Risk-Neutral and Risk-Averse Approximate Dynamic Programming Methods for Bidding in the Energy Market

Daniel R. Jiang
Joint work with Warren B. Powell

May 2016

Dept. of Operations Research and Financial Engineering

PRINCETON UNIVERSITY

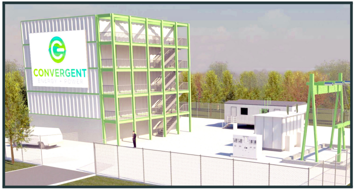
Table of Contents

Hour-Ahead Bidding in the Real-Time Market for Energy Arbitrage

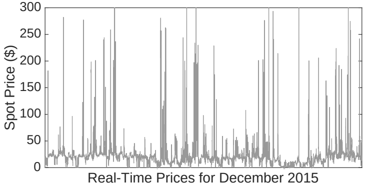
Problem Overview

We consider the problem of using new energy storage technologies to profit off of the real–time electricity market through energy arbitrage[1] (Jiang and Powell 2015c).

- Trade (buy, store, and sell) physical energy to exploit electricity spot prices.
- One of several ways to pay for investments in energy storage on the grid.
- Understanding this problem has implications for the valuation of energy storage.



(a) Multiple 1MW, 6MWh Batteries



(b) Energy Prices

[1]Collaboration with an energy startup in NYC

Related Work

This application can be considered a inventory/storage control problem, similar to the recent work:

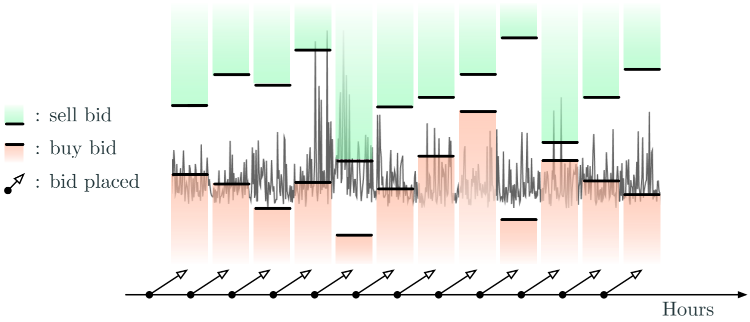
- M. Thompson, M. Davison, and H. Rasmussen (2009). “Natural gas storage valuation and optimization: A real options application”. In: *Naval Research Logistics* 56.3, pp. 226–238
- R. Carmona and M. Ludkovski (2010). “Valuation of energy storage: An optimal switching approach”. In: *Quantitative Finance* 10.4, pp. 359–374
- N. Secomandi (2010). “Optimal commodity trading with a capacitated storage asset”. In: *Management Science* 56.3, pp. 449–467
- G. Lai et al. (2011). “Valuation of storage at a liquefied natural gas terminal”. In: *Operations Research* 59.3, pp. 602–616
- J. H. Kim and W. B. Powell (2011). “Optimal energy commitments with storage and intermittent supply”. In: *Operations Research* 59.6, pp. 1347–1360
- N. Löhndorf, D. Wozabal, and S. Minner (2013). “Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming”. In: *Operations Research* 61.4, pp. 810–823

In our case, there is the additional complication that the ability to interact with the market is uncertain.

There are inter–hour and intra–hour components in our problem.

Inter–Hour Behavior: At hour t, we place the bid b_t into the market.

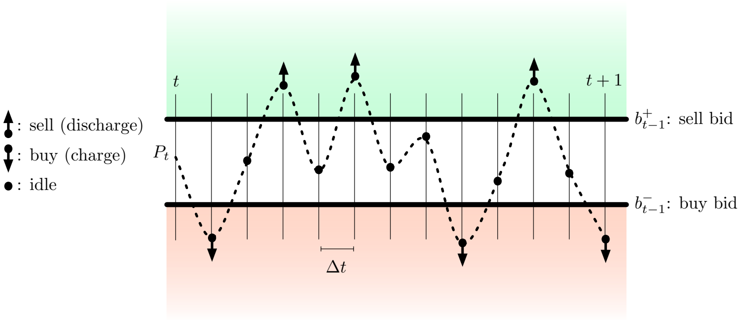
- A bid $b_t = (b_t^-, b_t^+)$ is a pair of prices consisting buy bid b_t^- and a sell bid b_t^+.
- It is called an hour–ahead bid because it is active on the interval $(t+1, t+2]$.
- b_t is fixed for the whole hour $(t+1, t+2]$ even though M settlements (transactions) occur within the hour.



Intra–Hour Behavior: Within $(t, t+1]$, the spot price P_t fluctuates every $\Delta t = 5$ min. When the spot price P_t moves

- below the buy bid b^-_{t-1}, we are obligated to buy or charge from the market;
- above the sell bid b^+_{t-1}, we are obligated to sell or discharge to the market;
- otherwise, we are "out of the market" and remain idle.

Transactions in both directions occur at the spot price P_t.



State Variable: $S_t = (R_t, L_t, P_t, b_{t-1}) \in \mathcal{S}$.

- R_t is the resource state taking values between 0 and R_{max},
- L_t is the number of trades left (to consider loss of storage efficiency),
- P_t is the spot price,
- b_{t-1} is the previous bid (needed for transitioning from $t \to t+1$).

Decision: $b_t = (b_t^-, b_t^+)$, the bid that is active during $(t+1, t+2]$.

- $b_t \in \mathcal{B} \subseteq \{(b^-, b^+) : 0 \le b^- \le b^+\}$,
- A bidding policy is $\{X_0^\pi, X_1^\pi, \ldots, X_{T-1}^\pi\}$ where $X_t^\pi : \mathcal{S} \to \mathcal{B}$ (π indexes the policy).

Transitions: $R_{t+1} \approx R_t + \sum_{m=1}^{M} \left[\mathbf{1}_{\{b_t^- > P_m\}} - \mathbf{1}_{\{b_t^+ < P_m\}} \right]$

- For each settlement outcome, we add either 1, -1, or 0 to the previous 5-minute resource state.
- Similar transition for L_t.

State Variable: $S_t = (R_t, L_t, P_t, b_{t-1}) \in \mathcal{S}$.

- R_t is the resource state taking values between 0 and $R_{\max}$,
- L_t is the number of trades left (to consider loss of storage efficiency),
- P_t is the spot price,
- b_{t-1} is the previous bid (needed for transitioning from $t \to t+1$).

Decision: $b_t = (b_t^-, b_t^+)$, the bid that is active during $(t+1, t+2]$.

- $b_t \in \mathcal{B} \subseteq \{(b^-, b^+) : 0 \leq b^- \leq b^+\}$,
- A bidding policy is $\{X_0^\pi, X_1^\pi, \ldots, X_{T-1}^\pi\}$ where $X_t^\pi : \mathcal{S} \to \mathcal{B}$ (π indexes the policy).

Transitions: $R_{t+1} \approx R_t + \sum_{m=1}^{M} \left[\mathbf{1}_{\{b_t^- > P_m\}} - \mathbf{1}_{\{b_t^+ < P_m\}} \right]$

- For each settlement outcome, we add either 1, -1, or 0 to the previous 5-minute resource state.
- Similar transition for L_t.

State Variable: $S_t = (R_t, L_t, P_t, b_{t-1}) \in \mathcal{S}$.

- R_t is the resource state taking values between 0 and R_{max},
- L_t is the number of trades left (to consider loss of storage efficiency),
- P_t is the spot price,
- b_{t-1} is the previous bid (needed for transitioning from $t \to t+1$).

Decision: $b_t = (b_t^-, b_t^+)$, the bid that is active during $(t+1, t+2]$.

- $b_t \in \mathcal{B} \subseteq \{(b^-, b^+) : 0 \leq b^- \leq b^+\}$,
- A bidding policy is $\{X_0^\pi, X_1^\pi, \ldots, X_{T-1}^\pi\}$ where $X_t^\pi : \mathcal{S} \to \mathcal{B}$ (π indexes the policy).

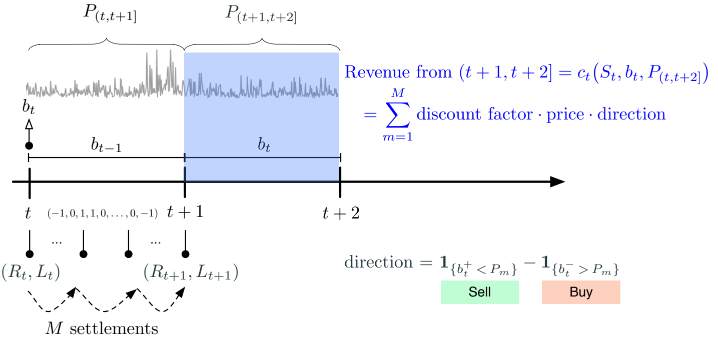
Transitions: $R_{t+1} \approx R_t + \sum_{m=1}^{M} \left[\mathbf{1}_{\{b_t^- > P_m\}} - \mathbf{1}_{\{b_t^+ < P_m\}} \right]$

- For each settlement outcome, we add either 1, -1, or 0 to the previous 5-minute resource state.
- Similar transition for L_t.

Contribution Function: $c_t(S_t, b_t, P_{(t,t+2]})$ is the (random at time t) revenue made in time interval $(t+1, t+2]$.

- $C^{\pi}_{t+2} = c_t(S^{\pi}_t, X^{\pi}_t(S^{\pi}_t), P_{(t,t+2]})$ is the revenue at time t using policy π.

Timeline of Notation:



The Risk-Neutral Case

Objective Function: Let Π be the set of all admissible policies.

$$\max_{\pi \in \Pi} \mathbf{E}\left[\sum_{t=1}^{T} C_t^{\pi}\right]$$

Bellman Recursion: For $s \in \mathcal{S}$, the optimal value function V^* is given by

$$V_t^*(s) = \max_{b_t \in \mathcal{B}} \mathbf{E}\Big[c_t\big(S_t, b_t, P_{(t,t+2]}\big) + V_{t+1}^*(S_{t+1}) \,|\, S_t = s\Big] \text{ for } t < T,$$

$$V_T^*(s) = 0.$$

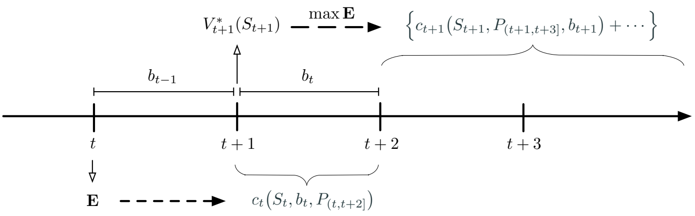


Figure 2: “Two Steps Ahead”

The following reasons make this dynamic program expensive to compute.

- Lack of convexity in the value function (optimization at each stage nonconvex as well). Popular methods, such as stochastic dual dynamic programming for convex problems (Pereira and Pinto, 1991), are not applicable.
- Large state space, due to the fact that $b_{t-1} = (b_{t-1}^{-}, b_{t-1}^{+})$ needs to be finely discretized. This also leads to a large action space.
- If the support of P_t is finite, the $\mathbf{E}$ over $P_{(t,t+2]}$ is computable but computationally challenging, even for $M = 1$.

Even for a simple versions of the bidding problem, naive dynamic programming takes over a week to solve. What can we do to speed this up?

Intractable Dynamic Program

The following reasons make this dynamic program expensive to compute.

- Lack of convexity in the value function (optimization at each stage nonconvex as well). Popular methods, such as stochastic dual dynamic programming for convex problems (Pereira and Pinto, 1991), are not applicable.
- Large state space, due to the fact that $b_{t-1} = (b_{t-1}^-, b_{t-1}^+)$ needs to be finely discretized. This also leads to a large action space.
- If the support of P_t is finite, the $\mathbf{E}$ over $P_{(t,t+2]}$ is computable but computationally challenging, even for $M = 1$.

Even for a simple versions of the bidding problem, naive dynamic programming takes over a week to solve. What can we do to speed this up?

The following reasons make this dynamic program expensive to compute.

- Lack of convexity in the value function (optimization at each stage nonconvex as well). Popular methods, such as stochastic dual dynamic programming for convex problems (Pereira and Pinto, 1991), are not applicable.
- Large state space, due to the fact that $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$ needs to be finely discretized. This also leads to a large action space.
- If the support of P_t is finite, the $\mathbf{E}$ over $P_{(t,t+2]}$ is computable but computationally challenging, even for $M = 1$.

Even for a simple versions of the bidding problem, naive dynamic programming takes over a week to solve. What can we do to speed this up?

Intractable Dynamic Program

The following reasons make this dynamic program expensive to compute.

- Lack of convexity in the value function (optimization at each stage nonconvex as well). Popular methods, such as stochastic dual dynamic programming for convex problems (Pereira and Pinto, 1991), are not applicable.
- Large state space, due to the fact that $b_{t-1} = (b_{t-1}^{-}, b_{t-1}^{+})$ needs to be finely discretized. This also leads to a large action space.
- If the support of P_t is finite, the $\mathbf{E}$ over $P_{(t,t+2]}$ is computable but computationally challenging, even for $M = 1$.

Even for a simple versions of the bidding problem, naive dynamic programming takes over a week to solve. What can we do to speed this up?

The following reasons make this dynamic program expensive to compute.

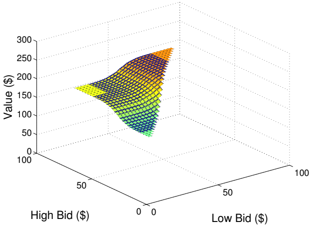
- Lack of convexity in the value function (optimization at each stage nonconvex as well). Popular methods, such as stochastic dual dynamic programming for convex problems (Pereira and Pinto, 1991), are not applicable.
- Large state space, due to the fact that $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$ needs to be finely discretized. This also leads to a large action space.
- If the support of P_t is finite, the $\mathbf{E}$ over $P_{(t,t+2]}$ is computable but computationally challenging, even for $M = 1$.

Even for a simple versions of the bidding problem, naive dynamic programming takes over a week to solve. What can we do to speed this up?

The following property is the motivation behind our ADP (approximate dynamic programming) algorithm, *Monotone–ADP*.

Proposition

The optimal value functions $V_t^*(R_t, L_t, P_t, b_{t-1}^-, b_{t-1}^+)$ *are nondecreasing in* R_t, L_t, b_{t-1}^-, *and* b_{t-1}^+. *In other words, there exists a partial order* $\preceq$ *on the state space* $\mathcal{S}$.



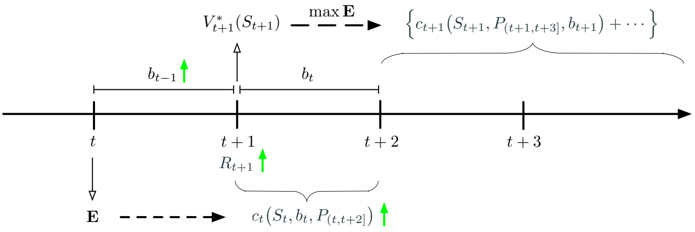
(a) V_t^* at $t = 12$ for $R_t = 3$ (b) V_t^* at $t = 12$ and $R_t \in \{0, \ldots, 6\}$

Figure 3: Illustration of Monotonicity in b_{t-1}^-, b_{t-1}^+, and R_t (computed using BDP)

Intuition for Why More is Better

Clearly, higher values of R_t (more energy to sell) are preferable to lower values. Why is it monotone in the previous bid, $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$?

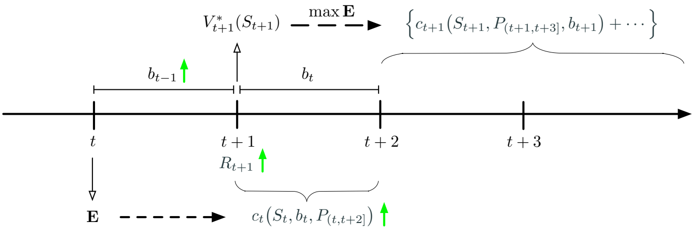
- Let P be a sample from $P_{(t,t+2]}$ and consider the point of view starting at time t.
- Increasing $b^-_{t-1} \Rightarrow$ "easier to buy." Increasing $b^+_{t-1} \Rightarrow$ "more difficult to sell."
- $R_{t+1}(P, b_{t-1})$ is increasing in b_{t-1}.
- The revenue during the period $(t, t+1]$ is not included in $c_t(S_t, b_t, P)$.
- Therefore, $c_t(S_t, b_t, P)$ is increasing in b_{t-1}.



Intuition for Why More is Better

Clearly, higher values of R_t (more energy to sell) are preferable to lower values. Why is it monotone in the previous bid, $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$?

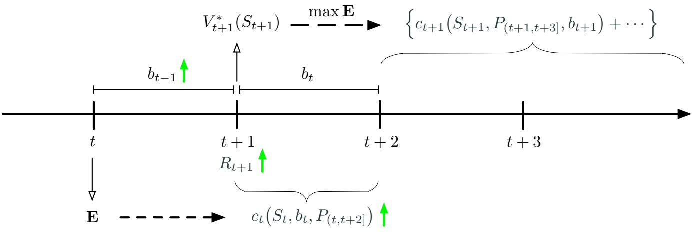
- Let P be a sample from $P_{(t,t+2]}$ and consider the point of view starting at time t.
- Increasing $b^-_{t-1} \Rightarrow$ "easier to buy." Increasing $b^+_{t-1} \Rightarrow$ "more difficult to sell."
- $R_{t+1}(P, b_{t-1})$ is increasing in b_{t-1}.
- The revenue during the period $(t, t+1]$ is not included in $c_t(S_t, b_t, P)$.
- Therefore, $c_t(S_t, b_t, P)$ is increasing in b_{t-1}.



Intuition for Why More is Better

Clearly, higher values of R_t (more energy to sell) are preferable to lower values. Why is it monotone in the previous bid, $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$?

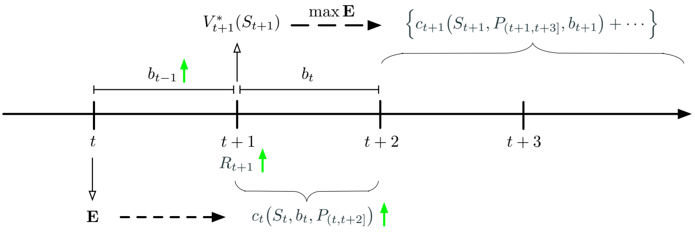
- Let P be a sample from $P_{(t,t+2]}$ and consider the point of view starting at time t.
- Increasing $b^-_{t-1} \Rightarrow$ "easier to buy." Increasing $b^+_{t-1} \Rightarrow$ "more difficult to sell."
- $R_{t+1}(P, b_{t-1})$ is increasing in b_{t-1}.
- The revenue during the period $(t, t+1]$ is not included in $c_t(S_t, b_t, P)$.
- Therefore, $c_t(S_t, b_t, P)$ is increasing in b_{t-1}.



Intuition for Why More is Better

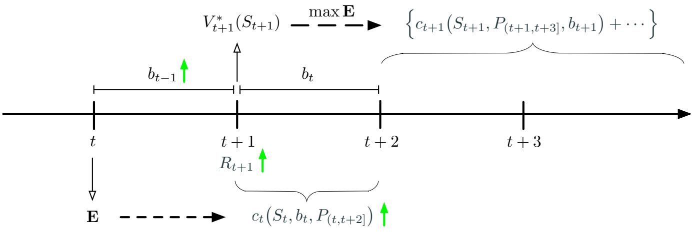
Clearly, higher values of R_t (more energy to sell) are preferable to lower values. Why is it monotone in the previous bid, $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$?

- Let P be a sample from $P_{(t,t+2]}$ and consider the point of view starting at time t.
- Increasing $b^-_{t-1} \Rightarrow$ "easier to buy." Increasing $b^+_{t-1} \Rightarrow$ "more difficult to sell."
- $R_{t+1}(P, b_{t-1})$ is increasing in b_{t-1}.
- The revenue during the period $(t, t+1]$ is not included in $c_t(S_t, b_t, P)$.
- Therefore, $c_t(S_t, b_t, P)$ is increasing in b_{t-1}.



Clearly, higher values of R_t (more energy to sell) are preferable to lower values. Why is it monotone in the previous bid, $b_{t-1} = (b^-_{t-1}, b^+_{t-1})$?

- Let P be a sample from $P_{(t,t+2]}$ and consider the point of view starting at time t.
- Increasing $b^-_{t-1} \Rightarrow$ "easier to buy." Increasing $b^+_{t-1} \Rightarrow$ "more difficult to sell."
- $R_{t+1}(P, b_{t-1})$ is increasing in b_{t-1}.
- The revenue during the period $(t, t+1]$ is not included in $c_t(S_t, b_t, P)$.
- Therefore, $c_t(S_t, b_t, P)$ is increasing in b_{t-1}.



Monotone–ADP Algorithm

Goal: Design a data–driven method that updates an approximate policy recursively (e.g., as new prices are observed on the market) by taking advantage of the monotone structure of the value function.

Overview of Monotone–ADP

Step 1. Set $n = 1$.

Step 2. For $t = 0, 1, 2, \ldots, T - 1$ do:

- **Step 2a.** **Visit** a state S_t^n.
- **Step 2b.** **Sample/observe** new spot price data.
- **Step 2c.** **Compute** a noisy, biased observation of $V_t^*(S_t^n)$.
- **Step 2d.** **Update** the approximate value function.
- **Step 2e.** **Project** to (some) space of monotone functions.

Step 3. If $n < N$ (stopping iteration), increment n and return to **Step 2**.

Monotone–ADP employs an adaptive projection step, where the (monotone) space onto which we project changes at every iteration.

- Let $\overline{V}_t^n \in \mathbb{R}^d$ be the value function approximation to the optimal value function $V_t^* \in \mathbb{R}^d$ in iteration n.
- Let $z_t^n(S_t^n)$ be the observed value of $V_t^*(S_t^n)$.
- For $s \in \mathcal{S}$ and $v \in \mathbb{R}$, let us define the following set of monotone value functions:

$$\mathcal{V}_{\mathcal{M}}(s, z) = \left\{ V \in \mathbb{R}^d : V(s) = z,\ V(s_1) \leq V(s_2)\ \forall s_1, s_2 \in \mathcal{S} \text{ where } s_1 \preceq s_2 \right\}$$

which fixes the value at s to be z, while restricting to the set of monotone V.

Adaptive Projection Step

$$\overline{V}_t^n \in \arg\min \left\{ \left\| V_t - \overline{V}_t^{n-1} \right\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\left(S_t^n, z_t^n(S_t^n)\right) \right\}.$$

Monotone–ADP employs an adaptive projection step, where the (monotone) space onto which we project changes at every iteration.

- Let $\overline{V}_t^n \in \mathbb{R}^d$ be the value function approximation to the optimal value function $V_t^* \in \mathbb{R}^d$ in iteration n.
- Let $z_t^n(S_t^n)$ be the observed value of $V_t^*(S_t^n)$.
- For $s \in \mathcal{S}$ and $v \in \mathbb{R}$, let us define the following set of monotone value functions:

$$\mathcal{V}_{\mathcal{M}}(s, z) = \left\{ V \in \mathbb{R}^d : V(s) = z, \; V(s_1) \leq V(s_2) \; \forall s_1, s_2 \in \mathcal{S} \text{ where } s_1 \preceq s_2 \right\}$$

which fixes the value at s to be z, while restricting to the set of monotone V.

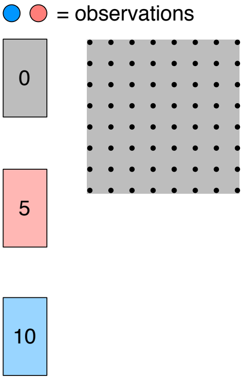
Adaptive Projection Step

$$\overline{V}_t^n \in \arg\min \left\{ \left\| V_t - \overline{V}_t^{n-1} \right\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\left(S_t^n, z_t^n(S_t^n)\right) \right\}.$$

Proposition

The solution to the minimization can be characterized using an operator Π_M.

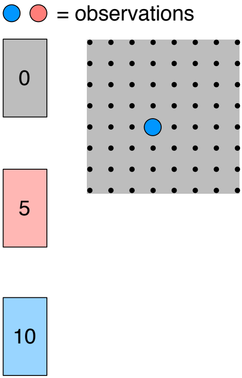
$$\Pi_M\big(S_t^n, z_t^n(S_t^n), \overline{V}_t^{n-1}\big) \in \arg\min\Big\{\big\| V_t - \overline{V}_t^{n-1}\big\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\big(S_t^n, z_t^n(S_t^n)\big)\Big\}.$$



Proposition

The solution to the minimization can be characterized using an operator Π_M.

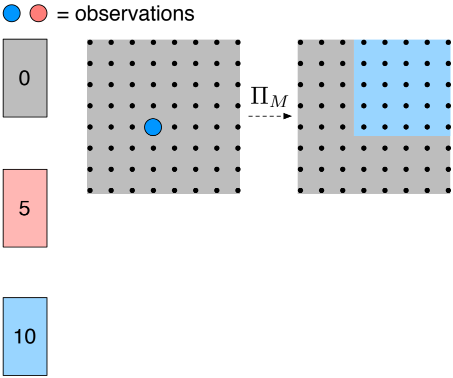
$$\Pi_M\big(S_t^n, z_t^n(S_t^n), \overline{V}_t^{n-1}\big) \in \arg\min\Big\{\big\| V_t - \overline{V}_t^{n-1}\big\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\big(S_t^n, z_t^n(S_t^n)\big)\Big\}.$$



Proposition

The solution to the minimization can be characterized using an operator Π_M.

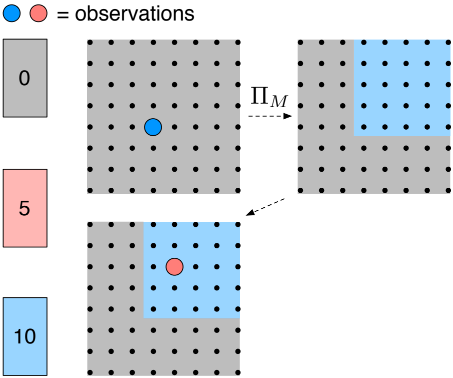
$$\Pi_M\left(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1}\right) \in \arg\min\left\{\left\| V_t - \bar{V}_t^{n-1}\right\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\left(S_t^n, z_t^n(S_t^n)\right)\right\}.$$



Proposition

The solution to the minimization can be characterized using an operator Π_M.

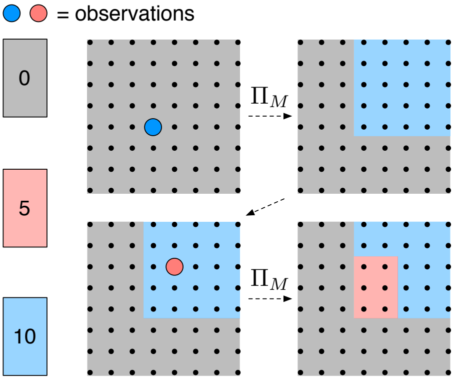
$$\Pi_M\big(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1}\big) \in \arg\min\Big\{\big\| V_t - \bar{V}_t^{n-1}\big\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\big(S_t^n, z_t^n(S_t^n)\big)\Big\}.$$



Proposition

The solution to the minimization can be characterized using an operator Π_M.

$$\Pi_M\left(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1}\right) \in \arg\min\left\{\left\| V_t - \bar{V}_t^{n-1}\right\|_2 : V_t \in \mathcal{V}_{\mathcal{M}}\left(S_t^n, z_t^n(S_t^n)\right)\right\}.$$



For $s^r \in \mathcal{S}$ and $z^r \in \mathbb{R}$, let (s^r, z^r) be a reference point to which other states are compared. Let $V_t \in \mathbb{R}^d$ and define the projection operator $\Pi_M : \mathcal{S} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, where the component of the vector $\Pi_M(s^r, z^r, V_t)$ at s is given by

$$\Pi_M\big(s^r, z^r, V_t\big)(s) = \begin{cases} z^r & \text{if } s = s^r, \\ z^r \vee V_t(s) & \text{if } s^r \preceq s,\ \ s \neq s^r, \\ z^r \wedge V_t(s) & \text{if } s^r \succeq s,\ \ s \neq s^r, \\ V_t(s) & \text{otherwise.} \end{cases}$$

Dynamic Programming Operator:

$$\left(H\overline{V}\right)_t(s) = \max_{b_t \in \mathcal{B}} \mathbf{E}\left[c_t\left(S_t, b_t, P_{(t,t+2]}\right) + \overline{V}_{t+1}(S_{t+1}) \,|\, S_t = s\right]$$

Algorithm Description:

Step 0a. Initialize $\overline{V}_t^0 \in [0, V_{\text{max}}]$ for each t.

Step 0b. Set $\overline{V}_T^n(s) = 0$ for each $s \in \mathcal{S}$ and $n \leq N$.

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:

  $$\hat{v}_t^n = \left(H\overline{V}^{n-1}\right)_t + w_t^n.$$

- **Step 2b.** Smooth in the new observation with previous value:

  $$z_t^n(s) = \left(1 - \alpha_t^n(s)\right) \overline{V}_t^{n-1}(s) + \alpha_t^n(s)\, \hat{v}_t^n(s).$$

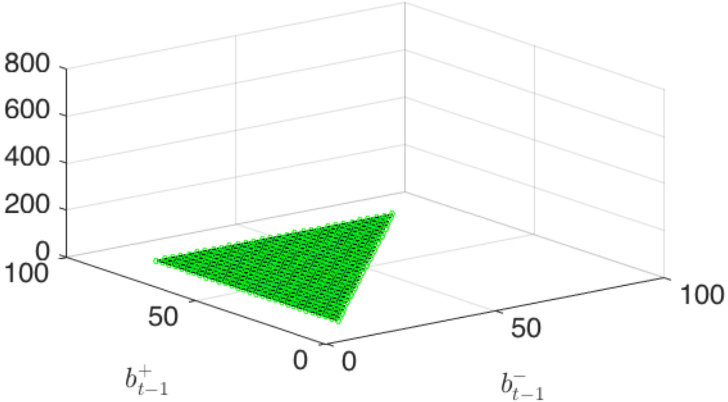
- **Step 2c.** Perform monotonicity projection operator:

  $$\overline{V}_t^n = \Pi_M\left(S_t^n, z_t^n(S_t^n), \overline{V}_t^{n-1}\right).$$

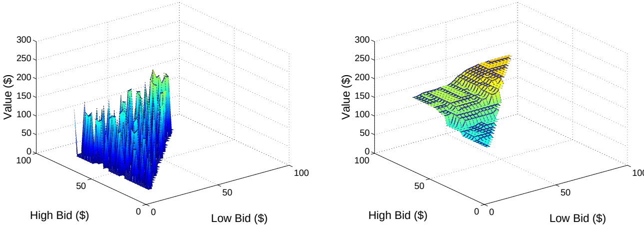
- **Step 2d.** Choose the next state S_{t+1}^n given $\mathcal{F}^{n-1}$.

Step 3. If $n < N$, increment n and return **Step 1.**

Here's how it works in practice.



800
600
400
200
0
100
50
0
0
50
100
b^+_{t-1}
b^-_{t-1}



(a) Naive ADP, $N = 1000$ (b) Monotone–ADP, $N = 1000$

Figure 4: Visual Comparison of Value Function Approximations (other dimensions fixed)

Dynamic Programming Operator:

$$\left(H\overline{V}\right)_t(s) = \max_{b \in \mathcal{B}} \mathbf{E}\Big[c_t\big(s, b, P_{(t,t+2]}\big) + \overline{V}_{t+1}(S_{t+1}) \,|\, S_t = s\Big]$$

But what if we are in a setting where the $\mathbf{E}$ cannot be computed (e.g., we may only have data, but no distribution)?

Another Dynamic Programming Operator[2] (State–Action Value Function $\overline{Q}_t$):

$$\left(H\overline{Q}\right)_t(s, b) = \mathbf{E}\underbrace{\Big[c_t\big(s, b, P_{(t,t+2]}\big) + \max_{b_{t+1} \in \mathcal{B}} \overline{Q}_{t+1}(S_{t+1}, b_{t+1}) \,|\, S_t = s\Big]}_{\text{This is an unbiased sample of } (H\overline{Q})_t \text{ w.r.t. to } \overline{Q}_{t+1}\text{!}}$$

[2] See, e.g., J. N. Tsitsiklis (1994). "Asynchronous stochastic approximation and Q-learning". In: *Machine Learning* 16.3, pp. 185–202

Some Assumptions

The following assumptions are necessary for the analysis of Monotone–ADP on a finite state MDP.

A1. $\sum_{n=1}^{\infty} \mathbf{P}(S_t^n = s \,|\, \mathcal{F}^{n-1}) = \infty \quad a.s.$

A2. The contribution at each time period is integrable.

A3. The noise sequence w_t^n satisfies: $\mathbf{E}\left[w_t^{n+1}(s) \,|\, \mathcal{F}^n\right] = 0.$

A4. For each $t \leq T$ and state s, suppose $\alpha_t^n \in [0, 1]$ is $\mathcal{F}^n$–measurable and

1. $\sum_{n=0}^{\infty} \alpha_t^n(s) = \infty \quad a.s.,$
2. $\sum_{n=0}^{\infty} \alpha_t^n(s)^2 < \infty \quad a.s.$

Theorem ([Jiang and Powell, 2015])

Under some technical assumptions (e.g., exploration, unbiased noise conditional on $\mathcal{F}^{n-1}$, bounded observations, a step-size condition), for each $t \leq T$ and $s \in \mathcal{S}$,

$$\bar{V}_t^n(s) \longrightarrow V_t^*(s) \quad a.s.$$

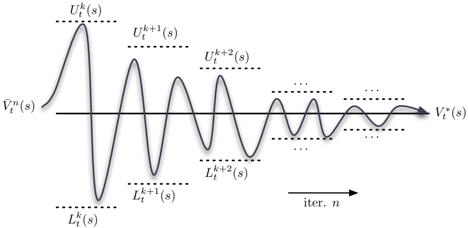
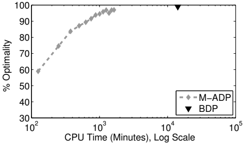


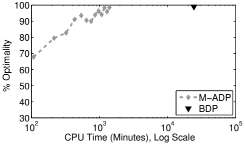
Figure 5: Illustration of Proof Technique

On a test suite of 6 bidding problems with varying parameters, where the optimal policy can be computed:

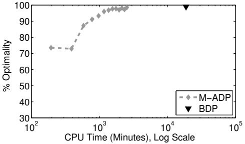
- Monotone–ADP achieves near–optimal (90%–96%) results,
- Uses up to an order of magnitude less computation than dynamic programming.



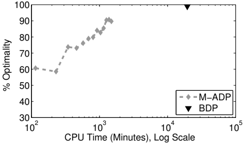
(a) Problem A, **7%**



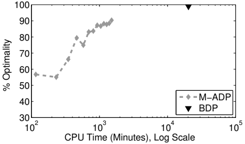
(b) Problem B, **4%**



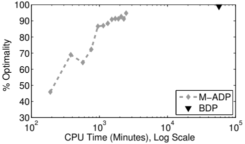
(c) Problem C, **6%**



(d) Problem D, **6%**



(e) Problem E, **7%**

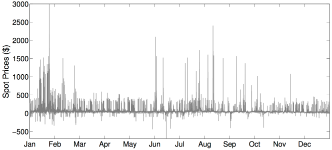


(f) Problem F, **4%**

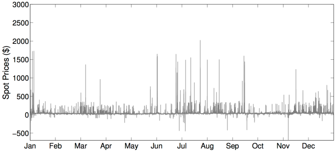
Figure 6: Computation Times of M–ADP vs. DP

In our case study, we run the data-driven version of Monotone–ADP and compare it to a bidding policy used in industry (given to us by an energy startup).

- No benchmark. No known distributions.
- Model contains (3.6 million states per time period) · (24 time periods) = **86.4 million states**.



(a) 2011 Real–Time Prices

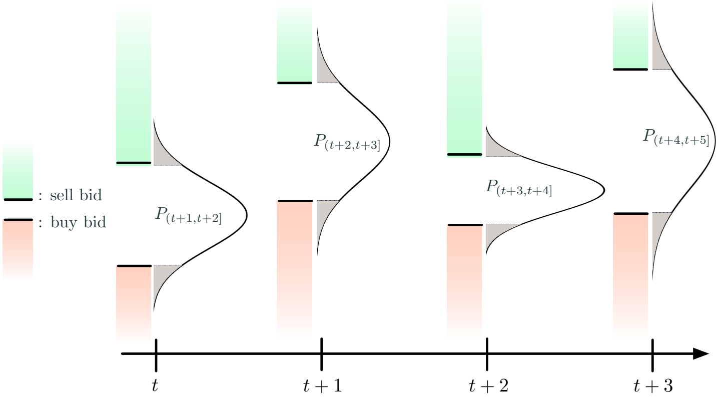


(b) 2012 Real–Time Prices

Figure 7: NYISO Real–Time, 5–Minute Prices Used for Training and Testing of an ADP Policy

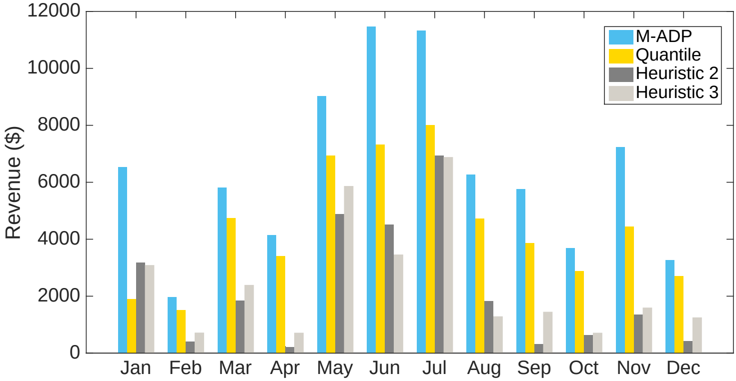
Analyze historical data to obtain an empirical distribution for prices in $(t+1, t+2]$.
Place buy bid at the α^- quantile and sell bid at the α^+ quantile.

- Idea is to emphasize high value trades.
- After tuning, $(\alpha^-, \alpha^+) \approx (0.1, 0.9)$.
- With some additional logic to deal with capacity of storage, this quantile method is the best performing heuristic policy.



Comparison to Monotone–ADP Bidding Policy

Policies were trained using data from 2011 and tested on data from 2012. Trained on the same data, the monotone policy produces significantly more value.



A brief aside...

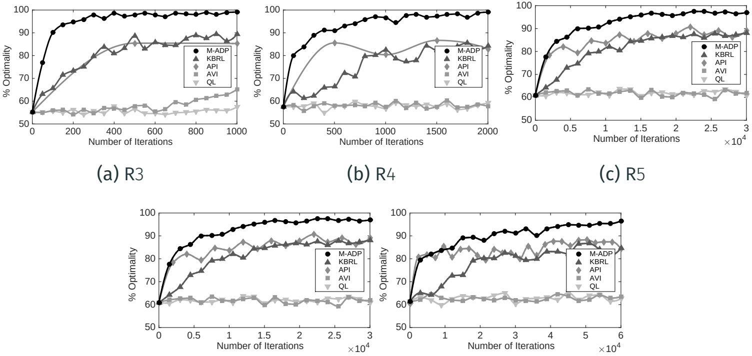
Any problem where "more is better" can potentially benefit from Monotone–ADP.

- optimal stopping or optimal replacement* (Rust 1987),
- dynamic pricing in revenue management (Gallego and van Ryzin 1994),
- glycemic control for diabetes patients* (Hsih 2010),
- allocating energy between renewables, demand, and storage* (Salas and Powell 2013),
- consumption behavior in economics (Kaplan and Violante 2014).

*See the following paper for numerical work on these problems.
If monotonicity exists, then it is beneficial to exploit it.

D. R. Jiang and W. B. Powell (2015a). "An approximate dynamic programming algorithm for monotone value functions". In: *Operations Research* 63.6, pp. 1489–1511

Benchmarking results of Monotone–ADP on an optimal stopping problem ranging from 3-7 dimensions with up to 487 million states.



(a) R3 (b) R4 (c) R5 (d) R6 (e) R7

Figure 8: Empirical Convergence Rates of M–ADP vs. Other ADP Algorithms

Back to the bidding problem...

The Risk-Averse Case

What if the energy in storage could not be solely dedicated to energy arbitrage?

- The energy in storage has other sources of demand, e.g., backup, which are often higher priority.
- There is the risk of a shortage penalty if storage level is too low to satisfy higher priority demands — a "stockout event."

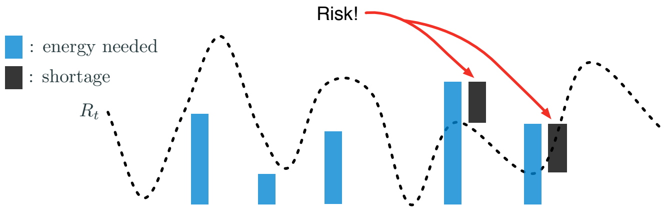


Figure 9: Illustration of Shared Storage

Let's review the idea of using dynamic risk measures in the context of MDPs[3].

Model Preliminaries ("Costs" — Smaller is Better)

- We consider a finite time–horizon, $t = 0, 1, 2, \ldots, T$, where the last decision is made at time $t = T - 1$.
- Our information process is a discrete–time stochastic process $(W_t)_{t=0}^{T}$, where W_t is adapted to $\{\mathcal{F}_t\}_{t=0}^{T}$. Includes both prices and random demands.
- The state variable is $S_t \in \mathcal{S}$ and the action is $a_t \in \mathcal{A}$ (finite state/action spaces).
- Let $\mathcal{Z}_t$ denote the space of $\mathcal{F}_t$–measurable random variables and $\mathcal{Z}_{t,T} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$.
- For a policy $\pi \in \Pi$, let the sequence of costs be represented by the process C_t^π for $t = 1, 2, \ldots, T$, where $C_t^\pi = c_{t-1}(S_{t-1}^\pi, A_{t-1}^\pi(S_{t-1}^\pi), W_t^\pi) \in \mathcal{Z}_t$.

[3] A. Ruszczynski (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261

Let's review the idea of using dynamic risk measures in the context of MDPs[3].

Model Preliminaries ("Costs" — Smaller is Better)

- We consider a finite time–horizon, $t = 0, 1, 2, \ldots, T$, where the last decision is made at time $t = T - 1$.
- Our information process is a discrete–time stochastic process $(W_t)_{t=0}^T$, where W_t is adapted to $\{\mathcal{F}_t\}_{t=0}^T$. Includes both prices and random demands.
- The state variable is $S_t \in \mathcal{S}$ and the action is $a_t \in \mathcal{A}$ (finite state/action spaces).
- Let $\mathcal{Z}_t$ denote the space of $\mathcal{F}_t$–measurable random variables and $\mathcal{Z}_{t,T} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$.
- For a policy $\pi \in \Pi$, let the sequence of costs be represented by the process C_t^π for $t = 1, 2, \ldots, T$, where $C_t^\pi = c_{t-1}(S_{t-1}^\pi, A_{t-1}^\pi(S_{t-1}^\pi), W_t^\pi) \in \mathcal{Z}_t$.

[3] A. Ruszczynski (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261

Let's review the idea of using dynamic risk measures in the context of MDPs[3].

Model Preliminaries ("Costs" — Smaller is Better)

- We consider a finite time–horizon, $t = 0, 1, 2, \ldots, T$, where the last decision is made at time $t = T - 1$.
- Our information process is a discrete–time stochastic process $(W_t)_{t=0}^{T}$, where W_t is adapted to $\{\mathcal{F}_t\}_{t=0}^{T}$. Includes both prices and random demands.
- The state variable is $S_t \in \mathcal{S}$ and the action is $a_t \in \mathcal{A}$ (finite state/action spaces).
- Let $\mathcal{Z}_t$ denote the space of $\mathcal{F}_t$–measurable random variables and $\mathcal{Z}_{t,T} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$.
- For a policy $\pi \in \Pi$, let the sequence of costs be represented by the process C_t^π for $t = 1, 2, \ldots, T$, where $C_t^\pi = c_{t-1}(S_{t-1}^\pi, A_{t-1}^\pi(S_{t-1}^\pi), W_t^\pi) \in \mathcal{Z}_t$.

[3] A. Ruszczynski (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261

Let's review the idea of using dynamic risk measures in the context of MDPs[3].

Model Preliminaries ("Costs" — Smaller is Better)

- We consider a finite time–horizon, $t = 0, 1, 2, \ldots, T$, where the last decision is made at time $t = T - 1$.
- Our information process is a discrete–time stochastic process $(W_t)_{t=0}^{T}$, where W_t is adapted to $\{\mathcal{F}_t\}_{t=0}^{T}$. Includes both prices and random demands.
- The state variable is $S_t \in \mathcal{S}$ and the action is $a_t \in \mathcal{A}$ (finite state/action spaces).
- Let $\mathcal{Z}_t$ denote the space of $\mathcal{F}_t$–measurable random variables and $\mathcal{Z}_{t,T} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$.
- For a policy $\pi \in \Pi$, let the sequence of costs be represented by the process C_t^π for $t = 1, 2, \ldots, T$, where $C_t^\pi = c_{t-1}(S_{t-1}^\pi, A_{t-1}^\pi(S_{t-1}^\pi), W_t^\pi) \in \mathcal{Z}_t$.

[3] A. Ruszczynski (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261

Let's review the idea of using dynamic risk measures in the context of MDPs[3].

Model Preliminaries ("Costs" — Smaller is Better)

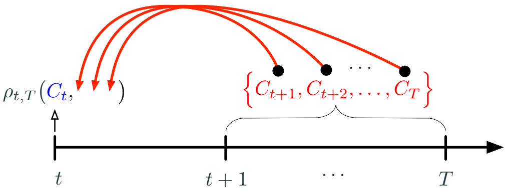
- We consider a finite time-horizon, $t = 0, 1, 2, \ldots, T$, where the last decision is made at time $t = T - 1$.
- Our information process is a discrete-time stochastic process $(W_t)_{t=0}^{T}$, where W_t is adapted to $\{\mathcal{F}_t\}_{t=0}^{T}$. Includes both prices and random demands.
- The state variable is $S_t \in \mathcal{S}$ and the action is $a_t \in \mathcal{A}$ (finite state/action spaces).
- Let $\mathcal{Z}_t$ denote the space of $\mathcal{F}_t$-measurable random variables and $\mathcal{Z}_{t,T} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$.
- For a policy $\pi \in \Pi$, let the sequence of costs be represented by the process C_t^π for $t = 1, 2, \ldots, T$, where $C_t^\pi = c_{t-1}(S_{t-1}^\pi, A_{t-1}^\pi(S_{t-1}^\pi), W_t^\pi) \in \mathcal{Z}_t$.

[3] A. Ruszczynski (2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261

Definition

A conditional risk measure[4] $\rho_{t,T} : \mathcal{Z}_{t,T} \to \mathcal{Z}_t$ is a monotone mapping that takes a sequence of future costs $C_t, \dots, C_T$ to an amount $\rho_{t,T}(C_t, \dots, C_T) \in \mathcal{Z}_t$.

Intuition: related to the idea of a certainty equivalent cost (i.e., one is indifferent between incurring $\rho_{t,T}(C_t, \dots, C_T)$ versus the stream of stochastic future costs).



Definition

A dynamic risk measure $\{\rho_{t,T}\}_{t=0}^{T}$ is a sequence of conditional risk measures, which allows us to evaluate the future risk at any time t using $\rho_{t,T}$.

[4] A. Ruszczynski and A. Shapiro (2006). "Conditional risk mappings". In: *Mathematics of Operations Research* 31.3, pp. 544–561

In the risk-neutral case, our objective was

$$\min_{\pi\in\Pi} \mathbf{E}\left[\sum_{t=1}^{T} C_t^\pi\right] = \min_{\pi\in\Pi} \mathbf{E}_0\Big(C_1^\pi + \mathbf{E}_1\big(C_2^\pi + \cdots + \mathbf{E}_{T-1}(C_T^\pi)\cdots\big)\Big).$$

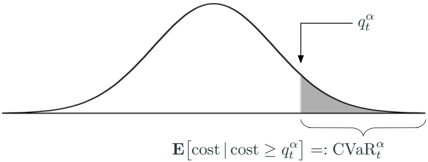
By the tower property, this means we are using

$$\rho_{t,T}\big(C_{t+1}^\pi, C_{t+2}^\pi, \ldots, C_T^\pi\big) = \mathbf{E}_t\big(C_{t+1}^\pi + C_{t+2}^\pi + \cdots + C_T^\pi\big).$$

Attempt at Risk-Averse Formulation

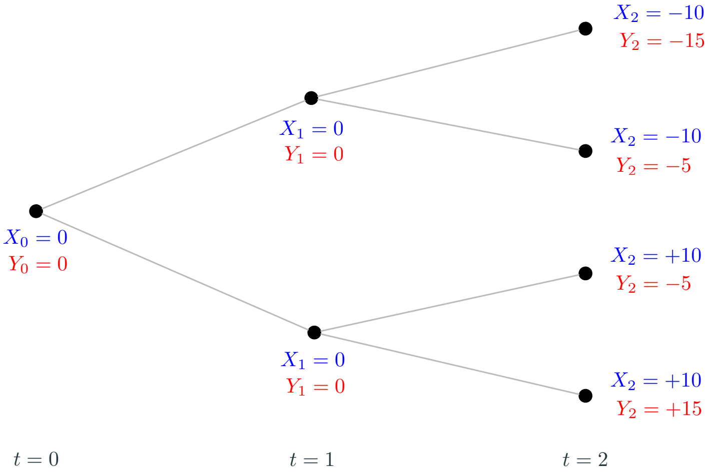
The first try at a risk-averse objective could be to simply take

$$\rho_{t,T}\big(C_{t+1}^\pi, C_{t+2}^\pi, \ldots, C_T^\pi\big) = \text{CVaR}_t^\alpha\big(C_{t+1}^\pi + C_{t+2}^\pi + \cdots + C_T^\pi\big).$$



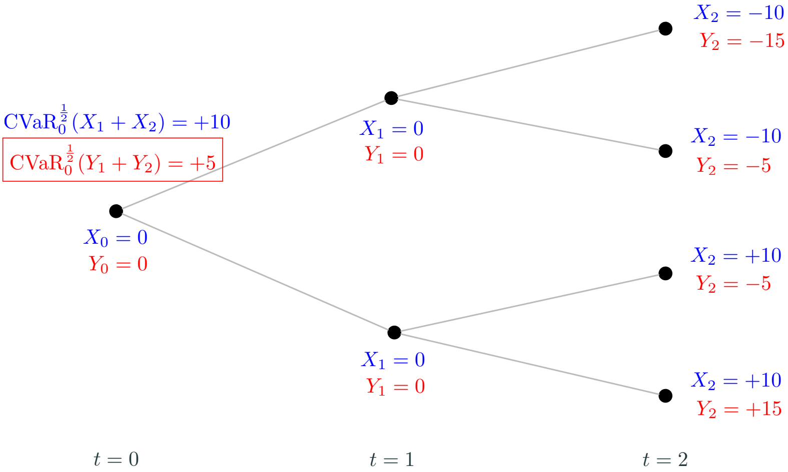
Let's take $\text{CVaR}_t^{\frac{1}{2}}$ (average of 50% of the worst cases). "Costs" — smaller is better.

Red or Blue?



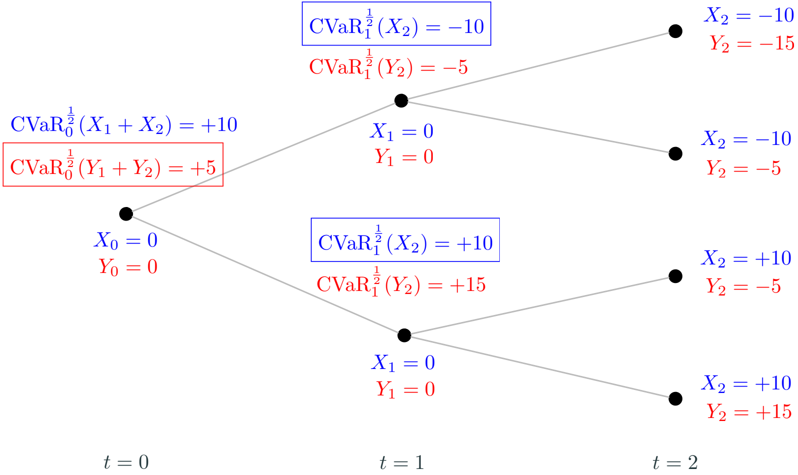
Let's take $\mathrm{CVaR}_t^{\frac{1}{2}}$ (average of 50% of the worst cases). "Costs" — smaller is better.

Red or Blue?



Let's take $\mathrm{CVaR}_t^{\frac{1}{2}}$ (average of 50% of the worst cases). "Costs" — smaller is better.

Red or **Blue**?



Theorem (Ruszczynski, 2010)

Suppose a dynamic risk measure $\{\rho_{t,T}\}_{t=0}^{T}$ *satisfies for all* t

$$\rho_{t,T}(\mathbf{0}) = 0 \quad \text{and} \quad \rho_{t,T}(C_t, C_{t+1}, \ldots, C_T) = C_t + \rho_{t,T}(0, C_{t+1}, \ldots, C_T).$$

Then, "time–consistency" means that $\{\rho_{t,T}\}_{t=0}^{T}$ *has the following nested representation:*

$$\rho_{t,T}(C_t, \ldots, C_T) = C_t + \rho_t\big(C_{t+1} + \rho_{t+1}(C_{t+2} + \cdots + \rho_{T-1}(C_T)\cdots)\big),$$

for some one–step conditional risk measures $\rho_t : \mathcal{Z}_{t+1} \to \mathcal{Z}_t$.

Recall our risk–neutral objective function: $\min_{\pi\in\Pi}\ \mathbf{E}\left[\sum_{t=0}^{T} C_t^\pi\right]$. Expanding, we have

$$\min_{\pi\in\Pi}\ \mathbf{E}_0\Big(C_1^\pi + \mathbf{E}_1\big(C_2^\pi + \cdots + \mathbf{E}_{T-1}(C_T^\pi)\cdots\big)\Big).$$

Given a time-consistent, dynamic risk measure $\{\rho_{t,T}\}_{t=0}^{T}$, a risk–averse version of the objective is

$$\min_{\pi\in\Pi}\ \rho_0\Big(C_1^\pi + \rho_1\big(C_2^\pi + \cdots + \rho_{T-1}(C_T^\pi)\cdots\big)\Big).$$

In applications [5][6][7], dynamic risk measures are built "bottom up" by choosing ρ_t.

[5] A. B. Philpott and V. L. de Matos (2012). "Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion". In: *European Journal of Operational Research* 218.2, pp. 470–483

[6] A. B. Philpott, V. L. de Matos, and E. Finardi (2013). "On solving multistage stochastic programs with coherent risk measures". In: *Operations Research* 61.4, pp. 957–970

[7] A. Shapiro et al. (2013). "Risk neutral and risk averse stochastic dual dynamic programming method". In: *European Journal of Operational Research* 224.2, pp. 375–391

The State-Action Value Function Formulation

The Bellman recursion is analogous to that of the risk-neutral case. For each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$Q_t^*(s, a) = \rho_t\Big(c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1})\Big) \text{ for } t = 0, 1, \ldots, T-1,$$

$$Q_T^*(s, a) = 0.$$

We choose ρ_t to be of the form

$$\rho_t(X) = (1 - \lambda)\, \mathbf{E}\big[X \,|\, \mathcal{F}_t\big] + \lambda\, \rho_t^{\alpha}(X),$$

where ρ_t^{α} is from a particular class called quantile–based risk measures (QBRM).

We now consider the following questions.

- Can we develop data-driven approximate dynamic programming (ADP) algorithms to approximate Q^* and make risk-averse decisions?
- Risk inherently deals with rare, but very costly events; in a simulated setting, can we learn to sample these "risky" events?

Definition

The (conditional) quantile or value at risk (VaR) of an $\mathcal{F}_{t+1}$–measurable random variable X is given by

$$q_t^\alpha(X) = \inf_{U \in \mathcal{Z}_t} \{\mathbf{P}(X \leq U \mid \mathcal{F}_t) \geq \alpha\}.$$

for a risk-level $\alpha \in (0,1)$.

Definition

Given a finite set of risk-levels $\alpha = (\alpha_i)_{i \in \mathcal{I}}$, a parameter $\lambda \in (0,1)$, and a risk aversion function Φ, we define the following class of QBRMs:

$$\rho_t^\alpha(X) = \mathbf{E}\left[\Phi\left(X, q_t^{\alpha_1}(X), q_t^{\alpha_2}(X), \ldots, q_t^{\alpha_m}(X)\right) \,\middle|\, \mathcal{F}_t\right],$$

Definition

The (conditional) quantile or value at risk (VaR) of an $\mathcal{F}_{t+1}$–measurable random variable X is given by

$$q_t^{\alpha}(X) = \inf_{U \in \mathcal{Z}_t} \{\mathbf{P}(X \leq U \,|\, \mathcal{F}_t) \geq \alpha\}.$$

for a risk-level $\alpha \in (0,1)$.

Definition

Given a finite set of risk-levels $\alpha = (\alpha_i)_{i \in \mathcal{I}}$, a parameter $\lambda \in (0,1)$, and a risk aversion function Φ, we define the following class of QBRMs:

$$\rho_t^{\alpha}(X) = \mathbf{E}\Big[\Phi\big(X, q_t^{\alpha_1}(X), q_t^{\alpha_2}(X), \ldots, q_t^{\alpha_m}(X)\big) \,\big|\, \mathcal{F}_t\Big],$$

Conditional Value at Risk (Our Focus)

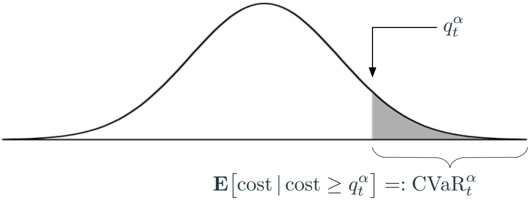
One of the most commonly used risk measures, conditional value at risk (CVaR)[8], is a QBRM with

$$\Phi(X, q^\alpha) = q^\alpha + \frac{1}{1-\alpha}\big[X - q^\alpha\big]^+.$$

A popular form of ρ_t is thus

$$\rho_t(X) = (1-\lambda)\,\mathbf{E}\big[X \,|\, \mathcal{F}_t\big] + \lambda\ \mathrm{CVaR}_t^\alpha(X),$$

which we use numerical experiments.



Other examples: VaR, piecewise constant distortion risk measures, GlueVaR, etc.

[8] R. T. Rockafellar and S. Uryasev (2000). "Optimization of conditional value-at-risk". In: *Journal of Risk* 2, pp. 21–41

The algorithm is based on the following relationships. For $t = 0, 1, \ldots, T-1$, along with the Bellman recursion, we also define an auxiliary variable u^* to refer to the α-quantiles:

$$Q_t^*(s, a) = \rho_t\Big(c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1})\Big),$$

$$u_t^*(s, a) = q^\alpha\Big(c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1})\Big).$$

Important Relationship

Substituting the definition of the risk-measure, we have

$$\begin{aligned} Q_t^*(s, a) = \mathbf{E}\Big[&(1-\lambda)\left[c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1})\right] \\ &+ \lambda\,\Phi\Big(c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1}), u_t^*(s, a)\Big) \,\Big|\, S_t = s,\ a_t = a\Big]. \end{aligned}$$

Algorithm Idea

We employ forward simulation algorithm with two update steps using one sample path of data for each iteration.

- Use "intertwined" approximations $\bar{u}^n$ and $\bar{Q}^n$ to track u^* and Q^*.
- $\bar{Q}$ can be updated using the estimate $\bar{u}$.
- At the same time, $\bar{u}$ can be updated using the estimate $\bar{Q}$.

Let (S_t^n, a_t^n) be the state visited at time t, iteration n, and W_{t+1}^n be a sample of the information process in iteration n. The structure of the algorithm is as follows.

Step 1. Set $n = 1$.

Step 2. For $t = 0, 1, 2, \ldots, T-1$ do:

- **Step 2a.** **Visit** a state (S_t^n, a_t^n).
- **Step 2b.** **Sample** information process W_{t+1}^n.
- **Step 2c.** **Update** auxiliary variable $\bar{u}_t^n$.
- **Step 2d.** **Update** value function $\bar{Q}_t^n$.

Step 3. If $n < N$ (stopping iteration), increment n and return to **Step 2**.

Let $\gamma_t^n(s,a)$ and $\eta_t^n(s,a)$ be stepsizes and define

$$\hat{v}_t^n(s,a) = c_t(s,a,W_{t+1}^n) + \min_{a_{t+1}} \bar{Q}_{t+1}^{n-1}(S_{t+1}, a_{t+1})$$

to be an observation of the cost-to-go.

First Approximation Step

The update to the auxiliary variable $\bar{u}$ is given by

$$\bar{u}_t^n(s,a) = \bar{u}_t^{n-1}(s,a) - \gamma_t^n(s,a)\left[1 - \frac{1}{1-\alpha}\mathbf{1}\left\{\hat{v}_t^n(s,a) \geq \bar{u}_t^{n-1}(s,a)\right\}\right].$$

Second Approximation Step

The update to the value function approximation $\bar{Q}$ is given by

$$\bar{Q}_t^n(s,a) = \left(1-\eta_t^n(s,a)\right)\bar{Q}_t^{n-1}(s,a) + \eta_t^n(s,a)\left[(1-\lambda)\,\hat{v}_t^n(s,a) + \lambda\,\Phi\left(\hat{v}_t^n(s,a), \bar{u}_t^{n-1}(s,a)\right)\right].$$

Let $\gamma_t^n(s,a)$ and $\eta_t^n(s,a)$ be stepsizes and define

$$\hat{v}_t^n(s,a) = c_t(s,a,W_{t+1}^n) + \min_{a_{t+1}} \bar{Q}_{t+1}^{n-1}(S_{t+1}, a_{t+1})$$

to be an observation of the cost-to-go.

First Approximation Step

The update to the auxiliary variable $\bar{u}$ is given by

$$\bar{u}_t^n(s,a) = \bar{u}_t^{n-1}(s,a) - \gamma_t^n(s,a)\left[1 - \frac{1}{1-\alpha}\mathbf{1}\left\{\hat{v}_t^n(s,a) \geq \bar{u}_t^{n-1}(s,a)\right\}\right].$$

Second Approximation Step

The update to the value function approximation $\bar{Q}$ is given by

$$\bar{Q}_t^n(s,a) = \left(1 - \eta_t^n(s,a)\right)\bar{Q}_t^{n-1}(s,a) + \eta_t^n(s,a)\left[(1-\lambda)\,\hat{v}_t^n(s,a) + \lambda\,\Phi\left(\hat{v}_t^n(s,a), \bar{u}_t^{n-1}(s,a)\right)\right].$$

Theorem (Jiang and Powell, 2015)

Under several assumptions (the typical stepsize conditions, states sampled infinitely often, Lipschitz distribution functions), Dynamic-QBRM ADP generates a sequence of iterates $\bar{Q}^n$ such that

$$\bar{u}_t^n(s,a) \to u_t^*(s,a), \;\; \bar{Q}_t^n(s,a) \to Q_t^*(s,a) \;\; a.s.$$

Theorem (Jiang and Powell, 2015)

Under similar assumptions, Dynamic-QBRM ADP generates a sequence of iterates $\bar{Q}^n$ that satisfies

$$\mathbf{E}\left[\|\bar{Q}^n - Q^*\|^2\right] \le \mathcal{O}(1/n).$$

Theorem (Jiang and Powell, 2015)

Under several assumptions (the typical stepsize conditions, states sampled infinitely often, Lipschitz distribution functions), Dynamic-QBRM ADP generates a sequence of iterates $\bar{Q}^n$ such that

$$\bar{u}_t^n(s,a) \to u_t^*(s,a), \;\; \bar{Q}_t^n(s,a) \to Q_t^*(s,a) \;\; a.s.$$

Theorem (Jiang and Powell, 2015)

Under similar assumptions, Dynamic-QBRM ADP generates a sequence of iterates $\bar{Q}^n$ that satisfies

$$\mathbf{E}\left[\| \bar{Q}^n - Q^* \|^2\right] \leq \mathcal{O}\left(1/n\right).$$

Empirical Behavior for a problem using $\rho_t^\alpha = \text{CVaR}_t^\alpha$ with $\lambda = 0.5$ and $\alpha = 0.99$. Results are for a fixed state in the energy arbitrage problem. The actual limit points are given by:

$$u^* \approx -555 \text{ and } Q^* \approx -387.$$

Volatile approximations like these are not conducive to ADP.

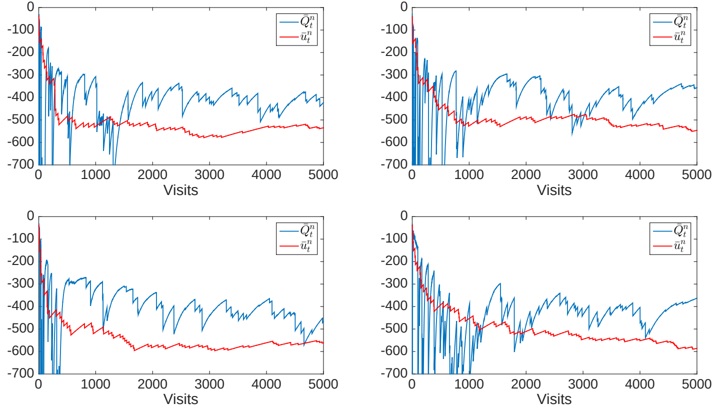


Figure 10: Sample Paths of Dynamic-QBRM ADP

Reasons for Poor Behavior

- By definition, when α is close to 1, the "risky events" are very rarely sampled.
- Combined with the fact that when $\alpha \to 1$, $\frac{1}{1-\alpha} \to \infty$, we are generating very volatile observations.

However, assume we are in a simulated setting and know the distribution of W_t. Then we can design a method to move our sampling towards to "risky" region.

Recall:

$$Q_t^*(s, a) = \mathbf{E}\Big[(1-\lambda)\left[c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1})\right] + \lambda\,\Phi\big(c_t(s, a, W_{t+1}) + \min_{a_{t+1}} Q_{t+1}^*(S_{t+1}, a_{t+1}), u_t^*(s, a)\big) \,\Big|\, S_t = s,\ a_t = a\Big].$$

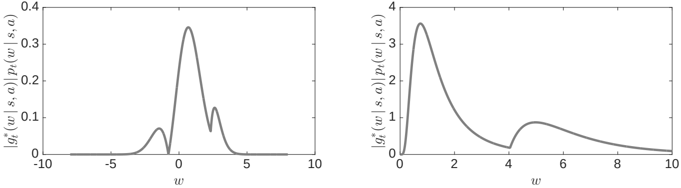
Let $(W_{t+1} \,|\, S_t = s,\ a_t = a) \sim p_t(w \,|\, s, a)$, a density that we assume is known. Notice that

$$Q_t^*(s, a) = \int g_t^*(w \,|\, s, a)\, p_t(w \,|\, s, a)\, dw,$$

where g_t^* depends on c_t, u_t^*, and Q_{t+1}^*.

By the principle of importance sampling, we should sample from a distribution that matches the shape of the (absolute value of) integrand

$$|g_t^*(w\,|\,s,a)|\,p_t(w\,|\,s,a).$$



(a) $\mathcal{N}(0,1)$ with $c_t(s,a,w) = w$ (b) $\log\mathcal{N}(0,0.5^2)$ with $c_t(s,a,w) = w^2$

Figure 11: Examples of Integrands of under Normal and Lognormal Distributions

Problem: we do not know $g_t^*(w\,|\,s,a)$ (as most IS procedures assume).

Solution: run an adaptive procedure in conjunction with Dynamic-QBRM ADP, using $g_t^n(w\,|\,s,a)$, an approximation derived from $\bar{u}^n$ and $\bar{Q}^n$, instead of $g_t^*(w\,|\,s,a)$.

Our Approach is Adaptive and Makes Use of Biased Observations of Integrand

We propose the following technique, based on importance sampling.

- Specify a set of "basis densities" $\phi = (\phi_t^k)_{k=1}^K$. Sampling distribution at iteration n, time t, and state (s, a) is taken to be proportional to

$$\sum_k \bar{\theta}_t^{k,n}(s, a)\, \phi_t^k(w) \approx \left|g_t^*(w|\, s, a)\right| p_t(w|\, s, a).$$

  Motivation: Easy in practice to place several unimodal densities in the domain to approximate multiple risky regions.

- Observe a noisy, biased sample of the integrand, $|g_t^*\left(w\,|\, s, a\right)|\, p_t(w\,|\, s, a)$, using approximations $\bar{u}_t^n$ and $\bar{Q}_{t+1}^n$.
- Update $\bar{\theta}_t^{k,n}(s, a)$ iteratively to minimize mean square error to the target density using stochastic approximation.

Now, let $\beta_t^n(s, a)$ be another stepsize.

First Approximation Step

The update to the auxiliary variable $\bar{u}$ is given by

$$\bar{u}_t^n(s, a) = \bar{u}_t^{n-1}(s, a) - \gamma_t^n(s, a) \left[1 - \frac{1}{1-\alpha} \mathbf{1}\left\{\hat{v}_t^n(s, a) \geq \bar{u}_t^{n-1}(s, a)\right\}\right].$$

Second Approximation Step

The update to the value function approximation $\bar{Q}$ is given by

$$\bar{Q}_t^n(s, a) = (1 - \eta_t^n(s, a))\, \bar{Q}_t^{n-1}(s, a) + \eta_t^n(s, a) \left[(1-\lambda)\, \hat{v}_t^n(s, a) + \lambda\, \Phi^\alpha(\hat{v}_t^n(s, a), \bar{u}_t^n(s, a))\right].$$

Update Step for the Sampling Distribution

The update step for the weights is given by

$$\bar{\theta}_t^n(s, a) = \left[\bar{\theta}_t^{n-1} + \beta_t^n(s, a) \left(\text{approximate direction to better represent “risky” regions}\right)\right]^+.$$

New Update Steps for the Dynamic-QBRM ADP Algorithm

Now, let $\beta_t^n(s, a)$ be another stepsize.

First Approximation Step

The update to the auxiliary variable $\bar{u}$ is given by

$$\bar{u}_t^n(s, a) = \bar{u}_t^{n-1}(s, a) - \gamma_t^n(s, a) \left[1 - \frac{1}{1-\alpha} \mathbf{1}\left\{\hat{v}_t^n(s, a) \geq \bar{u}_t^{n-1}(s, a)\right\}\right].$$

Second Approximation Step

The update to the value function approximation $\bar{Q}$ is given by

$$\begin{aligned} \bar{Q}_t^n(s, a) = &\left(1 - \eta_t^n(s, a)\right) \bar{Q}_t^{n-1}(s, a) \\ &+ \eta_t^n(s, a) \left[(1-\lambda)\, \hat{v}_t^n(s, a) + \lambda\, \Phi^\alpha(\hat{v}_t^n(s, a), \bar{u}_t^n(s, a))\right]. \end{aligned}$$

Update Step for the Sampling Distribution

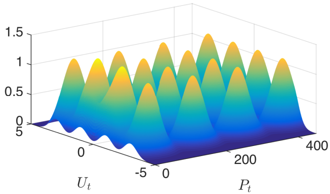
The update step for the weights is given by

$$\bar{\theta}_t^n(s, a) = \left[\bar{\theta}_t^{n-1} + \beta_t^n(s, a) \left(\text{approximate direction to better represent “risky” regions}\right)\right]^+.$$

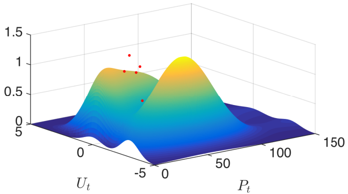
Update Step for the Sampling Distribution

The update step for the weights is given by

$$\theta_t^n(s,a) = \left[\theta_t^{n-1} + \beta_t^n(s,a)\left(\text{approximate direction to better represent “risky” regions}\right)\right]^+.$$



(a) Basis distributions ϕ^k (equally weighted)



(b) Sampling density after 500 iterations

Figure 12: Example Illustration of Risk Directed Sampling ($\lambda = 0.5$)

For a function h, let the projection operator Π_ϕ be given by

$$\Pi_\phi h = \arg\min_{\theta \geq 0} \mathbf{E}\Big[\big(\theta^\top \phi(X) - h(X)\big)^2\Big],$$

where X is some distribution against which we measure error.

Theorem (Convergence of the Sampling Density, Jiang and Powell, 2015)

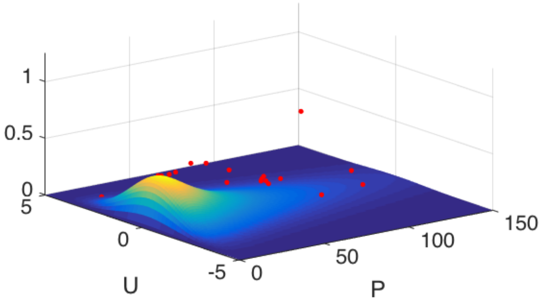
For each t and (s, a), our approximations converge to the optimal sampling density (in the sense of closest shape under ϕ) as if the unknown integrand $g_t^*(\cdot \,|\, s, a)$ were known.

$$\bar{\theta}_t^n(s, a) \longrightarrow \Pi_\phi\Big[\,\big|g_t^*(\cdot \,|\, s, a)\big|\; p_t(\cdot \,|\, s, a)\Big] \quad a.s.$$

Here's how it works in practice.

Recall that our energy arbitrage problem contained two random variables: P_t (the spot prices of electricity) and U_t (the amount of energy left after "sharing").

In this example, we employ a grid of bivariate normal distributions whose contribution to the sampling density is determined by the learned weights.



Some Sample Paths of Dynamic-QBRM ADP with Risk-Directed Sampling

Empirical Behavior for the exact same problem as before, using risk-directed sampling. The actual limit points are given by:

$$u^* \approx -555 \text{ and } Q^* \approx -387.$$

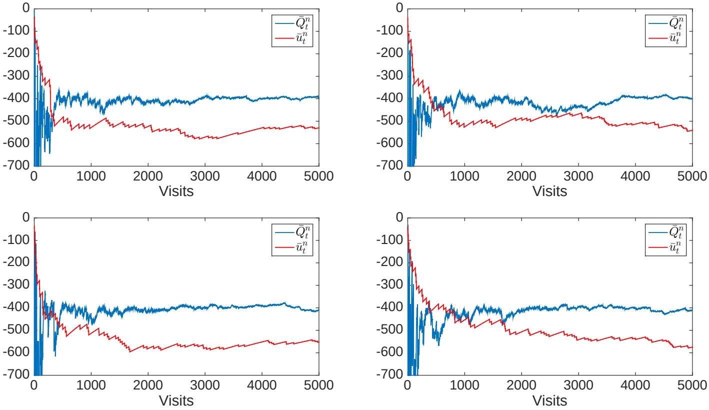


Figure 13: Sample Paths of Dynamic-QBRM ADP with Risk-Directed Sampling

This is a drastic improvement over what we had previously, reproduced here.

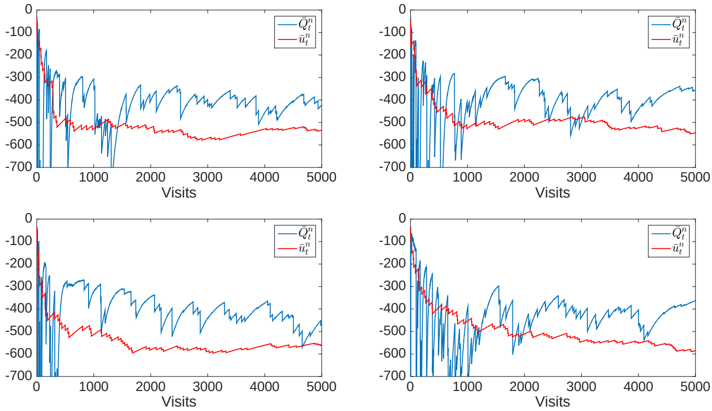
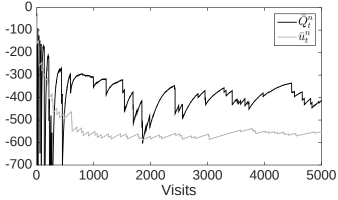


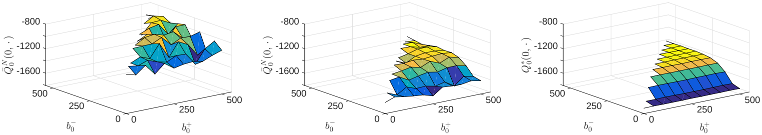
Figure 14: Sample Paths of Dynamic-QBRM ADP



(a) Without RDS

(b) With RDS

Figure 15: Sample Paths of Approximations Generated by Dynamic–QBRM ADP ($\lambda = 0.5$)



(a) Without RDS

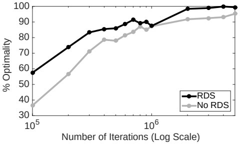
(b) With RDS

(c) Optimal

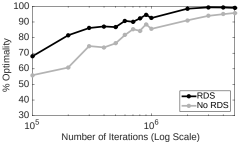
Figure 16: Surface Plots of Value Function Approximations at $t = 0$ ($\lambda = 0.5$)

Here we compute an optimality percentage of approximate policies via

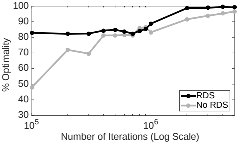
$$V_t^\pi(s) = \rho_t\big(c_t(s, A_t^\pi(s), W_{t+1}) + V_{t+1}^\pi(S_{t+1}^\pi)\big) \text{ for all } s \in \mathcal{S},\ t \in \mathcal{T},$$
$$V_T^\pi(s) = 0 \text{ for all } s \in \mathcal{S}.$$



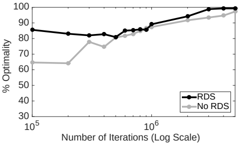
(a) $\lambda = 0.6$



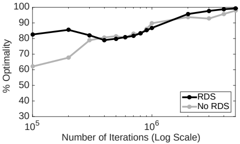
(b) $\lambda = 0.55$



(c) $\lambda = 0.5$



(d) $\lambda = 0.45$



(e) $\lambda = 0.4$

Figure 17: Comparison of Dynamic–QBRM ADP with and without RDS

Recall that $\rho_t(X) = (1-\lambda)\,\mathbf{E}\big[X \,|\, \mathcal{F}_t\big] + \lambda\ \mathrm{CVaR}_t^{\alpha}(X)$.

We examine the risk vs. reward tradeoff of risk-averse policies on the energy arbitrage problem by solving it for $\lambda = 0,\ 0.05,\ 0.1, \ldots, 0.5$ and $\alpha = 0.99$.

Let $B_t^{\pi} = \{\text{stockout events under } \pi\}$ and

$$\mathrm{Risk}(\pi) = \mathbf{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{1}_{B_t^{\pi}}\right] \quad \text{and} \quad \mathrm{Reward}(\pi) = \mathbf{E}\left[-\sum_{t=0}^{T-1} C_t^{\pi}\right].$$

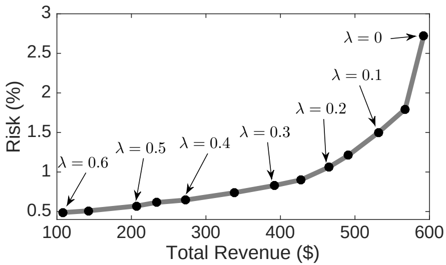


Figure 18: Risk–Reward Frontier from Dynamic–QBRM ADP with RDS for $N = 5{,}000{,}000$

Relationships between Parameterized Risk-Neutral Policies to Corresponding Risk-Averse Policies

- Under what conditions (properties of the risk measure, properties of the problem) is it true that

$$\text{optimal risk-averse policy} = f(\text{optimal risk-neutral policy, risk parameters})$$

  where f is a simple, implementable relationship?
- Simple Example: if order-up-to policy is optimal with a risk-neutral objective, is the risk-averse optimal policy another order-up-to policy with a "relaxed" threshold?

Future Research Questions

Forecasts in Dynamic Programming

- Without re-optimization, forecasts of quantities that influence energy prices (e.g., temperature, gas prices) can be difficult to fully and rigorously incorporate into sequential problems (curse of dimensionality).
- Can we develop theory and a set of conditions to understand how optimal policies (or optimal value functions) behave as a function of changing forecasts?

Exploration in Dynamic Programming

- In the bidding problem, the decision space is very large. Without convexity, we need to search a large part of it to find an optimal decision, even in the training phase.
- Can we use perfect foresight upper bounds to make exploration-exploitation decisions for approximate dynamic programming?

Applications in Energy and Sustainability

- Policymakers and utilities are interested in an accurate economic valuation of solar that takes into account 1) the role of solar in conjunction with conventional generation, 2) the economics of co-located storage, and 3) forecasting issues.
- A risk-based analysis of strategies for a quickly growing industry, demand response. For example, what is the optimal notification time to give customers ahead of demand response events?

Thank you!