INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Optimal Hour-Ahead Bidding in the Real-Time Electricity Market with Battery Storage Using Approximate Dynamic Programming

Daniel R. Jiang, Warren B. Powell

To cite this article:
Daniel R. Jiang, Warren B. Powell (2015) Optimal Hour-Ahead Bidding in the Real-Time Electricity Market with Battery Storage Using Approximate Dynamic Programming. INFORMS Journal on Computing 27(3):525-543. http://dx.doi.org/10.1287/ijoc.2015.0640

INFORMS Journal on Computing
Vol. 27, No. 3, Summer 2015, pp. 525–543
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2015.0640

Optimal Hour-Ahead Bidding in the Real-Time Electricity Market with Battery Storage Using Approximate Dynamic Programming

Daniel R. Jiang, Warren B. Powell
Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08540
{drjiang@princeton.edu, powell@princeton.edu}

There is growing interest in the use of grid-level storage to smooth variations in supply that are likely to arise with an increased use of wind and solar energy. Energy arbitrage, the process of buying, storing, and selling electricity to exploit variations in electricity spot prices, is becoming an important way of paying for expensive investments into grid-level storage. Independent system operators such as the New York Independent System Operator (NYISO) require that battery storage operators place bids into an hour-ahead market (although settlements may occur in increments as small as five minutes, which is considered near "real-time"). The operator has to place these bids without knowing the energy level in the battery at the beginning of the hour and simultaneously accounting for the value of leftover energy at the end of the hour. The problem is formulated as a dynamic program. We describe and employ a convergent approximate dynamic programming (ADP) algorithm that exploits monotonicity of the value function to find a revenue-generating bidding policy; using optimal benchmarks, we empirically show the computational benefits of the algorithm. Furthermore, we propose a distribution-free variant of the ADP algorithm that does not require any knowledge of the distribution of the price process (and makes no assumptions regarding a specific real-time price model). We demonstrate that a policy trained on historical real-time price data from the NYISO using this distribution-free approach is indeed effective.

Keywords: approximate dynamic programming; bidding; hour-ahead; energy storage; real-time market
History: Accepted by Alice Smith, Area Editor for Heuristic Search and Learning; received March 2014; revised November 2014; accepted November 2014.

1. Introduction

Bidding into the electricity market can be a complicated process, mainly because of the requirement of balancing supply and demand at each point in the grid. To solve this issue, the independent system operators (ISOs) and the regional transmission organizations (RTOs) generally use multi-settlement markets: several tiers of markets covering planning horizons that range from *day-ahead* to *real time*. The idea is that the markets further away from the operating time settle the majority of the generation needed to handle the predicted load, whereas the markets closer to the operating time correct for the small yet unpredictable deviations that may be caused by issues like weather, transmission problems, and generation outages (see, for example, Shahidehpour et al. 2002, Eydeland and Wolyniec 2003, Harris 2011 for more details). Settlements in these real-time markets are based on a set of intrahour prices, typically computed at 5-, 10-, or 15-minute intervals, depending on the specific market in question. A settlement refers to the financial transaction after a generator *clears the market*, which refers to being selected to either buy or sell energy from the market. If a generator does not clear the market, it remains idle and no settlement occurs. We refer to this situation as being *out of the market*.

Many ISOs and RTOs, such as the Pennsylvania–New Jersey–Maryland Interconnection (PJM), deal with the balancing market primarily through the day-ahead market. PJM's balancing market clears every five minutes (considered to be near "real time"), but the bids are all placed the previous day. See Eydeland and Wolyniec (2003) and the PJM Energy and Ancillary Services Market Operations Manual (http://www.pjm.com/~/media/documents/manuals/m11.ashx) for more information. In certain markets, however, it is not only possible to settle in real time, but market participants can also submit bids each hour, for an hour in the future. Thus, a bid (consisting of buy and sell prices) can be made at 1 P.M. that will govern the battery between 2 and 3 P.M. The process of both bidding and settling in real-time is a characteristic of the New York Independent System Operator (NYISO) real-time market and is the motivating example for this paper. Other

prominent examples of markets that include a real-time bidding aspect include California ISO and Mid-continent ISO. In particular, our goal is to pair battery storage with hour-ahead bidding in the real-time market for revenue maximization, a strategy sometimes referred to as *energy arbitrage*.

It is unlikely that profits from energy arbitrage alone can be sustainable for a company; however, if performed optimally, it can be an important part of a range of profit generating activities (one such example is the frequency regulation market). Walawalkar et al. (2007) provides an economic analysis of using a storage device for both energy arbitrage (using a simple "charge-off-peak and discharge-on-peak" policy) and frequency regulation in the New York area. The conclusion is that in New York City (but not the surrounding areas), there is a "high probability of positive NPV [net present value] for both energy arbitrage and regulation" (p. 2566), but even so, there is still significant risk of not being able to recover the initial capital cost. However, the potential for more efficient and cost-effective technology combined with better control policies can make energy arbitrage feasible in the near future. Other studies on the topic of the value of storage include Sioshansi et al. (2009, 2011) and Byrne and Silva-Monroy (2012).

In our problem, we assume that the goal is to optimally control a 1 MW battery; in practice, a company may operate a fleet of such batteries. Market rules state that we must bid in integer increments, meaning the possible actions at each settlement are to charge, discharge (both at a rate of 1 MW), or do nothing. Hence, our precise problem is to optimize the placement of two hour-ahead bids, a "positive" bid (for a quantity of $+1$ MW) and a "negative" bid (for a quantity of -1 MW) that correspond to selling (generation) and buying (negative generation), respectively, over a period of time such that purchased energy can be stored in the finite capacity battery. The goal is to maximize expected revenue. Further, given that our model is tailored to battery storage (inherently small capacity), it is reasonable to assume no price impact (i.e., our bids do not affect the spot prices of electricity). In the real-time market, bidding for the operating hour closes an hour in advance and the hour-ahead bid is fixed for the entire operating hour.

This paper makes the following contributions. We describe, in detail, a mathematical model of the bidding process in the real-time electricity market and formulate the sequential decision problem as a Markov decision process (MDP). Along the way, we show the structural properties of the problem (monotonicity of the contribution and value functions) that we utilize in our solution technique. Next, we describe and benchmark a convergent approximate dynamic programming (ADP) algorithm called *Monotone-ADP* (M-ADP; Jiang and Powell 2015) that can be used to obtain an approximate but near-optimal bidding policy. We also present a new version of Monotone-ADP utilizing post-decision states that allows us to train bidding policies without any model or knowledge of the distribution of real-time prices (which we call a distribution-free method), allowing our solution technique to be easily adopted in practice. Finally, we present a case study detailing the results of an ADP policy trained using only historical real-time price data from the NYISO. In the case study, we also compare the ADP policy to other rule-based policies, two of which are from the energy arbitrage literature and one from our industry contacts. All proofs can be found in the online supplement (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2015.0640).

2. Literature Review

With renewable energy sources like wind and solar becoming more established, the problem of energy storage is also becoming increasingly important. In this section, we first review studies dedicated solely to storage and then move on to those that consider the bidding aspect. Lastly, we discuss algorithmic techniques similar to our proposed method of Monotone-ADP.

Coupling wind energy with storage has been well studied in a variety of ways. The paper by Kim and Powell (2011) poses a wind energy commitment problem given storage and then analytically determines the optimal policy for the infinite horizon case. Sioshansi (2011) uses ideas from economics and game theory (i.e., the Stackelberg Model) to make several conclusions, including the finding that the value of storage increases with market competitiveness. In addition, Greenblatt et al. (2007) find that for high greenhouse gas emissions prices, compressed air energy storage is a better choice as a supplemental generator to wind energy when compared to natural gas turbines. The well-known smoothing effects of energy storage on intermittent renewable sources is studied in the context of wind power output by Paatero and Lund (2005).

Another problem within this realm is the storage of natural gas, which involves optimally controlling injection and withdrawal of gas from a storage facility that is typically underground. Carmona and Ludkovski (2010) use a technique known as *optimal switching* to solve a natural gas storage problem; computationally, the value function is approximated basis functions. In a similar vein, Thompson et al. (2009) formulate a stochastic control problem and numerically solve the resulting integro-differential equation to arrive at the optimal policy. Lai et al. (2010) propose using an ADP algorithm along with an approximation technique to reduce the number of state space dimensions for natural gas storage valuation.

Other energy storage problems include reservoir management (see Nandalal and Bogardi 2007) and pairing solar with battery storage (see Barnhart et al. 2013). It quickly becomes clear that all of these problems are similar; in fact, Secomandi (2010) gives the structure of the optimal policy for trading generic commodities given storage. At its core, energy storage has similarities to an array of classical problems related to operations research, such as resource allocation and inventory control.

There are also many studies that consider the bidding aspect of the electricity markets. One significant difference between many of these studies and our paper is that, rather than considering the placement of many bids at once, we consider a sequential, hourly bidding problem. Löhndorf and Minner (2010) consider a day-ahead bidding problem different from ours using an infinite horizon MDP; Conejo et al. (2002) solve a price-taker bidding problem using a deterministic look-ahead policy; Gross and Finlay (2000) formulate a constrained optimization problem for optimal bidding in a competitive power pool; and David (1993) develops both deterministic and stochastic models for bidding under the consideration of other market players. Finally, Löhndorf et al. (2013) uses approximate dual dynamic programming to solve a day-ahead bidding problem involving hydro storage. Along with the major algorithmic differences from our paper, Löhndorf et al. (2013) also works in a day-ahead setting with individual bids for each hourly subinterval, whereas we work in an hourly setting with bids that must be simultaneously active for every five-minute subinterval. Furthermore, to have complete information to make the optimal decision and to implement the transition dynamics, the previous bid (placed in the last time interval) is a part of our state variable, which is not the case for Löhndorf et al. (2013). For more details, the literature survey by Wen and David (2000) provides an excellent overview to strategic bidding.

In the case of real-world problems with large state spaces, backward dynamic programming is typically not a viable solution strategy, so we often use ADP techniques. In this paper, we consider a variant of the approximate value iteration (AVI) algorithm (see both Bertsekas and Tsitsiklis 1996 and Powell 2011) that exploits the monotonicity in certain dimensions of the optimal value function (also known as the *cost-to-go* function) to quickly approximate the shape of the value function. The algorithm, called *Monotone-ADP*, is analyzed in Jiang and Powell (2015) and was used previously as a heuristic in Papadaki and Powell (2003).

Like monotonicity, convexity/concavity also often arises in applications, and similar algorithms to Monotone-ADP that exploit these structural properties have been studied in Godfrey and Powell (2001), Topaloglu and Powell (2003), Powell et al. (2004), and Nascimento and Powell (2009). In general, these studies on monotonicity and convexity have shown that it is advantageous to use the structural properties of value functions in ADP algorithms.

3. Mathematical Formulation

We can formulate the problem mathematically as follows. Let M be the number of settlements made per hour and let $\Delta t = 1/M$ be the time increment between settlements (in hours). For example, in the NYISO, settlements occur every five minutes, so we choose $M = 12$. Although settlements are made intrahour, bidding decisions are always made on the hour, for an hour in the future. Thus, the operator places bids at 1 P.M. to operate the battery between 2 and 3 P.M., with settlements made in five-minute intervals within the hour. For time indexing, we use t (measured in hours); bidding decisions are made when $t \in \mathbb{N}$ and settlements occur when $t \in \mathcal{T} = \{k \cdot \Delta t\colon k \in \mathbb{N}\}$.

Let the price P_t for $t \in \mathcal{T}$ be a discrete-time, nonnegative, stochastic process. Because bidding decisions and settlements occur on two different schedules (every hour versus every Δt), we use the following notation. For $t \in \mathbb{N}$, let $P_{(t,\,t+1]}$ be an M-dimensional vector that represents the spot prices that occurred within the hour from t to $t+1$:

$$P_{(t,\,t+1]} = \left(P_{t+\Delta t}, P_{t+2\cdot\Delta t}, \ldots, P_{t+(M-1)\cdot\Delta t}, P_{t+1}\right). \quad (1)$$

Hence, $P_{(t,\,t+1]}$ does not become fully known until time $t+1$. Next, let our set of bidding decisions be a finite set $\mathcal{B}$ such that

$$\mathcal{B} \subseteq \left\{(b^-, b^+)\colon 0 \le b^- \le b^+ \le b_{\max}\right\}, \quad (2)$$

with $b_{\max} \in \mathbb{R}_+$. Let $b_t = (b_t^-, b_t^+) \in \mathcal{B}$ be the bidding decision made at t used for the interval $(t+1, t+2]$. All sell bids b_t^+ (or "positive" bids because we are transferring at a rate of $+1$ MW) less than the spot price are picked up for dispatch (releasing energy into the grid). All buy bids b_t^- (or "negative" bids because we are transferring at a rate of -1 MW) greater than the spot price are picked up for charge. If the spot price falls in between the two bids, we are out of the market and the battery stays in an idle state. When we are obligated to sell to the market but are unable to deliver, we are penalized $K \cdot P_t$, where $K \ge 0$. The restriction of $b_t^- \le b_t^+$ guarantees that we are never obligated to buy and sell simultaneously.

We remark that in the actual bidding process, the buy bid is a negative number and the criteria for clearing the market is that the bid is less than the negative of the spot price. Because our bids are for only two

quantities (±1 MW), the above reformulation of the bidding process is cleaner and more intuitive.

Let $R_t \in \mathcal{R} = \{0, 1, 2, \ldots, R_{\max}\}$ be the energy stored in the battery. For simplicity, assume that $R_{\max}$ is adjusted so that a unit of resource represents $1/M$ MWh of energy. Thus, it is clear that at each settlement within the hour, the change in resource is either $+1$, -1, or 0. We also define a deterministic function that maps a vector of intrahour prices $P \in \mathbb{R}_+^M$ and a bid $b = (b^-, b^+) \in \mathcal{B}$ to a vector of outcomes (charge $= -1$, discharge $= +1$, or idle $= 0$). Define q: $\mathbb{R}^M \times \mathcal{B} \to \{-1, 0, 1\}^M$ such that the mth component of $q(P, b)$ is

$$q_m(P, b) = \mathbf{1}_{\{b^+ < e_m^\intercal P\}} - \mathbf{1}_{\{b^- > e_m^\intercal P\}}, \tag{3}$$

where e_m is a vector of zeros with a one at the mth row (and thus picks out the mth component of the price vector P). Note that q is not dependent on time, but in the context of our hour-ahead bidding problem, we use it in the form of $q(P_{(t-1,\,t]}, b_{t-2})$, which is deterministic at time t. Figure 1 illustrates the intrahour behavior.

To define the hourly transition function between R_t and R_{t+1}, we model each of the individual settlements within the hour and then combine them recursively (because from t to $t+1$, we settle M times). Let $q_s \in \{-1, 0, 1\}^M$ be a vector of settlement outcomes and suppose $g_m^R(R_t, q_s)$ represents the amount of resource after the mth settlement. Thus, we have

$$\begin{aligned} g_0^R(R_t, q_s) &= R_t, \\ g_{m+1}^R(R_t, q_s) &= \left[\min\{g_m^R(R_t, q_s) - e_m^\intercal q_s, R_{\max}\}\right]^+ \end{aligned} \tag{4}$$

for $1 \le m \le M$. The intrahour resource levels are

$$R_{t+m\Delta t} = g_m^R(R_t, q_s).$$

Finally, let g^R be the hourly transition function, which is defined as a composition of the functions g_M^R and q in the following way:

$$R_{t+1} = g^R(R_t, P_{(t,\,t+1]}, b_{t-1}) = g_M^R\big(R_t, q(P_{(t,\,t+1]}, b_{t-1})\big). \tag{5}$$

The need for an hourly transition function from R_t directly to R_{t+1} (rather than simply defining the subtransitions between the intrahour settlements) is due to the hourly decision epoch of the problem.

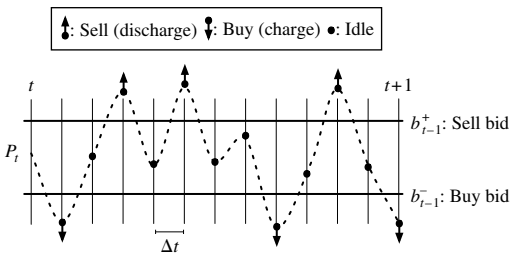


Figure 1 **Illustration of the Intrahour Bidding Behavior**

Proposition 1. *For an initial resource level $r \in \mathcal{R}$, a vector of intrahour prices $P \in \mathbb{R}^M$, a bid $b = (b^-, b^+) \in \mathcal{B}$, and a subinterval m, the resource transition function $g_m^R(r, q(P, b))$ is nondecreasing in r, b^-, and b^+.*

We now consider another dimension to our problem by allowing a limit to be imposed on the number of charge–discharge cycles used by the battery, for the sake of increasing the lifetime of the battery. Battery cycle life (the approximate number of cycles before capacity diminishes to around 80%), a key issue when considering economic feasibility, varies between the different types of battery technologies and the operating conditions, but are typically in the range of 1,000 (e.g., lead-acid) to 5,000 (e.g., vanadium redox); for an extensive review, see Yang et al. (2011). In our correspondence with industry colleagues, we found that a common (though possibly somewhat conservative) estimate of battery usage is 300 cycles/year, meaning that most devices can last at least three to four years. However, the model developed in this paper is for hourly decision making, and it would be impractical to solve the model for time horizons of several years. Note that different battery technologies degrade in different ways, but in general, degradation occurs slowly (nearly linearly with charge–discharge cycles) at first; after a point, efficiency drops much more rapidly.

Over a short horizon (on the order of days), the effects of battery degradation is negligible, but we propose the following way for one to impose a sort of artificial limit to the number of trades (charge–discharge cycles) performed. Let $L_t \in \mathcal{L} = \{0, 1, 2, \ldots, L_{\max}\}$ be decremented on every discharge of the battery (starting with $L_0 = L_{\max}$) and suppose that when selling to the market at a settlement time t' in $(t, t+1]$, the revenue is discounted by a factor of $\beta(L_{t'})$ where β: $\mathcal{L} \to [0, 1]$ is a nondecreasing function. Depending on the battery technology, preferences of the operator, and the time horizon of the model, the choice of β may vary greatly; the following list offers a few examples:

1. Constant: $\beta(l) = c \in [0, 1]$ for all $l \in \mathcal{L}$,
2. Step: $\beta(0) = 0$ and $\beta(l) = 1$ for $l \in \mathcal{L} \setminus \{0\}$,
3. Linear: $\beta(l) = l/L_{\max}$ for all $l \in \mathcal{L}$,
4. Power: $\beta(l) = (l/L_{\max})^{1/n}$ for some $n > 1$ and all $l \in \mathcal{L}$,

where (4) seeks to very roughly mimic the efficiency degradation of a real battery. We assume that the physical characteristics of the battery are summarized through β and the dynamics of L_t, which we now describe.

Similar to the specification of q in (3), we define a function d: $\mathbb{R}^M \times \mathcal{B} \to \{0, 1\}^M$ such that the mth component of $d(P, b)$ is

$$d_m(P, b) = \mathbf{1}_{\{b^+ < e_m^\intercal P\}},$$

which indicates the settlements for which a discharge occurred. Like before, we define the transition function from L_t to L_{t+1} using a sequence of subtransitions. Let $d_s \in \{0,1\}^M$ be a vector of settlement outcomes (in this case, whether a discharge happened) and

$$\begin{aligned} g^L_0(L_t, d_s) &= L_t, \\ g^L_{m+1}(L_t, d_s) &= [g^L_m(L_t, d_s) - e_m^\intercal d_s]^+ \end{aligned} \tag{6}$$

for $1 \le m \le M$. The intrahour values are

$$L_{t+m\Delta t} = g^L_m(L_t, d_s),$$

and the hourly transition function g^L is defined as

$$L_{t+1} = g^L(L_t, P_{(t,t+1]}, b_{t-1}) = g^L_M\big(L_t, d(P_{(t,t+1]}, b_{t-1})\big). \tag{7}$$

Proposition 2. *For an initial $l \in \mathscr{L}$, a vector of intrahour prices $P \in \mathbb{R}^M$, a bid $b = (b^-, b^+) \in \mathscr{B}$, and a subinterval m, the transition function $g^L_m(l, d(P, b))$ is nondecreasing in l, b^-, and b^+.*

At time t, we can determine the revenue from the previous hour $(t-1, t]$, which depends on the initial resource R_{t-1}, the remaining lifetime L_{t-1}, the intrahour prices $P_{(t-1,t]}$, and the bid placed in the previous hour, b_{t-2}. The revenue made at the mth settlement depends on four terms, the price $P_{t+m\Delta t}$, the settlement outcome $q_m(P, b)$ (which establishes the direction of energy flow), a *discount factor* γ_m (due to L_t), and the *undersupply penalty* U_m. Let $r \in \mathscr{R}$, $l \in \mathscr{L}$, $P \in \mathbb{R}^M$, and $b \in \mathscr{B}$. Because we discount only when selling to the market, let

$$\gamma_m(l, P, b) = \beta(l) \cdot \mathbf{1}_{\{q_m(P,b)=1\}} + \mathbf{1}_{\{q_m(P,b)\neq 1\}}. \tag{8}$$

The undersupply penalty takes values of either 1 (no penalty) or $-K$ (penalty):

$$U_m(r, P, b) = \big(1 - (K+1) \cdot \mathbf{1}_{\{r=0\}} \cdot \mathbf{1}_{\{q_m(P,b)=1\}}\big). \tag{9}$$

This penalization scheme reflects reality: the NYISO penalizes using a price-proportional penalty of $K = 1$ (in addition to lost revenue) to uphold the market balance. When a market participant reneges on a promise to deliver energy to the market, it must pay the penalty of the quantity times the market price to correct the imbalance; this is equivalent to *purchasing the energy from another generator* at the market price and delivering to the market.

Hence, we can write the following sum (over the settlements) to arrive at the hourly revenue, denoted by the function C:

$$\begin{aligned} &C(R_{t-1}, L_{t-1}, P_{(t-1,t]}, b_{t-2}) \\ &= \sum_{m=1}^{M} \gamma_m(L_{t-1+m\Delta t}, P_{(t-1,t]}, b_{t-2}) \cdot P_{t+m\Delta t} \cdot q_m(P_{(t-1,t]}, b_{t-2}) \\ &\quad \cdot U_m(R_{t-1+m\Delta t}, P_{(t-1,t]}, b_{t-2}). \end{aligned} \tag{10}$$

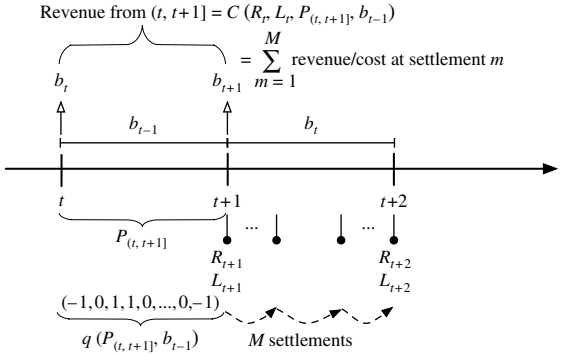


Figure 2 Illustration of the Bidding Process

Note that C is not time dependent. The timeline of events and notation we use is summarized in Figure 2. The top half of Figure 2 shows the contrast between when bids are placed and when bids are active: b_t and b_{t+1} are placed at times t and $t+1$ (arrows pointing up), whereas b_{t-1} is active for the interval $(t, t+1]$ and b_t is active for the interval $(t+1, t+2]$. It also shows that the revenue function $C(R_t, L_t, P_{(t,t+1]}, b_{t-1})$ refers to the interval $(t, t+1]$. The bottom half of Figure 2 shows an example of the bidding outcomes, i.e., the output of $q(P_{(t,t+1]}, b_{t-1})$. Finally, we emphasize that M settlements (and thus, transitions) occur between consecutive values of R_t and L_t because of the discrepancy between the bidding timeline (hourly) and the settlement timeline (every five minutes).

3.1. Markov Decision Process

The problem of optimizing revenue over a time horizon is a sequential decision problem that we can formulate as an MDP. First, suppose the set of state variables associated with the price process P_t is denoted $P^S_t \in \mathscr{P}$, where $\mathscr{P}$ is the space of price model state variables. The MDP can be characterized by the following components:

- The *state variable* for the overall problem is $S_t = (R_t, L_t, b^-_{t-1}, b^+_{t-1}, P^S_t) \in \mathscr{S}$, where $\mathscr{S}$ is the state space. The previous bid b_{t-1} is included because it is the bid that becomes valid at time t for the interval $(t, t+1]$ and is necessary for computing the resource transition function.
- The *decision* is the hour-ahead bid $b_t = (b^-_t, b^+_t) \in \mathscr{B}$ that is active for the interval $(t+1, t+2]$.
- The *exogenous information* in this problem is the price process P_t.
- The *state transition function* or *system model* S^M is given by

$$\begin{aligned} S_{t+1} &= S^M(S_t, b_t, P_{(t,t+1]}) \\ &= \big(g^R(R_t, P_{(t,t+1]}, b_{t-1}), \\ &\qquad g^L(L_t, P_{(t,t+1]}, b_{t-1}), b_t, P^S_{t+1}\big). \end{aligned} \tag{11}$$

- The *contribution function* in this model represents the expected value of the revenue in the interval from $t+1$ to $t+2$ using bid b_t given the current state S_t. Define

$$C_{t,\,t+2}(S_t, b_t) = \mathbf{E}\big(C(R_{t+1}, L_{t+1}, P_{(t+1,\,t+2]}, b_t) \mid S_t\big). \quad (12)$$

The double subscript of t and $t+2$ signifies that the contribution is determined at t (hence, variables indexed by $t' \le t$ are known) but represents the expectation of the revenue in the interval $(t+1, t+2]$. In practice, it is likely that we must redefine $C_{t,\,t+2}(S_t, b_t)$ as a sample expectation over the available training data (see §6) if (1) a stochastic model of the prices is unavailable or (2) the expectation is impossible to compute. Regardless of its form, we assume that $C_{t,\,t+2}(S_t, b_t)$ can be computed exactly at time t.

- Let $T-1$ be the last time for which a bid needs to be placed (hence, the trading horizon lasts until $T+1$ and the last value function we need to define is at T) and let $B_t^\pi\colon \mathcal{S} \to \mathcal{B}$ be the decision function for a policy π from the class Π of all admissible policies. The following is the *objective function* for maximizing expected revenue:

$$\max_{\pi\in\Pi} \mathbf{E}\left[\sum_{t=0}^{T-1} C_{t,\,t+2}(S_t, B_t^\pi(S_t)) + C_{\text{term}}(S_T) \,\middle|\, S_0\right],$$

where $C_{\text{term}}(S_T)$ represents a terminal contribution that is nondecreasing in R_T, L_T, and b_{T-1}.

We can express the optimal policy in the form of a stochastic dynamic program using Bellman's optimality equation (Bellman 1957). The optimal value function V^* is defined for each t and each state S_t:

$$V_t^*(S_t) = \max_{b_t\in\mathcal{B}}\big\{C_{t,\,t+2}(S_t, b_t) + \mathbf{E}(V_{t+1}^*(S_{t+1}) \mid S_t)\big\} \quad \text{for } t = 0, 1, 2, \ldots, T-1, \quad (13)$$

$$V_T^*(S_T) = C_{\text{term}}(S_T).$$

Figure 3 illustrates the above notation. Notice that at any decision epoch t, both the contribution and value functions are looking one step ahead, i.e., from $t+1$ onward, in the form of an expectation. Because of this, the *revenue* from t to $t+1$ becomes, in a sense, irrelevant. However, the link between the time periods comes from the dependence of R_{t+1} and L_{t+1} on R_t, L_t, and b_{t-1} (and of course, the random prices). In other words, at time t, our bid has to be placed for $(t+1, t+2]$ with an uncertain amount of resource, R_{t+1} in the battery. It is important to note that it is precisely because $C_{t,\,t+2}(S_t, b_t)$ *does not* include the revenue made in $(t, t+1]$ that allows us to show the important structural property of monotonicity for $C_{t,\,t+2}$ in b_{t-1} (see Proposition 3).

We now provide some results regarding the structure of the contribution and value functions. The algorithm (Monotone-ADP-Bidding) that we implement

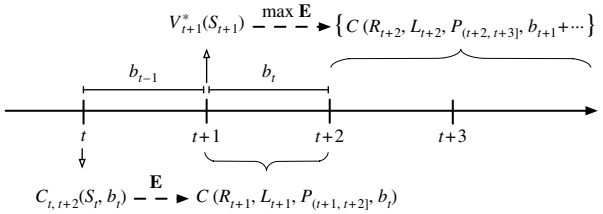


Figure 3 Illustration of the Dynamic Programming Notation

to solve for the optimal value function is inspired by the following monotonicity properties.

Proposition 3. *The contribution function $C_{t,\,t+2}(S_t, b_t)$, with $S_t = (R_t, L_t, b_{t-1}, P_t^S)$, is nondecreasing in R_t, L_t, b_{t-1}^-, and b_{t-1}^+.*

Proposition 4. *The optimal value function $V_t^*(S_t)$, with $S_t = (R_t, L_t, b_{t-1}, P_t^S)$, is nondecreasing in R_t, L_t, b_{t-1}^-, and b_{t-1}^+.*

4. Algorithmic Technique

The traditional way to solve for the optimal value function in (13) is by backward dynamic programming. Because this technique requires us to visit every state (which is computationally difficult), we propose the use of approximate dynamic programming. We first note that both methods require a finite state space. Because $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{B}$ were assumed to be finite, we need to assume, in particular, that $\mathcal{P}$ is also finite or that it is properly discretized.

The idea behind our ADP algorithm, which we call Monotone-ADP-Bidding (see Jiang and Powell 2015), is to iteratively learn the approximations $\bar{V}_t^n(S_t)$ (after n iterations) of $V_t^*(S_t)$ that obey the structural property of monotonicity. The algorithm is a form of asynchronous (or approximate) value iteration, therefore for each time t in iteration n, only one state S_t^n is visited. In addition, at each step, we perform a monotonicity preservation step to ensure the approximation is in a sense, structurally similar to V_t^*. We show experimentally that failure to maintain monotonicity, despite the availability of convergence proofs, produces an algorithm that simply does not work in practice.

4.1. Preliminaries

Let $\hat{v}_t^n(S_t^n)$ be an observation of the value of being in state S_t^n at iteration n and time t. Define the noise term

$$w_t^n(S_t^n) = \hat{v}_t^n(S_t^n) - \max_{b_t\in\mathcal{B}}\big\{C_{t,\,t+2}(S_t^n, b_t) + \mathbf{E}(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n)\big\}$$

to be the difference between the observation and the optimal value using the iteration $n-1$ *approximation*. We remark, for the sake of clarity, that this is *not* the noise representing the deviation from the true value, $V_t^*(S_t^n)$. Rather, $w_t^n(S_t^n)$ is the noise from an inability to exactly observe the optimal value of the maximization: $\max_{b_t\in\mathcal{B}}\{C_{t,\,t+2}(S_t^n, b_t) + \mathbf{E}(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n)\}$. Thus,

we can rearrange to arrive at

$$\hat{v}_t^n(S_t^n) = \max_{b_t \in \mathcal{B}} \{C_{t,t+2}(S_t^n, b_t) + \mathbf{E}(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n)\} + w_t^n(S_t^n).$$

Before we continue, let us define a partial order $\preccurlyeq$ on the state space $\mathcal{S}$ so that for $s = (r, l, b, p)$ and $s' = (r', l', b', p')$ where $r, r' \in \mathcal{R}$, $l, l' \in \mathcal{L}$, $b, b' \in \mathcal{B}$, and $p, p' \in \mathcal{P}$, we have that $s \preccurlyeq s'$ if and only if the following are satisfied:

$$(r, l, b) \le (r', l', b') \quad \text{and} \quad p = p'.$$

The values of any two states that can be related by $\preccurlyeq$ can be compared using Proposition 4. The main idea of the algorithm is that every observation $\hat{v}_t^n(S_t^n)$ is smoothed with the previous estimate of the value of S_t^n, and the resulting smoothed estimate $z_t^n(S_t^n)$ can be used to generalize to the rest of the state space by means of a monotonicity preserving operator, Π_M. Let $s \in \mathcal{S}$ be an arbitrary state that has a current estimated value of v. After $z_t^n(S_t^n)$ is known, Π_M adjusts the value of s in the following way:

$$\Pi_M(S_t^n, z_t^n, s, v) = \begin{cases} z_t^n & \text{if } s = S_t^n, \\ z_t^n \vee v & \text{if } S_t^n \preccurlyeq s,\ s \neq S_t^n, \\ z_t^n \wedge v & \text{if } s \preccurlyeq S_t^n,\ s \neq S_t^n, \\ v & \text{otherwise.} \end{cases} \tag{14}$$

First, we note that if monotonicity is already satisfied, then nothing changes because in the second and third cases of (14), we get that $z_t^n \vee v = v$ and $z_t^n \wedge v = v$, respectively. If, however, monotonicity is violated, then the newly observed value z_t^n prevails and replaces the previous value of v. Figure 4 shows an example of this operation for the two bids b_{t-1}^- and b_{t-1}^+. In the illustration, assume that the observations are made for fixed values of R_t and L_t, but note that when we run the algorithm, this adjustment is made over all four dimensions. The figure should be interpreted as a three-dimensional plot of the value function, where all state variables aside from b_{t-1} are fixed. Each bid pair $b_{t-1} = (b_{t-1}^-, b_{t-1}^+)$ is associated with a z-coordinate value represented by shades in gray scale (darker colors correspond to larger values). In the first and third plots, new observations arrive, and in the second and fourth plots, we see how the Π_M operator uses monotonicity to generalize the observed values to the rest of the state space.

Step 0a. Initialize $\bar{V}_t^0(s) = 0$ for each $t \le T-1$ and $s \in \mathcal{S}$.
Step 0b. Set $\bar{V}_T^n(s) = 0$ for each $s \in \mathcal{S}$ and $n \le N$.
Step 0c. Set $n = 1$.
Step 1. Select an initial state S_0^n.
Step 2. For $t = 0, 1, \ldots, (T-1)$:
  Step 2a. Sample a noisy observation:
  $\hat{v}_t^n(S_t^n) = \max_{b_t \in \mathcal{B}} \{C_{t,t+2}(S_t^n, b_t) + \mathbf{E}(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n)\} + w_t^n(S_t^n)$.
  Step 2b. Smooth in the new observation with previous value:
  $z_t^n(S_t^n) = (1 - \alpha_t^n(S_t^n))\bar{V}_t^{n-1}(S_t^n) + \alpha_t^n(S_t^n)\hat{v}_t^n(S_t^n)$.
  Step 2c. Perform monotonicity projection operator. For each $s \in \mathcal{S}$:
  $\bar{V}_t^n(s) = \Pi_M(S_t^n, z_t^n(S_t^n), s, \bar{V}_t^{n-1}(s))$.
  Step 2d. Choose the next state S_{t+1}^n given $\mathcal{F}^{n-1}$.
Step 3. If $n < N$, increment n and return **Step 1**.

Figure 5 Monotone-ADP-Bidding Algorithm for Training a Bidding Policy

The stepsize sequence used for smoothing in new observations with the previous approximation is denoted α_t^n, which can be thought of as a (possibly stochastic) sequence in n, for each t. Furthermore, states that are not visited do not get updated unless the update is made through the operator Π_M, so we also define

$$\alpha_t^n(s) = \alpha_t^{n-1} \mathbf{1}_{\{s = S_t^n\}}.$$

For notational purposes, let us also define the history of the algorithm until iteration n by the filtration

$$\mathcal{F}^n = \sigma\{(S_t^m, w_t^m(S_t^m))_{m \le n,\, t \le T}\}.$$

4.2. Algorithm Description and Convergence

The full description of the algorithm is given in Figure 5.

Monotone-ADP-Bidding can be shown to converge; we reproduce the set of assumptions and the resulting theorem here.

Assumption 1. *For all $s \in \mathcal{S}$ and $t \le T$,*

$$\sum_{n=1}^{\infty} \mathbf{P}(S_t^n = s \mid \mathcal{F}^{n-1}) = \infty \quad a.s.;$$

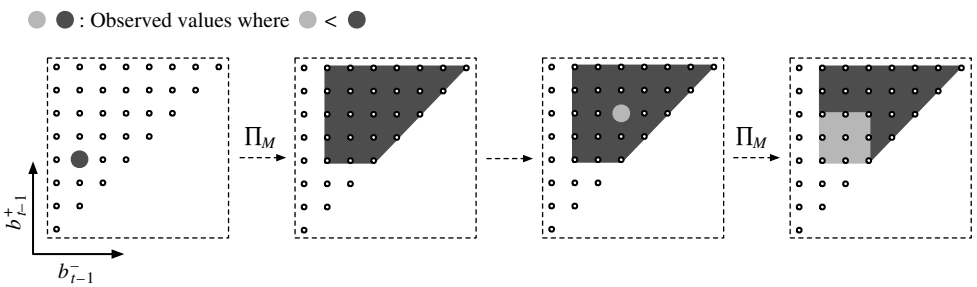


Figure 4 Illustration of Monotonicity Preservation (Darker Colors = Larger Values)

i.e., every state is visited infinitely often (see the Extended Borel–Cantelli lemma in Breiman 1992).

Assumption 2. *The optimal value function $V_t^*(s)$ and the observations $\hat{v}_t^n(S_t^n)$ are bounded above and below by $\pm V_{\max}$, where $V_{\max}$ is a positive constant.*

Assumption 3. *The noise sequence w_t^n satisfies the relation $\mathbf{E}[w_t^{n+1}(s) \mid \mathcal{F}^n] = 0$.*

Assumption 4. *For each $t \le T$ and state s, suppose $\alpha_t^n \in [0,1]$ is $\mathcal{F}^n$-measurable and*
(i) $\sum_{n=0}^{\infty} \alpha_t^n(s) = \infty$ *a.s.*,
(ii) $\sum_{n=0}^{\infty} \alpha_t^n(s)^2 < \infty$ *a.s.*

Theorem 1. *Under Assumptions 1–4, for each $t \le T$ and $s \in \mathcal{S}$, the estimates $\bar{V}_t^n(s)$ produced by the Monotone-ADP-Bidding Algorithm of Figure 5 converge to the optimal value function $V_t^*(s)$ almost surely.*

Proof. The proof is based on the result for a generalized MDP with a monotone value function in Jiang and Powell (2015).

4.3. Approximating the Expectation

Our algorithm can be applied to any model of spot prices P_t, with the caveat that more complex models generally require a higher dimensional state space. These include diffusion models (i.e., Schwartz 1997, Cartea and Figueroa 2005, Coulon et al. 2013), which often incorporate features such as Ornstein–Uhlenbeck processes, jump terms, and regime switching. Recently, there has also been interest in structural models of electricity prices, where the underlying supply, demand, and bid-stack behavior is taken into account; see Carmona and Coulon (2014) for a survey.

On one hand, the state space becoming larger or higher dimensional is indeed a computational difficulty that requires the availability of more powerful computational resources, but the convergence of the algorithm is unaffected (as long as P_t^S is properly discretized). On the other hand, any model without finite support (or finite, but with large cardinality) necessitates the approximation of an expectation using a sample mean in step 2a of the Monotone-ADP-Bidding algorithm (see Figure 5). In other words, because the expectation $\mathbf{E}(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n)$ of step 2a is, in general, impossible to compute, we must resort to letting $v_t^n(S_t^n)$ be the solution to the *sample average approximation* (see Kleywegt et al. 2002) problem:

$$\max_{b_t \in \mathcal{B}} \left\{ C_{t,\,t+2}(S_t^n, b_t) + J^{-1} \sum_{j=1}^{J} \bar{V}_{t+1}^{n-1}(S_{t+1}^j) \right\}, \tag{15}$$

where S_{t+1}^j are samples drawn independently from the distribution $S_{t+1} \mid S_t^n$. Suppose we take the observation $\hat{v}_t^n(S_t^n)$ to be the value of (15). By an interchange of the conditional expectation and the max operator, we see that:

$$\mathbf{E}\left[\max_{b_t \in \mathcal{B}} \left\{ C_{t,t+2}(S_t^n, b_t) + J^{-1} \sum_{j=1}^{J} \bar{V}_{t+1}^{n-1}(S_{t+1}^j) \right\} \,\middle|\, S_t^n \right]$$
$$\ge \max_{b_t \in \mathcal{B}} \left\{ C_{t,t+2}(S_t^n, b_t) + \mathbf{E}\big(\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t^n\big) \right\} \quad \text{a.s.},$$

and thus, after conditioning on $\mathcal{F}^{n-1}$ on both sides, we see that $\mathbf{E}[w_t^n(S_t^n) \mid \mathcal{F}^{n-1}]$ is biased upward from zero, a contradiction of Assumption 3. When J is large, we can certainly solve the sample average approximation problem in step 2a, apply the algorithm as is, and expect an effective heuristic solution. Practically speaking, our informal tests (using $J = 1{,}000$ on a diffusion price model) showed no significant convergence issues. Even so, we cannot claim that such an approximation produces a theoretically sound and convergent algorithm because of the biased noise term. This calls for us to propose another version of Monotone-ADP, for which Assumption 3 can be easily satisfied, without restricting to price process models that facilitate an easily computable expectation of the downstream value.

4.4. Post-Decision, Distribution-Free Approach

Using the idea of a post-decision state (see Powell 2011), we can make a small adjustment to the algorithm so that Assumption 3 is satisfied. In the case of the hourly bidding problem, the post-decision state S_t^b is the state-action pair (S_t, b_t). Oftentimes, post-decision states help simplify the computational aspect of an MDP, but unfortunately, for this problem instance, the post-decision state space is higher dimensional than is the predecision state space. Let $S_t^b = (S_t, b_t) \in \mathcal{S}^b$ and define the post-decision value function

$$V_t^b(S_t^b) = V_t^b(S_t, b_t) = \mathbf{E}(V_{t+1}^*(S_{t+1}) \mid S_t^b).$$

Notice that we can rewrite Bellman's optimality equation as

$$V_{t-1}^b(S_{t-1}^b) = \mathbf{E}\Big[\max_{b_t \in \mathcal{B}} \big\{ C_{t,\,t+2}(S_t, b_t) + V_t^b(S_t^b) \big\} \,\big|\, S_{t-1}^b \Big]. \tag{16}$$

Instead of attempting to learn V_t^*, the idea now is to algorithmically learn the post-decision value function V_t^b using the relation (16) and to implement the policy by solving

$$b_t^* = \arg\max_{b_t \in \mathcal{B}} \big\{ C_{t,\,t+2}(S_t, b_t) + V_t^b(S_t^b) \big\}.$$

Not surprisingly, the post-decision value function V_t^b also satisfies a monotonicity property over six dimensions.

Proposition 5. *The post-decision value function* $V_t^b(S_t^b)$, *with* $S_t^b = (R_t, L_t, b_{t-1}, b_t, P_t^S)$ *is nondecreasing in* R_t, L_t, b_{t-1}^-, b_{t-1}^+, b_t^-, *and* b_t^+.

Let $\bar{V}_t^{b,n}$ be the iteration n approximation of the post-decision value function, $S_t^{b,n}$ be the state visited by the algorithm in iteration n, $\hat{v}_t^{b,n}(S_t^{b,n})$ be an observation of $V_t^b(S_t^{b,n})$ using the iteration $n-1$ approximation, $w_t^{b,n}(S_t^{b,n})$ be the observation noise, $\mathcal{F}^{b,n}$ be a filtration defined analogously to $\mathcal{F}^n$, and Π_M^b be the monotonicity preservation operator on $\mathcal{S}^b$ defined analogously to Π_M. More precisely,

$$\begin{aligned}
&\hat{v}_t^{b,n}(S_t^{b,n}) \\
&\quad = \mathbf{E}\Big[\max_{b_{t+1}\in\mathcal{B}}\big\{C_{t+1,t+3}(S_{t+1}, b_{t+1}) + \bar{V}_{t+1}^{b,n-1}(S_{t+1}^b)\big\} \,\big|\, S_t^{b,n}\Big] \\
&\qquad + w_t^{b,n}(S_t^{b,n}), \\
&\mathcal{F}^{b,n} = \sigma\big\{\big(S_t^{b,m}, w_t^{b,m}(S_t^{b,m})\big)_{m\le n,\, t\le T}\big\},
\end{aligned} \tag{17}$$

and

$$\begin{aligned}
&\Pi_M^b(S_t^{b,n}, z_t^{b,n}, s, v) \\
&\qquad = \begin{cases}
z_t^{b,n} & \text{if } s = S_t^{b,n}, \\
z_t^{b,n} \vee v & \text{if } S_t^{b,n} \preccurlyeq^b s,\ s \ne S_t^{b,n}, \\
z_t^{b,n} \wedge v & \text{if } s \preccurlyeq^b S_t^{b,n},\ s \ne S_t^{b,n}, \\
v & \text{otherwise,}
\end{cases}
\end{aligned} \tag{18}$$

where $s = (r, l, b_1, b_2, p) \preccurlyeq^b s' = (r', l', b_1', b_2', p')$ with $r, r' \in \mathcal{R}$, $l, l' \in \mathcal{L}$, $b_1, b_1', b_2, b_2' \in \mathcal{B}$, and $p, p' \in \mathcal{P}$ if and only if

$$(r, l, b_1, b_2) \le (r', l', b_1', b_2') \quad \text{and} \quad p = p'.$$

The new algorithm for post-decision states is shown in Figure 6, and a set of analogous assumptions are provided below. We remark that by definition, $V_{T-1}^b(S_{T-1}^b) = \mathbf{E}[C_{\text{term}}(S_T) \mid S_{T-1}^b]$; thus, we only need to loop until $T-2$ in step 2.

Assumption 5. *For all* $s \in \mathcal{S}^b$ *and* $t \le T$,

$$\sum_{n=1}^{\infty} \mathbf{P}(S_t^{b,n} = s \mid \mathcal{F}^{b,n-1}) = \infty \quad a.s.$$

Assumption 6. *The optimal post-decision value function* $V_t^b(s)$ *and the observations* $\hat{v}_t^{b,n}(S_t^{b,n})$ *are bounded above and below by* $\pm V_{\max}$.

Assumption 7. *The noise sequence* $w_t^{b,n}$ *satisfies the relation* $\mathbf{E}[w_t^{b,n+1}(s) \mid \mathcal{F}^{b,n}] = 0$.

The advantage to applying this revised algorithm is that even when we cannot compute the expectation in step 2a and must rely on sample paths, we can still easily satisfy Assumption 7 (the unbiased noise assumption), unlike in the predecision case. To do so, we simply use

$$\hat{v}_t^{b,n}(S_t^{b,n}) = \max_{b_{t+1}\in\mathcal{B}}[C_{t+1,t+3}(S_{t+1}^n, b_{t+1}) + \bar{V}_{t+1}^{b,n-1}(S_{t+1}^n, b_{t+1})],$$

Step 0a. Initialize $\bar{V}_t^{b,0}(s) = 0$ for each $t \le T-1$ and $s \in \mathcal{S}^b$.
Step 0b. Set $\bar{V}_T^{b,n}(s) = 0$ for each $s \in \mathcal{S}^b$ and $n \le N$.
Step 0c. Set $n = 1$.
Step 1. Select an initial state $S_0^{b,n} = (S_0^n, b_0^n)$.
Step 2. For $t = 0, \ldots, (T-2)$:
  Step 2a. Sample a noisy observation:
  $\hat{v}_t^{b,n}(S_t^{b,n}) = \mathbf{E}[\max_{b_{t+1}\in\mathcal{B}}\{C_{t+1,t+3}(S_{t+1}, b_{t+1}) + \bar{V}_{t+1}^{b,n-1}(S_{t+1}^b)\} \mid S_t^{b,n}] + w_t^{b,n}(S_t^{b,n})$.
  Step 2b. Smooth in the new observation with previous value:
  $z_t^{b,n}(S_t^{b,n}) = (1-\alpha_t^n(S_t^{b,n}))\bar{V}_t^{b,n-1}(S_t^{b,n}) + \alpha_t^n(S_t^{b,n})\hat{v}_t^{b,n}(S_t^{b,n})$.
  Step 2c. Perform monotonicity preservation operator. For each $s \in \mathcal{S}^b$:
  $\bar{V}_t^{b,n}(s) = \Pi_M^b(S_t^{b,n}, z_t^{b,n}(S_t^{b,n}), s, \bar{V}_t^{b,n-1}(s))$.
  Step 2d. Choose the next state $S_{t+1}^{b,n}$ given $\mathcal{F}^{b,n-1}$.
Step 3. If $n < N$, increment n and return **Step 1**.

Figure 6 Monotone-ADP-Bidding Algorithm Using Post-Decision States

where we transition from $S_t^{b,n}$ to S_{t+1}^n using a *single sample outcome* of prices $P_{(t,t+1]}$. Hence, the noise term $w_t^{b,n}(S_t^{b,n})$ is trivially unbiased.

Along with allowing us to work with more complex price models, the revised algorithm gives us another important advantage, especially for implementation in practice/industry. As long as historical data are available, a model of the real-time prices is *not required* to train the algorithm. We propose an alternative idea: instead of fitting a stochastic model to historical data and then sampling $P_{(t,t+1]}$ from the model, we can simply take a price path directly from historical data. Since no specific knowledge regarding the distribution of prices is needed (besides the boundedness assumption needed for the convergence of the ADP algorithm), as previously mentioned, we refer to this as a *distribution-free* approach, and the technique is employed in §6. We now state the convergence theorem for the post-decision state version of Monotone-ADP-Bidding. Because the convergence theory for the post-decision state version is not discussed in detail in Jiang and Powell (2015), we provide a sketch of the proof here. First, we define the following post-decision Bellman operator that acts on a vector of values $V \in \mathbb{R}^{T\cdot|\mathcal{S}^b|}$ (any V, not necessarily corresponding to the optimal value function) for $S_t^b \in \mathcal{S}^b$ and $t \le T$:

$$\begin{aligned}
&(HV)_t(S_t^b) \\
&\quad = \begin{cases}
\mathbf{E}\Big[\max_{b_{t+1}\in\mathcal{B}}\big\{C_{t+1,t+3}(S_{t+1}, b_{t+1}) + V_{t+1}(S_{t+1}^b)\big\} \,\big|\, S_t^b\Big] \\
\qquad\qquad \text{for } t = 0, 1, 2, \ldots, T-2, \\
\mathbf{E}[C_{\text{term}}(S_{t+1}) \mid S_t^b] \quad \text{for } t = T-1.
\end{cases}
\end{aligned} \tag{19}$$

Step 2a of the algorithm (Figure 6) can thus be rewritten as

$$\hat{v}_t^{b,n}(S_t^{b,n}) = (H\bar{V}^{b,n-1})_t(S_t^{b,n}) + w_t^{b,n}(S_t^{b,n}).$$

Theorem 2. *Under Assumptions 4–7, for each $t \le T$ and $s \in \mathcal{S}^b$, the estimates $\bar{V}_t^{b,n}(s)$ produced by the post-decision version of Monotone-ADP-Bidding Algorithm of Figure 6 converge to the optimal post-decision value function $V_t^b(s)$ almost surely.*

Before discussing the proof, we state two necessary lemmas (proofs available in the online supplement). The idea of the first lemma is attributed to Tsitsiklis (1994).

Lemma 1. *Define deterministic bounding sequences L_t^k and U_t^k in the following way. Let $U^0 = V^* + V_{\max} \cdot e$ and $L^0 = V^* - V_{\max} \cdot e$, where e is a vector of ones. In addition, $U^{k+1} = (U^k + HU^k)/2$ and $L^{k+1} = (L^k + HL^k)/2$. Then for each $s \in \mathcal{S}^b$ and $t \le T-1$,*

$$L_t^k(s) \to V_t^b(s),$$

$$U_t^k(s) \to V_t^b(s),$$

where the limit is in k.

Lemma 2. *U^k and L^k both satisfy the monotonicity property: for each t, k, and $s_1, s_2 \in \mathcal{S}^b$ such that $s_1 \preccurlyeq^b s_2$,*

$$\begin{aligned} U_t^k(s_1) &\le U_t^k(s_2), \\ L_t^k(s_1) &\le L_t^k(s_2). \end{aligned} \tag{20}$$

Sketch of Proof of Theorem 2. With Lemmas 1 and 2, we can proceed to show convergence of the post-decision state version of Monotone-ADP using the general steps to prove convergence of Monotone-ADP for *predecision* states taken in Jiang and Powell (2015). The steps are as follows:

1. Given a fixed k and a state $s \in \mathcal{S}^b$ such that s is increased *finitely often* by the monotonicity preservation operator Π_M^b, we can show that for any sufficiently large n,

$$L_t^k(s) \le \bar{V}_t^{b,n}(s) \le U_t^k(s). \tag{21}$$

There exists at least one such state, i.e., the minimal state $(0, 0, (b_{\min}, b_{\min}), P_t^S)$. Repeat the argument for states that are decreased finitely often by Π_M^b.

2. Next, we must show that states s that are affected by Π_M^b *infinitely often* also satisfy (21). This leverages the fact that the result has already been proven for states that are affected finitely often. The idea is that if all states immediately less than s (i.e., x is immediately less than y if $x \preccurlyeq^b y$ and there does not exist z such that $x \preccurlyeq^b z \preccurlyeq^b y$) satisfy (21), then s satisfies (21) as well. Lemma 2 and an induction argument are used in this part of the proof.

3. Finally, combining Lemma 1 along with the fact that all post-decision states $s \in \mathcal{S}^b$ satisfy (21), it is easy to see that from a type of squeeze argument,

$$\bar{V}_t^{b,n}(s) \to V_t^b(s),$$

for each t and s, as desired.

Note that both steps (1) and (2) require Assumption 7, hence the focus that we have placed on it in this paper.

4.5. Stepsize Selection

The selection of the stepsize α_t^n, also known as a *learning rate*, can have a profound effect on the speed of convergence of an ADP algorithm. A common example of stepsize rule that satisfies Assumption 4 is simply

$$\alpha_t^n = \frac{1}{N(S_t^n, n)},$$

where $N(S_t^n, n) = \sum_{m=1}^n \mathbf{1}_{\{S_t^m = S_t^n\}}$ is the number of visits by the algorithm to the state S_t^n. The issue is that this method weighs all observations equally, even though we know that the error can be extremely large in early iterations of any ADP algorithm. See Powell (2011, Chap. 11) for an overview of the numerous available stepsize rules.

After some experimentation, we found that the bias-adjusted Kalman Filter (BAKF) developed in George and Powell (2006), performed better than simpler alternatives. The main idea behind BAKF is to choose α_t^n such that the mean squared error to the true value function is minimized; we omit the details and refer interested readers to the original paper.

5. Benchmarking on Stylized Problems Using Predecision Monotone-ADP

In this section, we present results of running Monotone-ADP-Bidding and traditional approximate value iteration on a tractable problem (i.e., the optimal solution is computable) to show the advantages of using Π_M. In this section, we consider both four- and five-dimensional versions of the sequential bidding problem. We first describe some simplifications to make benchmarking possible.

To benchmark the algorithm against a truly optimal solution, we make some simplifying assumptions (to be relaxed in §6) so that backward dynamic programming can be used to compute an optimal solution. First, we suppose that P_t has finite support and that $M = 1$, so the exact value of $\mathbf{E}(V_{t+1}(S_{t+1}) \mid S_t)$ can be computed easily. When M is larger, we can only compute an approximation to the expectation because an exponential in M number of outcomes of the price process need to be considered for an exact result.

In addition, in the numerical work of this paper, we take the traditional approach and choose $C_{\text{term}}(s) = 0$; however, we remark that this may not always be the best choice in practice. See §6.3 for further discussion on the issue of selecting a terminal contribution function.

To test the approximate policies, we compute a *value of the policy* in the following way. For a particular set of value function approximations $\bar{V}$, the set of decision functions can be written as

$$\bar{B}_t(S_t) = \arg\max_{b_t \in \mathcal{B}} \big[C_{t,t+2}(S_t, b_t) + \mathbf{E}(\bar{V}_{t+1}(S_{t+1}) \mid S_t)\big].$$

For a sample path $\omega \in \Omega$, let

$$F(\bar{V}, \omega) = \sum_{t=0}^{T+1} C\big(R_{t+1}(\omega), L_{t+1}(\omega), P_{(t+1,t+2]}(\omega), \bar{B}_t(S_t)\big)$$

be a sample outcome of the revenue. We report the empirical value of the policy, which is the sample mean of $F(\bar{V}, \omega)$ over 1,000 sample paths ω.

5.1. Variation 1

First, we consider a four-dimensional variation of the bidding problem, where $S_t = (R_t, L_t, b^-_{t-1}, b^+_{t-1})$. In particular, we assume that the price process has no state variables. Several versions of this problem are explored by altering the parameter values: a typical size for the batteries under consideration for the energy arbitrage application is $R_{\max} = 6$ MWh, but we also allow values of $R_{\max} = 12$ MWh and $R_{\max} = 18$ MWh for variety. The decision space is fixed in the following way: we set $b_{\min} = 15$ and $b_{\max} = 85$, and discretized linearly between $b_{\min}$ and $b_{\max}$ for a total of 30 possible values in each dimension of the bid. The price process P_t has the form

$$P_t = S(t) + \epsilon_t,$$

where the sinusoidal (representing the hour-of-day effects on price) deterministic component is

$$S(t) = 15\sin(2\pi t/24) + 50,$$

and $\epsilon_t \in \{0, \pm 1, \pm 2, \ldots, \pm 20\}$, a sequence of mean zero, independent, and identically distributed (i.i.d.) random variables distributed according to the *discrete pseudonormal distribution* with $\sigma_X^2 = 49$ (a discrete distribution where the probability masses are defined by the evaluating at the density function of $\mathcal{N}(0, \sigma_X^2)$ and then normalizing). We consider both the cases where the battery age does and does not matter (by setting $\beta(l) = 1$), in effect introducing an irrelevant state variable. When aging does matter, the aging function we use is $\beta(l) = (l/L_{\max})^{1/6}$, which provides a roughly linear decline in efficiency from 100% to around 70%, followed by a much steeper decline. Lastly, in Problem 4, we considered a uniform distribution for the noise, while the remaining problems used pseudonormal noise. In line with the operation procedures of the NYISO, the undersupply penalty parameter K is set to 1 in our simulations—this means that if one is unable to deliver energy to the market, then the penalty is precisely the current spot price (essentially, we are paying another generator to produce the energy instead). The different problem instances, labeled A_1–F_1, along with their state space cardinalities are summarized in Table 1.

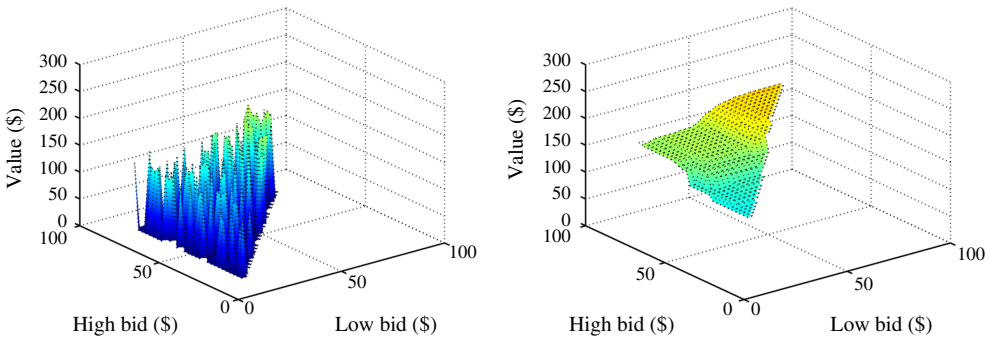
Table 1 Parameter Choices for Variation 1 Benchmark Problems

Problem	T	$R_{\max}$	$L_{\max}$	$\beta(l)$	Distribution of ϵ_t	Cardinality of $\mathcal{S}$
A_1	24	6	8	1	Pseudonormal	22,320
B_1	24	6	8	$(l/8)^{1/6}$	Pseudonormal	22,320
C_1	36	6	8	1	Pseudonormal	22,320
D_1	24	12	12	$(l/12)^{1/6}$	Uniform	66,960
E_1	24	12	12	$(l/12)^{1/6}$	Pseudonormal	66,960
F_1	36	18	18	$(l/18)^{1/6}$	Pseudonormal	150,660

5.2. Numerical Results for Variation 1

We first evaluate the effectiveness of Monotone-ADP-Bidding versus approximate value iteration, a traditional ADP algorithm (exactly the same as Monotone-ADP-Bidding with Π_M removed); the results for Variation 1 are given in Table 2.

Figure 7 gives a quick visual comparison between the two types of approximate value functions, generated



(a) AVI, $N = 1{,}000$ (b) M-ADP, $N = 1{,}000$

Figure 7 (Color online) Visual Comparison of Value Function Approximations for $t = 12$ and $R_t = 3$

Table 2 **Percentage of Optimality for Policies Generated from the M-ADP and AVI Algorithms for Variation 1**

Iterations	Algorithm	Problem (%)					
		A_1	B_1	C_1	D_1	E_1	F_1
$N = 1{,}000$	M-ADP	58.9	67.8	73.5	60.7	56.8	45.9
	AVI	53.7	45.7	66.6	23.4	24.8	7.8
$N = 5{,}000$	M-ADP	83.7	82.8	87.2	73.8	66.1	64.1
	AVI	60.7	67.3	82.1	43.8	52.6	49.0
$N = 9{,}000$	M-ADP	89.4	93.6	93.3	76.2	74.9	86.6
	AVI	70.2	75.8	85.3	46.7	58.8	57.3
$N = 13{,}000$	M-ADP	93.8	89.9	96.8	79.8	83.7	88.5
	AVI	76.3	83.1	87.8	49.8	68.2	57.8
$N = 17{,}000$	M-ADP	95.8	96.4	97.8	82.7	86.8	91.4
	AVI	78.0	85.1	90.7	62.2	72.0	70.6
$N = 21{,}000$	M-ADP	95.0	98.4	98.1	90.5	87.8	92.7
	AVI	81.1	87.7	90.0	61.0	73.7	76.3
$N = 25{,}000$	M-ADP	97.0	98.5	98.5	89.7	90.4	94.8
	AVI	86.4	89.4	92.1	60.0	75.1	76.0

by approximate value iteration and Monotone-ADP-Bidding. We remark that after $N = 1{,}000$ iterations, the value function approximation in Figure 7(b) obtained by exploiting monotonicity has developed a discernible shape and structure, with a relatively wide range of values. The result in Figure 7(a), on the other hand, is relatively unusable as a policy.

We notice that as the cardinality of the state space increases, the value of monotonicity preservation becomes more pronounced. This is especially evident in problem F, where after $N = 1{,}000$ iterations, Monotone-ADP-Bidding achieves 45.9% optimality, whereas traditional approximate value iteration does not even reach 10%. Although this finite state, lookup table version of approximate value iteration (for lookup table) is also a convergent algorithm (see Bertsekas and Tsitsiklis 1996, Proposition 4.6), its performance is markedly worse, especially when the state space is large. Because it exploits the monotone structure, Monotone-ADP-Bidding has the ability to quickly attain the general shape of the value function. Figure 8 illustrates this by showing the approximations at early iterations of the algorithm. These numerical results suggest that the convergence rate of the ADP algorithm is substantially increased through the use of the monotonicity preserving operation.

With the effectiveness of Monotone-ADP-Bidding on Variation 1 established, we now examine its *computational* benefits over backward dynamic programming. A comparison of CPU times between Monotone-ADP-Bidding and backward dynamic programming is shown in Figure 9, where the horizontal axis is in log scale. Once again, we notice the order of magnitude difference in computation time for the exact solution and for the near-optimal ADP solution. Indeed from Table 3, we see that we can achieve very good solutions using an ADP approach while cutting computational resources by more than 93%. In the most drastic case, problem F (over 150,000 states), a 95% optimal solution is achieved using only 4% the amount of computational power.

5.3. Variation 2

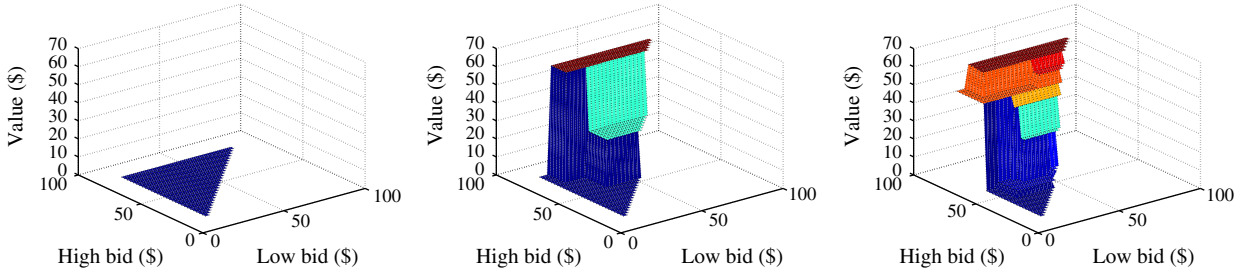
Briefly, we also consider a problem with a more complex price process: a Markov regime-switching model with two regimes denoted by the process X_t. We represent the normal regime as $X_t = 0$ and the spike regime as $X_t = 1$. Let $S(t)$ be a deterministic seasonal component, ϵ_t be discrete i.i.d. random variables representing noise in the normal regime, and ϵ_t^s be discrete i.i.d. random variables representing noise in the spike regime. The price process can be written as

$$P_t = S(t) + \mathbf{1}_{\{X_t=0\}} \cdot \epsilon_t + \mathbf{1}_{\{X_t=1\}} \cdot \epsilon_t^s.$$

Also, we define the transition probabilities of the (time-inhomogeneous) Markov chain X_t:

$$p_{i,j}(t) = \mathbf{P}(X_{t+1} = j \mid X_t = i).$$

Because X_t only takes two states, let $p(t) = p_{0,1}(t)$ (the probability, at time t, of moving from the normal regime into the spike regime) and $q(t) = p_{1,0}(t)$ (the probability, at time t, of returning to the normal regime). The state variable for this problem is five-dimensional: $S_t = (R_t, L_t, b_{t-1}^-, b_{t-1}^+, X_t)$. To generate a



(a) Iteration $n = 0$ (b) Iteration $n = 10$ (c) Iteration $n = 50$

Figure 8 **(Color online) Value Function Approximations from Early Iterations of Monotone-ADP**

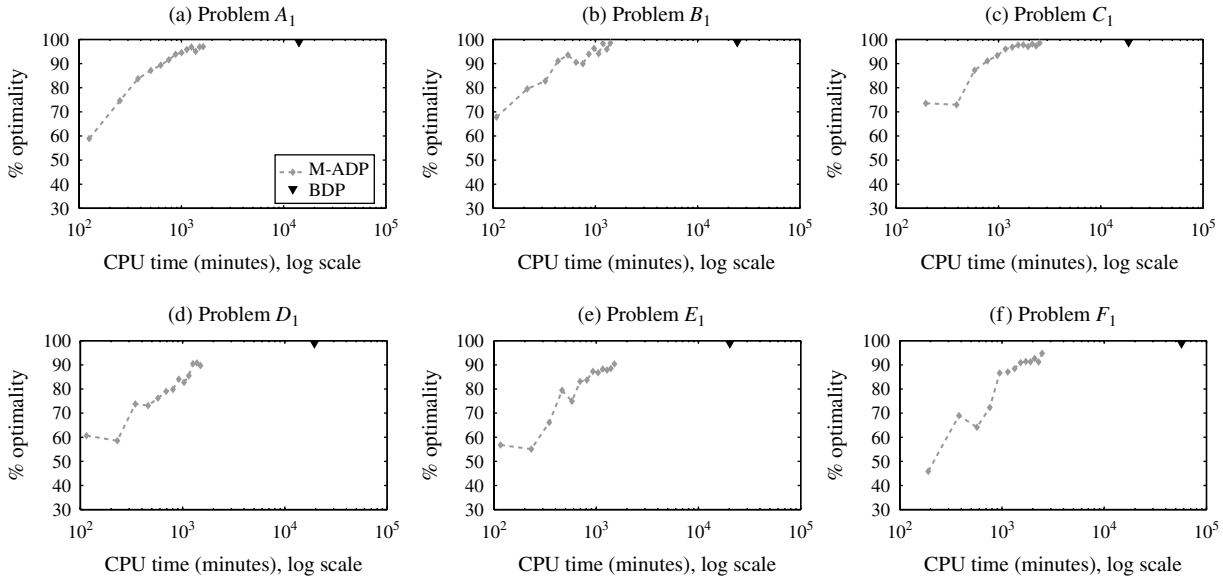


Figure 9 Computation Times of M-ADP vs. BDP for Variation 1

small library of test problems, we considered two versions of the seasonal component:

$$S_i(t) = 15 f_i(2\pi t/12) + 50,$$

for $i \in \{1, 2\}$ and $f_1(x) = \sin(x)$ and $f_2(x) = \cos(x)$. We roughly model the fact that price spikes tend to occur more frequently when demand is high. Since demand is often modeled using sinusoidal functions, we use the following for $p(t)$ (the probability of moving from the normal regime to the spike regime) when the seasonal component is $S_i(t)$:

$$p_i(t) = \alpha_p[f_i(2\pi t/12) + 1]/2,$$

for some parameter $\alpha_p \leq 1$, representing the maximum probability of moving to the spike regime: $p_t(t) \in [0, \alpha_p]$. In these numerical results, $q(t)$, the probability of returning to the normal regime, is always modeled as a constant α_q. Moreover, both ϵ_t and ϵ_t^s have support $\{-10, -9, -8, \ldots, +39, +40\}$ and are distributed according to the discrete pseudonormal distribution (with parameters $(\mu_X, \sigma_X) = (0, 7)$ and $(\mu_X, \sigma_X) = (15, 20)$, respectively). The skewed support allows us to model the preponderance of upward spikes in electricity spot prices. The remainder of the parameters vary across the test problems and are summarized in Table 4.

We ran both Monotone-ADP-Bidding and traditional approximate value iteration for 10,000 iterations on each of the test problems. The results of the benchmarking are summarized in Table 5 (for brevity, we omit plots of the approximate value function and

Table 3 Time Savings from BDP When Using M-ADP Near-Optimal Solution (%)

	Problem					
	A_1	B_1	C_1	D_1	E_1	F_1
BDP CPU time (minutes)	14,112	24,392	18,720	19,448	20,256	56,968
M-ADP CPU time (mins.)/Percentage of Optimality (%)	1,003/95	1,077/94	1,167/96	1,264/90	1,506/90	2,470/95
CPU time savings (%)	93	96	94	94	93	96

Table 4 Parameter Choices for Variation 2 Benchmark Problem

Problem	T	R_{max}	L_{max}	$\beta(l)$	Trend	α_p	α_q	Cardinality of $\mathcal{S}$
A_2	24	4	6	$(l/6)^{1/6}$	$S_2(t)$	0.9	0.5	22,320
B_2	24	4	8	$(l/8)^{1/6}$	$S_1(t)$	0.8	0.7	29,760
C_2	12	8	6	$(l/6)^{1/6}$	$S_2(t)$	0.9	0.5	44,640
D_2	12	6	8	$(l/8)^{1/6}$	$S_2(t)$	0.8	0.7	44,640
E_2	12	8	10	$(l/10)^{1/6}$	$S_1(t)$	0.9	0.5	74,400
F_2	12	10	8	$(l/8)^{1/6}$	$S_2(t)$	0.8	0.7	74,400

Table 5 **Percentage of Optimality for Policies Generated from the M-ADP and AVI Algorithms for Variation 2**

Iterations	Algorithm	Problem (%)					
		A_2	B_2	C_2	D_2	E_2	F_2
$N = 2{,}000$	M-ADP	82.4	82.6	94.6	93.6	82.8	82.8
	AVI	31.3	32.2	43.6	60.9	33.0	46.2
$N = 4{,}000$	M-ADP	86.7	83.7	96.1	99.1	93.2	90.0
	AVI	53.4	46.1	62.1	76.9	54.0	62.4
$N = 6{,}000$	M-ADP	93.6	81.0	88.3	98.2	90.2	90.5
	AVI	64.8	51.0	69.3	82.6	63.5	76.1
$N = 8{,}000$	M-ADP	95.3	86.8	92.2	93.8	93.4	88.8
	AVI	77.4	68.0	67.5	79.6	77.0	77.0
$N = 10{,}000$	M-ADP	94.4	87.8	95.8	96.3	95.2	98.2
	AVI	84.1	58.7	77.9	71.8	84.3	60.8

computation times and instead state that they are very comparable to those of Variation 1). It is clear that, once again, Monotone-ADP-Bidding provides significantly better solutions than approximate value iteration, particularly in the early iterations.

6. Case Study: Training and Testing an ADP Policy Using Real NYISO Data

In this section, we use the distribution-free, post-decision state version of Monotone-ADP to produce bidding policies for the New York City zone of the NYISO, with the goal of demonstrating the idea of training using only historical data as "sample paths." The case study uses two full years of five-minute real-time price data obtained from the NYISO, for the recent years of 2011 and 2012. See Figure 10 for a visual comparison.

The concern with the historical prices is that we must satisfy Assumption 6; i.e., we must assume that the unknown stochastic process P_t is bounded. This is not an unreasonable assumption if we allow the bound to be high, say \$3,000, which is consistent with the prices in our data set. We remark again that, to satisfy Assumption 7, we use

$$\hat{v}_t^{b,n}(S_t^{b,n}) = \max_{b_{t+1}\in\mathcal{B}}[C_{t+1,t+3}(S_{t+1}^n, b_{t+1}) + \bar{V}_{t+1}^{b,n-1}(S_{t+1}^n, b_{t+1})],$$

in Step 2a of Figure 6, where the transition from $S_t^{b,n}$ to S_{t+1}^n is accomplished using a single sample from historical data. The remaining assumptions are satisfied for the same reasons as before.

6.1. ADP Policies

There are many sensible ways to choose training data for a specific operating time horizon. In this paper, we consider two commonsense methods: (1) using historical samples from the same month of the previous year to train a policy for the current month ("ADP Policy 1") and (2) using samples from the previous month to train a policy for the current month ("ADP Policy 2"). The rationale for the first method is that the price process may behave similarly in the same month across years (though factors like weather, natural gas prices, etc., should be considered before assuming that such a statement is true), and the rationale for the second method is simply using the most recent data available. We train an hourly bidding policy that has a horizon of one day ($T + 1 = 24$) and the post-decision state variable for this case study is five-dimensional:

$$S_t^b = (R_t, b_{t-1}^-, b_{t-1}^+, b_t^-, b_t^+) \in \mathcal{S}^b,$$

where the bids are linearly discretized between $b_{\min} = 0$ and $b_{\max} = 150$ into 15 grid points in each dimension. Although it is difficult to discern from Figure 10, 98.2% of the prices in our data set are below \$150. To have a lower dimensional state variable for more reasonable runtimes, we elect to assume $P_t^S = \{\}$ and $\beta(l) = 1$ (it is also not typical for a battery manufacturer to provide an expression for $\beta(l)$; an accurate model for $\beta(l)$ would require estimation from empirical data, which is outside the scope of this paper). Conversations with industry colleagues suggested that, for this application, it is reasonable to model a 1 MW, 6 MWh battery. Because $M = 12$, we choose $R_{\max} = 72$, giving us a state space of size $|\mathcal{S}^b| = 3.6$ million states, much larger than that of the benchmark problems in §5. The remaining details are summarized in the following list.

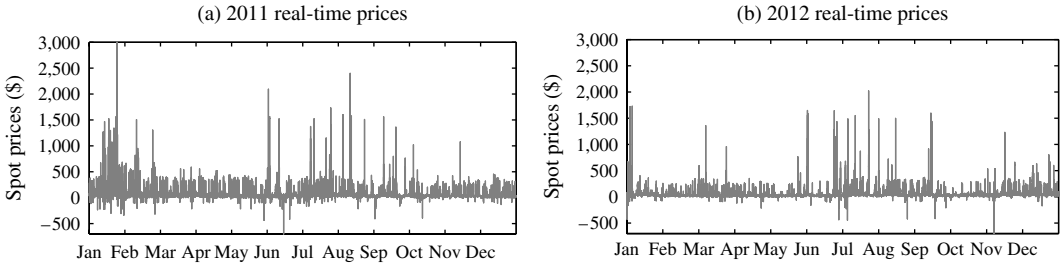


Figure 10 **NYISO Real-Time, Five-Minute Prices Used for Training and Testing ADP Policies**

1. Because the characteristics of the spot prices can be very different on weekends (see, e.g., Coulon et al. 2013), we considered weekdays only. In a true application, it would be important to train a separate policy for weekends.

2. To have a larger data set for our simulations, our main assumption is that spot prices of a particular hour are identically distributed across weekdays of the same month, allowing us to train and test on a large set of sample paths.

3. We train a daily value function for each month of the year. In essence, we combine the data for the weekdays of each month to produce a policy that is valid for any given weekday of the same month.

4. As before, we set the undersupply penalty parameter K to 1.

The real-time prices from 2011 are used as training data and the prices from 2012 are used simultaneously as training and test data: for each month of 2012, we generate two policies: one trained using data from the same month in 2011 and the other trained using data from the previous month. The revenues generated by these policies are given in Table 6, where the evaluation method from §5 is used. The results correspond to running the algorithm for $N = 100{,}000$ iterations. Note that because the post-decision version does not compute an expectation, each iteration is significantly faster than that of the pre-decision version but in general requires more iterations. The results show that ADP Policy 1 (training on data from the same month of the previous year) narrowly outperforms ADP Policy 2 (training on data from the previous month) in most cases. Although our MDP only optimizes for revenue in expectation, we nevertheless report that the (0.05-quantile, 0.95-quantile) of daily revenue for ADP Policy 1 is ($60.37, $474.24) with a median of $174.55. For ADP Policy 2, we have a (0.05-quantile, 0.95-quantile) of daily revenue of ($44.13, $453.30) with a median of $154.12. These results confirm that the policies consistently generate revenue.

Table 6 Performance of Monotone-ADP-Bidding Policy Trained and Tested on Real Data

	ADP Policy 1		ADP Policy 2	
Test data set	Training data set	Revenue ($)	Training data set	Revenue ($)
January-12	January-11	6,539.00	December-11	7,857.69
February-12	February-11	1,966.02	January-12	2,061.99
March-12	March-11	5,810.70	February-12	5,511.07
April-12	April-11	4,147.60	March-12	4,223.85
May-12	May-11	9,030.54	April-12	8,296.17
June-12	June-11	11,465.39	May-12	10,934.07
July-12	July-11	11,323.50	June-12	9,042.77
August-12	August-11	6,277.31	July-12	6,206.56
September-12	September-11	5,754.93	August-12	5,561.24
October-12	October-11	3,693.01	September-12	3,623.33
November-12	November-11	7,228.85	October-12	2,768.00
December-12	December-11	3,275.84	November-12	3,160.28
Yearly revenue:		76,512.68		69,247.02

6.2. Comparison to Standard Trading Policies

This section compares the ADP policies to several other rule-based policies developed from both the literature and discussions with industry. Because of the existence of (and lack of access to) proprietary trading strategies, we cannot claim to be comparing against the best; however, we do believe that the basic strategies surveyed in this paper are involved in a significant portion of high performing trading/bidding policies. Trading Policies A and B are based on determining peak and off-peak hours using historical price data and inspired by strategies considered in the paper by Walawalkar et al. (2007) and the technical report by Byrne and Silva-Monroy (2012), but adapted to our bidding setting. Trading Policy C uses the idea of bidding at certain quantiles and is attributed to ideas given to us by our industry colleagues. Policies subscripted by 1 (i.e., A_1, B_1, and C_1) use historical price data from the same month of the previous year, and policies subscripted by 2 use data from the previous month.

Policies A_1 and A_2: Taking advantage of the trend that lower prices occur at night, we split the operating day hours into two intervals 1 to h^* and h^*+1 to 24, with $h^* > 6$. The intervals are then sorted using average historical prices. If hour h of the first interval has one of the six lowest prices, then it is designated a *buy* interval. Similarly, if hour h of the second interval has one of the six highest prices, then it is a *sell* interval. All other hours are *idle* intervals. When placing a bid b_t, we consider the hour h corresponding to $(t+1, t+2]$: if hour h is a buy interval, we choose $b_t = (b_{\max}, b_{\max})$; if hour h is a sell interval, we choose $b_t = (b_{\min}, b_{\min})$; and if hour h is an idle interval, we choose $b_t = (b_{\min}, b_{\max})$. This policy essentially guarantees (with the possible exception of spike situations where prices exceed $b_{\max}$) that we fill up the battery in the interval from 1 to h^* and then empty it in the interval from h^*+1 to 24. With some tuning, we found that $h^* = 12$ provided the highest valued policies.

Policies B_1 and B_2: The second set of policies is again based on the idea of pairing periods of low prices with periods of high prices, but with more flexibility than Policies A_1 and A_2. Instead, we sort all hours of a given day using average historical prices and designate the k^* lowest priced hours as *buy* intervals, corresponding to $b_t = (b_{\max}, b_{\max})$ and the k^* highest priced hours as *sell* intervals, corresponding to $b_t = (b_{\min}, b_{\min})$. The remaining hours are *idle* intervals, meaning we set $b_t = (b_{\min}, b_{\max})$. Again using historical

prices, at time t, we estimate the level of resource $\hat{R}_{t+1}$ at the beginning of the next hour as the average of the outcomes of R_{t+1} over historical sample paths. When encountering a buy interval with $\hat{R}_{t+1} > 60$ (nearly full battery) or a sell interval with $\hat{R}_{t+1} < 12$ (nearly empty battery), we place the idle bid instead. Finally, if we detect that we have more energy in storage than can be sold in the time left until the end of horizon, we place sell bids thereafter. We report results for the tuned parameter $k^* = 10$.

Policies C_1 and C_2: Let $\alpha < 0.5$ be the parameter to our final set of policies. For each hour h, we compute the empirical quantiles of the historical prices at α and $1-\alpha$, denoted q_α and $q_{(1-\alpha)}$, respectively (note the suppressed dependence on h). When bidding at time t, we again estimate $\hat{R}_{t+1}$ using historical data. For times when the battery is estimated to be nearly full, we place the bid $b_t = (b_{\min}, q_{(1-\alpha)})$. Similarly, if the battery is nearly empty, we place the bid $b_t = (q_\alpha, b_{\max})$. For anything in between, we simply bid $b_t = (q_\alpha, q_{(1-\alpha)})$, with the hope of consistently buying low and selling high. We implement the same logic for when we hold more energy than the maximum that can be sold in the time remaining and initiate a sell-off. In the numerical results below, we use $\alpha = 0.1$. Smaller values of α correspond to the notion of reserving the battery for only the highest valued trades.

The results of running Policies A_i, B_i, and C_i are given in Table 7.

Given that they were afforded more nuanced actions than simply buy and sell, perhaps not surprisingly, Policies C_i outperformed the rest. However, we also notice that, unlike the other policies, Policy C_2 generated large negative revenues in July-12 and November-12. Comparing Policy C_1 against ADP Policy 1 and comparing Policy C_2 against ADP Policy 2, we see the revenues generated are still a disappointing 68.5% and 55.3%, respectively, of the ADP revenues, suggesting that it is difficult, even after tuning, for simple rule-based heuristics to perform at the level of a well-trained ADP policy that considers downstream value. Moreover, the months of July-12 and November-12 (during which Policy C_2 posted negative revenues) suggest that the ADP strategy is more robust to the differences in training data when compared to Policy C_i. A possible driving force behind Policy C_2's failure to generate revenue during these months is that the training data from June-12 and October-12 has largely differing characteristics (e.g., many spikes) from the testing data in July-12 and November-12 (see Figure 10).

6.3. Additional Insights

Applying Monotone-ADP-Bidding to real data from the NYISO has given us several insights into the topic of energy arbitrage. First, we note that for both ADP Policy 1 and ADP Policy 2 (see Table 6), the largest revenues were generated in the months of May, June, and July, presumably because of changes in weather. The difference between the revenues generated in the months of highest and lowest revenue, June and February, is more drastic than one might expect: Jun Revenue − Feb Revenue = \$9,499.37 for ADP Policy 1 and Jun Revenue − Feb Revenue = \$8,872.08 for ADP Policy 2. These results suggest that perhaps energy arbitrage should not be a year-round investment but rather one that is active only during months with potential for high revenue. As Sioshansi et al. (2009) conclude, when it comes to the value of energy storage, it is important to consider various revenue sources.

Costs of energy storage can be as low as \$160 kWh^{-1} today, and it is reasonable to expect that they will continue to decrease. As mentioned earlier, with optimal storage control strategies and decreased capital costs, energy arbitrage can soon become profitable on its own; as it currently stands, storage costs are

Table 7 Performance of Standard Trading Policies Trained and Tested on Real Data

	Revenue (\$)					
Test data set	Policy A_1	Policy A_2	Policy B_1	Policy B_2	Policy C_1	Policy C_2
January-12	3,078.56	3,539.68	3,182.07	3,445.84	1,901.89	8,461.97
February-12	707.02	404.68	397.38	(349.51)	1,503.52	1,487.59
March-12	2,380.97	2,343.57	1,837.57	2,154.49	4,744.29	6,214.73
April-12	702.84	1,247.13	205.12	1,078.62	3,403.25	3,412.50
May-12	5,855.13	3,564.74	4,888.52	3,797.41	6,944.26	5,013.73
June-12	3,449.75	4,742.00	4,511.81	3,427.11	7,329.25	7,618.00
July-12	6,871.67	4,488.28	6,940.68	6,781.36	8,003.43	(7,066.45)
August-12	1,278.66	1,482.63	1,824.57	1,273.28	4,724.14	4,908.08
September-12	1,438.39	1,638.63	315.94	1,665.22	3,868.75	4,336.50
October-12	701.91	751.93	633.58	321.80	2,879.64	2,750.99
November-12	1,585.50	1,938.98	1,354.96	1,359.01	4,438.00	(1,270.90)
December-12	1,240.97	1,012.26	424.56	431.05	2,703.46	2,445.46
Yearly revenue:	29,291.36	27,154.52	26,516.76	25,385.68	52,443.88	38,312.20

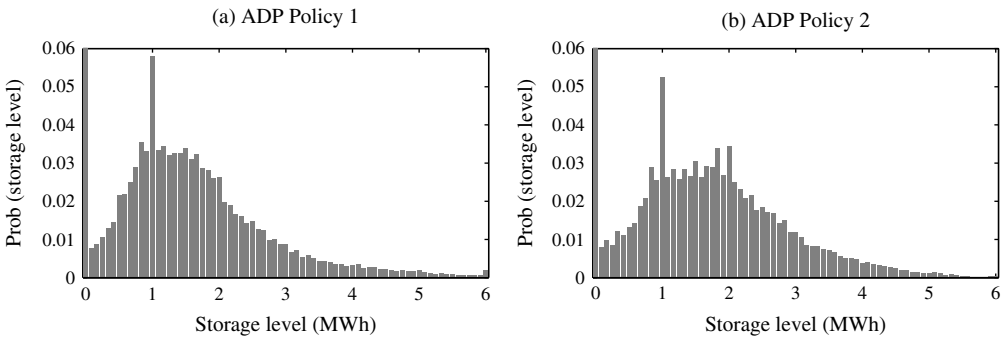


Figure 11 Empirical Distribution of Storage Level Using ADP Policies 1 and 2

still relatively high compared to potential revenue. Therefore, it is also imperative that the precise storage needs of our trading/bidding policies are well understood; it may be the case that in some months, one would choose to dedicate the entire battery to frequency regulation, whereas in high revenue months, the better strategy may be to use some of the capacity toward arbitrage. It is clear that some policies, such as Policies A_i, are designed with fully utilizing the available storage in mind, but for more complex policies such as those generated by Monotone-ADP-Bidding, the usage profiles are not obvious. Figure 11 shows the empirical distribution for the storage level of the battery (on an hourly basis) throughout the test data set. Note that for presentation purposes we have scaled the plot so that the bar at $R_t = 0$ is cut off; because of its designation as the initial state (and final state as well for most sample paths), its probability is skewed to 0.09 and 0.10 for the two plots, respectively. The high probability at 1 MWh is likely because it corresponds to full hourly charge, the minimum amount of energy needed to avoid the possibility of an undersupply penalty. The 0.9- and 0.95-quantiles for ADP Policy 1 occur at 3.00 MWh and 3.75 MWh, and for ADP Policy 2, they are 3.16 MWh and 3.75 MWh. This means for our (relatively short) daily trading horizon, a 6 MWh battery is unnecessary—a 33% smaller device with 4 MWh storage would have sufficed and delivered similar results at a steep discount in capital cost. However, if a longer trading horizon, say, weekly (allowing us to take into account the low prices on the weekends), is desired, it would be necessary to train a policy using a sufficiently large battery and then using simulation to determine the effective amount of storage needed by the policy. In summary, with today's substantial capital costs, it would be prudent to do an analysis of a policy's storage needs.

Finally, we discuss the issue of choosing $C_{\text{term}}(s)$ in a practical implementation of the algorithm. Because of the daily cycles present in the real-time market, there is likely to be little additional value in expending computational resources toward developing a bidding policy whose horizon lasts much longer than a few days or a week. In fact, from our conversations with industry colleagues, we envision that a bidding policy such as ours has a daily horizon that is used repeatedly day after day, with the policy retrained periodically (perhaps weekly). For such a usage scenario, it is important to correctly choose $C_{\text{term}}(s)$ because leftover energy has value that can be capitalized on even after the true horizon of the policy. We suggest the following practical methods for determining the functional form of $C_{\text{term}}(s)$.

1. Given the knowledge that the same policy is to be reused, in an effort to prevent the forced "sell-off" type behavior that is expected when $C_{\text{term}}(s) = 0$, it is reasonable to choose $C_{\text{term}}(s)$ to structurally resemble $V_0^*(s)$ (i.e., up to constant shifts). One strategy for accomplishing this is to first compute $V_0^*(s)$ using a zero terminal contribution, and then re-solving the dynamic program using the previously computed V_0^* as the terminal contribution. This process can be iterated until the resulting policies (not the value functions themselves) are observed to converge. Our (informal) implementation of this procedure shows that the desired behavior of not forcing the storage to zero at the end of the time horizon is indeed attained.

2. After training an initial policy, we can determine, by inspecting the resource paths, a point in time where the storage level is empty or very low (e.g., immediately after a period of high prices). The horizon of the problem can then be redefined so that T corresponds to this point in time and a new policy (with zero terminal contribution) can be trained. Essentially, we hope that the forced sell-off is translated to a point in time where a natural sell-off would have likely occurred.

7. Conclusion

In this paper, we describe an hour-ahead bidding and battery arbitrage problem for a real-time electricity market (e.g., NYISO's real-time market). We then formulate the problem mathematically as an MDP and show that the optimal value function satisfies a monotonicity property, a structural result that

can be exploited in order to accelerate the convergence of ADP algorithms. The algorithm that we employ is called Monotone-ADP-Bidding and uses monotonicity to infer the value of states nearby an observed state. When benchmarked against a traditional approximate value iteration algorithm, we found that the improvements in terms of solution quality were drastic. Furthermore, the ADP algorithm can reach near-optimal solutions without the need for significant computational time and power (which an exact solution technique like backward dynamic programming certainly requires). In fact, our empirical results show that near-optimal solutions can be generated using less than 10% of the computational resources necessary for backward dynamic programming. We also describe and sketch the proof of convergence for a distribution-free method where we can train value functions with Monotone-ADP-Bidding using historical spot prices—this removes the need for us to perform the difficult task of specifying and fitting an accurate stochastic model of spot prices. In our case study, the method is tested on two large data sets: the five-minute real-time prices from the NYISO from the years of 2011 and 2012. The policies from Monotone-ADP-Bidding help us conclude that energy arbitrage may be most valuable if practiced in a select few, high revenue months. Finally, the ADP policies consistently generated more revenue than several rule-based heuristic strategies that we considered, confirming that an ADP approach that approximates future value is worthwhile.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2015.0640.

References

Barnhart CJ, Dale M, Brandt AR, Benson SM (2013) The energetic implications of curtailing versus storing solar- and wind-generated electricity. *Energy Environ. Sci.* 6(10):2804–2810.

Bellman RE (1957) *Dynamic Programming* (Princeton University Press, Princeton, NJ).

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA).

Breiman L (1992) *Probability* (Society of Industrial and Applied Mathematics, Philadelphia).

Byrne RH, Silva-Monroy CA (2012) Estimating the maximum potential revenue for grid connected electricity storage: Arbitrage and regulation. Technical report SAND2012-3863, Sandia National Laboratories, Albuquerque, NM.

Carmona R, Coulon M (2014) A survey of commodity markets and structural models for electricity prices. Benth FE, Kholodnyi VA, Laurence P, eds. *Quantitative Energy Finance* (Springer, New York), 41–83.

Carmona R, Ludkovski M (2010) Valuation of energy storage: An optimal switching approach. *Quant. Finance* 10(4):359–374.

Cartea Á, Figueroa MG (2005) Pricing in electricity markets: A mean reverting jump diffusion model with seasonality. *Appl. Math. Finance* 12(4):313–335.

Conejo AJ, Nogales FJ, Arroyo JM (2002) Price-taker bidding strategy under price uncertainty. *IEEE Trans. Power Systems* 17(4):1081–1088.

Coulon M, Powell WB, Sircar R (2013) A model for hedging load and price risk in the Texas electricity market. *Energy Econom.* 40:976–988.

David AK (1993) Competitive bidding in electricity supply. *Generation Transm. Distrib. IEE Proc. C* 140(5):421–426.

Eydeland A, Wolyniec K (2003) *Energy and Power Risk Management* (Wiley, Hoboken, NJ).

George AP, Powell WB (2006) Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learn.* 65(1):167–198.

Godfrey GA, Powell WB (2001) An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Sci.* 47(8):1101–1112.

Greenblatt JB, Succar S, Denkenberger DC, Williams RH, Socolow RH (2007) Baseload wind energy: Modeling the competition between gas turbines and compressed air energy storage for supplemental generation. *Energy Policy* 35(3):1474–1492.

Gross G, Finlay D (2000) Generation supply bidding in perfectly competitive electricity markets. *Comput. Math. Organ. Theory* 6(1):83–98.

Harris C (2011) *Electricity Markets: Pricing, Structures and Economics* (John Wiley & Sons, Chichester, UK).

Jiang DR, Powell WB (2015) An approximate dynamic programming algorithm for monotone value functions. Available at http://arxiv.org/abs/1401.1590.

Kim JH, Powell WB (2011) Optimal energy commitments with storage and intermittent supply. *Oper. Res.* 59(6):1347–1360.

Kleywegt AJ, Shapiro A, Homem-de Mello T (2002) The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.* 12(2):479–502.

Lai G, Margot F, Secomandi N (2010) An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation. *Oper. Res.* 58(3):564–582.

Löhndorf N, Minner S (2010) Optimal day-ahead trading and storage of renewable energies: An approximate dynamic programming approach. *Energy Systems* 1(1):61–77.

Löhndorf N, Wozabal D, Minner S (2013) Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Oper. Res.* 61(4):810–823.

Nandalal KDW, Bogardi JJ (2007) *Dynamic Programming Based Operation of Reservoirs: Applicability and Limits* (Cambridge University Press, New York).

Nascimento JM, Powell WB (2009) An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Math. Oper. Res.* 34(1):210–237.

Paatero JV, Lund PD (2005) Effect of energy storage on variations in wind power. *Wind Energy* 8(4):421–441.

Papadaki KP, Powell WB (2003) An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Res. Logist.* 50(7):742–769.

PJM Manual 11: Energy and Ancillary Services Market Operations, http://www.pjm.com/~/media/documents/manuals/m11.ashx, April 2015.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. (Wiley, Hoboken, NJ).

Powell WB, Ruszczyński A, Topaloglu H (2004) Learning algorithms for separable approximations of discrete stochastic optimization problems. *Math. Oper. Res.* 29(4):814–836.

Schwartz ES (1997) The stochastic behavior of commodity prices: Implications for valuation and hedging. *J. Finance* 52(3):923–973.

Secomandi N (2010) Optimal commodity trading with a capacitated storage asset. *Management Sci.* 56(3):449–467.

Shahidehpour M, Yamin H, Li Z (2002) *Market Operations in Electric Power Systems* (John Wiley & Sons, New York).

Sioshansi R (2011) Increasing the value of wind with energy storage. *Energy J.* 32(2):1–29.

Sioshansi R, Denholm P, Jenkin T (2011) A comparative analysis of the value of pure and hybrid electricity storage. *Energy Econom.* 33(1):56–66.

Sioshansi R, Denholm P, Jenkin T, Weiss J (2009) Estimating the value of electricity storage in PJM: Arbitrage and some welfare effects. *Energy Econom.* 31(2):269–277.

Thompson M, Davison M, Rasmussen H (2009) Natural gas storage valuation and optimization: A real options application. *Naval Res. Logist.* 56(3):226–238.

Topaloglu H, Powell WB (2003) An algorithm for approximating piecewise linear concave functions from sample gradients. *Oper. Res. Lett.* 31(1):66–76.

Tsitsiklis JN (1994) Asynchronous stochastic approximation and *Q*-learning. *Machine Learn.* 16(3):185–202.

Walawalkar R, Apt J, Mancini R (2007) Economics of electric energy storage for energy arbitrage and regulation in New York. *Energy Policy* 35(4):2558–2568.

Wen F, David AK (2000) Strategic bidding in competitive electricity markets: A literature survey. *Power Engrg. Soc. Summer Meeting*, Vol. 4 (IEEE, New York), 2168–2173.

Yang Z, Zhang J, Kintner-Meyer MCW, Lu X, Choi D, Lemmon JP, Liu J (2011) Electrochemical energy storage for green grid. *Chemical Rev.* 111(5):3577–613.