t,\xi_t) + \sum_{k=1}^n w_n(s_t,S_t(\omega_k)) \quad (4)\\ \times[f_t^U(x_t,\xi_{t+1}(\omega_k)) + \bar{V}_{t+1}(R_{t+1}(\omega_k)|S_{t+1}(\omega_k))]\\ \text{s.t. } x_t \in \mathcal{X}(r_t)\\ R_{t+1}(\omega_k) = f^R(r_t,x_t,\xi_{t+1}(\omega_k)), \quad k=1,\dots,n.\end{aligned}$$

Here, $w_n(s_1,s_2)$ is a weight function between states s_1 and s_2 conditioned on the n samples; $w_n(s_1,s_2) \geq 0$ for every s_1, s_2 and $\sum_{k=1}^n w_n(s,S(\omega_k)) = 1$ for every s. Weights are discussed in Section 3.2.

Starting with $t=T$, we can approximate a value function for an arbitrary state-resource pair, (s_T,r_T). At $t=T-1$, Equation (4) requires value function approximations only for $(S_T(\omega_k))_{k=1}^n$. However, they are required to be continuous, concave *functions* in R_T. Convex functions can be produced by sampling from R_t and then fitting a convex regression; observations from other sample paths can be included in the regression by weighting them accordingly.

For simplicity, we use the following sampling and regression scheme. If R_t is a scalar variable, it can be discretized between 0 and R_{max} to produce an approximation of $V_t(R_t \mid s_t)$ for every fixed s_t. Assume R_t is scalar and let $\{0 = r_0, r_1, \dots, r_J = R_{\max}\}$ be such a

discretization. The value function for R_t is constructed by averaging over the two closest values of r,

$$\bar{V}_t(R_t \,|\, s_t) = \frac{R_t - r_i}{r_{i+1} - r_i} \bar{V}_t(r_i \,|\, s_t) + \frac{r_{i+1} - R_t}{r_{i+1} - r_i} \bar{V}_t(r_{i+1} \,|\, s_t), \tag{5}$$

for $r_i \le R_t < r_{i+1},\ i = 0, \ldots, J-1$. Then Equation (4) is solved at $(r_j, S_t(\omega_k))$ for each j and k.

3.2. Weight Functions via Dirichlet Process Mixture Models

The weight functions in Equation (4) try to weight the observations in a way that would make them act as if they were i.i.d. from the conditional distribution given $S_t = s_t$. We do this by assuming the pairs $(S_t, S_{t+1}, \xi_{t+1})$ come from a mixture model,

$$h(S_t, S_{t+1}, \xi_{t+1}) = \sum_{i=1}^{K} p_i \, g\left(S_t, S_{t+1}, \xi_{t+1} \,|\, \theta_i^*\right),$$

where h is the density, K is the number of components, p_i is the i^{th} component weight, and g is a parametric density function with parameter θ_i^*. Mixture models offer broad distributional support but still reduce the distribution into a finite dimensional model. Given samples $(S_t(\omega_k), S_{t+1}(\omega_k), \xi_{t+1}(\omega_k))_{k=1}^n$, let $(\theta_k)_{k=1}^n$ be the component parameters associated with each sample.

In practice, however, neither the parameters for each observation, $\theta_{1:n}$, nor the number of components, K, are known. To cope with these problems, a Dirichlet process mixture model places a prior over the number of observed components, their parameters, and the associations between observed data and components. A Dirichlet process (DP) with base measure $\mathbb{G}_0$ and concentration parameter α is used to place a distribution over the joint distribution of (p_i, θ_i^*), the mixture proportion and location of component i (Ferguson, 1973; Antoniak, 1974). For simplicity, assume generic i.i.d. data $Z_1, \ldots, Z_n$ with a distribution that is modeled by a mixture over the parametric distribution $G(\theta)$ (associated with density g),

$$P \sim DP(\alpha, \mathbb{G}_0), \quad \theta_i | P \sim P, \quad Z_i | \theta_i \sim G(\theta_i). \tag{6}$$

The distribution P is drawn from a Dirichlet process; it is an almost surely discrete measure over parameters, with the mixture proportion associated with θ as the atomic weight.

To construct weights from Equation (6), note that a Dirichlet process prior places a distribution over data partitions, which are induced by the number of components seen, K, and the parameters, $\theta_{1:n}$. Let $\mathbf{p} = \{C_1, \ldots, C_{K(\mathbf{p})}\}$ be the partition of the observations $\{1, \ldots, n\}$. Here $C_i = \{j : \theta_j = \theta_i^*\}$ is the partition set generated by $K(\mathbf{p})$ unique parameter values, denoted $\theta_1^*, \ldots, \theta_{K(\mathbf{p})}^*$. If the partition $\mathbf{p}$ is known, the query variable z is included into cluster C_i with probability

$$p_s(C_i|\mathbf{p}) = \mathbb{P}(z \in C_i \,|\, \mathbf{p}, Z_{1:n}) \propto n_i \int g(z \,|\, \theta^*) dH_{C_i}(\theta^*),$$

where n_i is the number of elements in C_i, and $H_{C_i}(\theta^*)$ is the posterior distribution of θ^* conditioned on $\mathbb{G}_0$ and the observations $\{Z_j : j \in C_i\}$. Given $\mathbf{p}$, the weight function is the probability that the hidden parameter for z would be θ_i, the parameter for Z_i,

$$w_n(z, Z_i) \,|\, \mathbf{p} = \sum_{j=1}^{K(\mathbf{p})} \frac{p_s(C_j \,|\, \mathbf{p})}{n_j} \mathbf{1}_{\{i \in C_j\}}. \tag{7}$$

Equation (7) is conditioned on a partition structure. Let $\pi(\mathbf{p})$ be the prior and $\pi(\mathbf{p}|Z_{1:n})$ the posterior distribution for partitions $\mathbf{p}$ induced by the Dirichlet process. Integrating with respect to the posterior, we obtain unconditional weights,

$$\begin{aligned} w_n(z, Z_i) &= \sum_{\mathbf{p}} \pi(\mathbf{p}|Z_{1:n}) \sum_{j=1}^{K(\mathbf{p})} \frac{p_s(C_j \,|\, \mathbf{p})}{n_j} \mathbf{1}_{\{i \in C_j\}} \qquad (8) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{K(\mathbf{p}^{(m)})} \frac{p_s(C_j \,|\, \mathbf{p}^{(m)})}{n_j} \mathbf{1}_{\{i \in C_j\}}. \end{aligned}$$

It is infeasible to integrate over all of the partitions; therefore, we approximate Equation (8) by performing a Monte Carlo integration with M posterior partition samples, $(\mathbf{p}^{(m)})_{m=1}^M$. We obtain $(\mathbf{p}^{(m)})_{m=1}^M$ by generating M samples of the hidden parameters, $\theta_{1:n}$, from the posterior of Equation (6) with Gibbs sampling (Neal, 2000).

The data $Z_{1:n}$ can either be the states $(S_t(\omega_i))_{i=1}^n$ or the pairs $(S_t(\omega_i), S_{t+1}(\omega_i), \xi_{t+1}(\omega_i))_{i=1}^n$. The former relies on assumed continuity of the expectation with respect to S_t. The latter requires that (S_{t+1}, ξ_{t+1}) be integrated out from the mixture to obtain $p_s(C_i|\mathbf{p})$; in general, this will give better weights.

3.3. ADPS

The storage problem is addressed by solving smaller, deterministic subproblems in succession. A full description of this method, approximate dynamic programming for storage, is given in Algorithm 1. For

Algorithm 1 ADPS: Approximate Dynamic Programming for Storage

Input: time horizon T, state variable observations $(S_0(\omega_k),\dots,S_T(\omega_k))_{k=1}^n$, random outcome observations $(\xi_1(\omega_k),\dots,\xi_T(\omega_k))_{k=1}^n$, storage discretization $\{0=r_0,r_1,\dots,r_J=R_{\max}\}$
Set $\bar{V}_T(R_T|S_T)=0$
for $t=T-1$ **to** 0 **do**
  for $k=1$ **to** n **do**
    Generate weights $(w_n(S_t(\omega_k),S_t(\omega_i)))_{i=1}^n$
    for $j=0$ **to** J **do**
      Set

$$\bar{V}_t(r_j|S_t(\omega_k)) = \max_{x_t} f_t^K(x_t,\xi_t) + \sum_{i=1}^{n} w_n(S_t(\omega_k),S_t(\omega_i)) \left[f_t^U(x_t,\xi_{t+1}(\omega_i)) + \bar{V}_{t+1}(R_{t+1}(\omega_i)|S_{t+1}(\omega_i))\right]$$

$$\text{subject to}: x_t \in \mathcal{X}(r_t), \qquad R_{t+1}(\omega_i) = f^R(r_j,x_t,\xi_{t+1}(\omega_i)), \quad i=1,\dots,n \tag{9}$$

    end for
    If $r_i \le R_t < r_{i+1},\ i=0,\dots,J-1$, set $\bar{V}_t(R_t \mid S_t(\omega_k)) = \frac{R_t-r_i}{r_{i+1}-r_i}\bar{V}_t(r_i \mid S_t(\omega_k)) + \frac{r_{i+1}-R_t}{r_{i+1}-r_i}\bar{V}_t(r_{i+1} \mid S_t(\omega_k))$
  end for
end for

this storage sampling scheme, ADPS requires $n \times J \times T$ optimization problems to be solved. However, given $\bar{V}_{t+1}(r|s)$, the $n \times J$ optimization problems that need to be solved for time t are all independent and thus their solutions can be parallelized. ADPS also maintains structural properties for all subproblems. As long as $f_t^K(x_t,\xi_t)$ and $f_t^U(x_t,\xi_{t+1})$ are concave in x_t and r_t for every ξ_t and ξ_{t+1}, then all optimization problems given by Equation (9) in Algorithm 1 are convex and hence quickly solvable. Nevertheless, convexity may need to be enforced due to approximation error.

4. Day Ahead Wind Commitment

We implemented ADPS on the day ahead wind commitment problem with storage. We obtained hourly wind speed data for the years 1998 through 2005 from the North American Land Data Assimilation Survey. Data were collected by satellite on a 1/8 degree grid over North America; we selected locations in the outer banks of North Carolina (33.9375N, 77.9375W); north of Cleveland, OH in Lake Erie (41.8125N, 81.5625W); and offshore from Point Judith, RI (41.3125N, 71.4375W). Day ahead market data were obtained for the PJM (New York and New Jersey area) market for the years 2002 through 2009. Power generation curves were used for the GE 1.5MW SL turbine. Average power generation was around 7.5 MWh per day. The loss function for unmet demand was two times the price for that hour; the loss for dumping excess energy was fixed at \$5 per MWh.

The state variable S_t was composed of the last 24 hours of wind velocity cubed, $(W_{t,1}^3,\dots,W_{t,24}^3)$, and the pricing data $(p_{t,1},\dots,p_{t,24})$. Both of these variables are high dimensional but can be viewed as functional data. We used functional principal component analysis (fPCA) (Ramsay & Silverman, 2005) to decompose the functions into their principal components. See Table 2 for the variance reduction associated with each principal component. We replaced the wind and price vectors with the scores for their first two principal components. In both cases, the first component corresponded to magnitude while the second to a price/wind velocity peak in the morning.

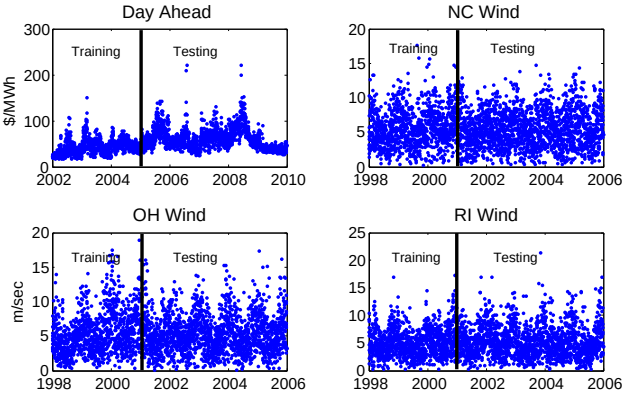

Figure 1. Day ahead market prices in \$ per MWh and wind levels in meters per second for hour 12 of each day. The first 3 years were used for training and the last 5 were used for testing.

We trained on the first three years of data (1096 - T days) and tested on the other five (1826 days); this corresponded to a training, testing breakdown of 1/1/02–12/31/04, 1/1/05–12/31/09 in prices and 1/1/98–12/31/00, 1/1/01–12/31/05 in wind. Time series data are shown in Figure 1. Training data are bro-

Table 2. Percent of variance explained by fPCA component for wind velocity cubed (by location) and day ahead market. The first two component scores were used as covariates for the mixture model.

Comp.	NC	OH	RI	Day Ahead
1	73.02%	75.92%	71.94%	78.87%
2	15.17%	15.25%	18.73%	15.71%
3	6.04%	5.26%	5.34%	2.18%
4	2.58%	1.72%	1.94%	1.19%
5	1.50%	0.86%	0.95%	1.10%
Total	98.32%	99.02%	98.85%	99.06%

ken into overlapping sets based on T; if $T = 2$, days $(1, 2)$, $(2, 3)$, etc. form the training datasets. We used a DP mixture of Gaussian distributions. Five policies were compared on four storage sizes: 0 MWh (no storage), 7.5 MWh, 15 MWh and 30 MWh.

Persistence. Electricity is pledged at the level it was generated the day before, i.e., $x_t = L_t$.

Average. We used the function-based method of (Hannah et al., 2010) to produce a weighted average of previous observations. This is equivalent to setting $T = 1$ in ADPS,

$$x_t^*(s_t, r_t) = \arg\max_{x_t} \sum_{k=1}^{n} w_n(s_t, S_t(\omega_k)) \times \sum_{i=1}^{24} x_{t,i} p_{t,i} - f^d(y^d_{t+1,i}(\omega_k)) - f^u(y^u_{t+1,i}(\omega_k))$$
$$\text{s.t. } R_{t+1}(\omega_k) = f^R(r_t, x_t, L_{t+1}(\omega_k)).$$

ADPS, $T = 2$. We created a value function, $\bar{V}_1(r|s)$ by running Algorithm 1 with $T = 2$ and saving $\bar{V}_1$. Then, we plugged the resulting value function $\bar{V}_1(r|s)$ into the following equation to generate a policy,

$$x_t^*(s_t, r_t) = \arg\max_{x_t} \sum_{k=1}^{n} w_n(s_t, S_t(\omega_k)) \times \sum_{i=1}^{24} x_{t,i} p_{t,i} - f^d(y^d_{t+1,i}(\omega_k)) - f^u(y^u_{t+1,i}(\omega_k)) + \bar{V}_1(R_{t+1}(\omega_k) \,|\, S_{t+1}(\omega_k))$$
$$\text{s.t. } R_{t+1}(\omega_k) = f^R(r_t, x_t, L_{t+1}(\omega_k)). \quad (10)$$

ADPS, $T = 4$. Same method as above, with $T = 4$.

ADPS, $T = 6$. Same method as above, with $T = 6$.

All policies were run in Matlab on a 2.66 GHz Intel Core i7 with 4 GB of RAM. Optimization problems were solved with the function *fmincon* using an interior points algorithm. Mean solution time for optimization problems with the form of Equation (10) was 0.15 seconds. Storage levels were discretized by 0.5, 1 and 2 MWh for 7.5, 15 and 30 MWh, respectively. Values for each policy were generated by using the first time period policy as a policy for the MDP generated by the testing data, which is not broken into T stage problems. Results are given by location and storage size in Table 3.

The results show that ADPS consistently adds value, which generally grows with decreasing returns. What is less expected, however, is that the time T in ADPS seems to make little difference. This appears to be due to the relatively small storage size (about 1 to 4 days of electricity). There are daily and seasonal cycles in electricity prices; the storage size is large enough to take advantage of the former but not the latter.

The day ahead wind commitment problem was chosen to demonstrate the ability of ADPS to solve problems that require math programming to generate a decision each stage, while that decision is still dependent on a large state variable. In practice, however, renewable penetration is low enough and tax subsidies are high enough that wind power is not (yet) in the day ahead market. Most production is owned by power companies who can use scheduling and short term forecasting to predict and make up for shortfalls (Smith et al., 2007). Scheduling is done using math programming; any incorporation of "future value" will require shape-restricted value functions that are dependent on a large state variable. ADPS provides a reasonable method to produce these. It should be noted that distributional forecasts can easily be included by training the value functions on a set of historical (or model-based) training data and then selecting m samples from the forecast to be used in place of $(\omega_i)_{i=1}^n$ in Equation (10).

5. Conclusions and Future Directions

In this paper, we presented a new method to solve stochastic storage problems that allows the use of math programming to search the action space even when the values of the actions depend on a large state variable. This is achieved by clustering states with a Dirichlet process mixture model and then fitting a shape-restricted value function within each cluster. Work is still needed for value function approximation: efficient sampling design for the storage resource, fast multivariate convex regression and Bayesian convex regression methods to include the value function in the state

Table 3. Average annual value of policy in $1,000 for given storage sizes and a 1.5 MW turbine. Percentages are percent change in value over the Average policy.

Site	Storage Size	Persistence	Average	ADPS, $T = 2$	ADPS, $T = 4$	ADPS, $T = 6$
NC	0 MWh	-0.42 (-102.7%)	**15.72**	–	–	–
	7.5 MWh	76.49 (-25.3%)	102.33	114.77 (12.2%)	**114.86 (12.2%)**	114.54 (11.9%)
	15 MWh	116.15 (-18.1%)	141.78	163.27 (15.2%)	**163.51 (15.3%)**	163.46 (15.3%)
	30 MWh	127.94 (-15.7%)	175.40	205.26 (17.0%)	**205.38 (17.0%)**	205.23 (17.0%)
OH	0 MWh	-13.91 (-461.2%)	**3.85**	–	–	–
	7.5 MWh	60.02 (-24.7%)	79.62	90.45 (13.6%)	**90.53 (13.7%)**	90.47 (13.6%)
	15 MWh	95.12 (-16.9%)	114.41	131.77 (15.2%)	**131.83 (15.2%)**	131.48 (14.9%)
	30 MWh	126.57 (-17.2%)	152.79	**171.82 (12.5%)**	170.81 (11.8%)	170.35 (11.5%)
RI	0 MWh	-10.80 (-202.7%)	**10.52**	–	–	–
	7.5 MWh	72.04 (-26.6%)	98.19	**107.60 (9.6%)**	107.58 (9.6%)	107.52 (9.5%)
	15 MWh	112.52 (-19.2%)	139.29	**155.00 (11.3%)**	154.46 (10.9%)	154.29 (10.8%)
	30 MWh	148.74 (-16.77%)	178.71	**200.83 (12.4%)**	198.72 (11.2%)	198.31 (11.0%)

mixture. For use in larger problems, policies need to be more robust: changes in training data can produce large changes in policy. This will require a combination of regularization, quantifying model uncertainty and using robust optimization.

Acknowledgments

We would like to thank the anonymous reviewers and editors who substantially improved this paper. Lauren A. Hannah is partially funded by the Duke University Provost's Postdoctoral Scholarship.

References

Antoniak, C. E. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

Ferguson, T. S. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Hannah, L. A., Powell, W. B., and Blei, D. M. Nonparametric Density Estimation for Stochastic Optimization with an Observable State Variable. In *Advances in Neural Information Processing Systems*, 2010.

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

Meyer, M. C. Inference using shape-restricted regression splines. *Annals of Applied Statistics*, 2(3):1013–1033, 2008.

Neal, R. M. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

Ramsay, J. O. and Silverman, B. W. *Applied functional data analysis: methods and case studies*. Springer Verlag, second edition, 2005.

Ruiz, P.A., Philbrick, C.R., and Sauer, P.W. Wind power day-ahead uncertainty management through stochastic unit commitment policies. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pp. 1–9. IEEE, 2009.

Seijo, E. and Bodhisattva, S. Nonparametric least squares estimation of a multivariate convex regression function. *The Annals of Statistics*, pp. to appear, 2011.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. *Lectures on stochastic programming: modeling and theory*. Society for Industrial Mathematics, 2009.

Singh, S., Jaakkola, T., and Jordan, M.I. Reinforcement learning with soft state aggregation. *Advances in Neural Information Processing Systems*, pp. 361–368, 1995.

Smith, J.C., Milligan, M.R., DeMeo, E.A., and Parsons, B. Utility wind integration and operating impact state of the art. *Power Systems, IEEE Transactions on*, 22(3):900–908, 2007. ISSN 0885-8950.

Tsitsiklis, J. N. and Van Roy, B. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.