From stochastic search to bandit problems to dynamic programs: Fresh perspectives of some old problems

Institute for Mathematics and its Applications

July 24-25, 2017

Warren B. Powell
Princeton University
Department of Operations Research and Financial Engineering

© 2017 Warren B. Powell, Princeton University
Outline

- Applications
- Problem classes
- Designing policies
Outline

- Applications
- Problem classes
- Designing policies
An energy storage problem

- An basic energy storage problem:

» How do we use our storage device to balance energy from renewables and the grid to serve a time-dependent load?
Grid level storage

- Congested grid:
  - Green and blue circles indicate energy storage
SMART-Solar

Congested grid:

» Green and blue circles indicate energy storage
Load acceptance

Load commitments four days from now
Load acceptance

Load commitments four days from now

Questions:
- When to say yes?
- What price?
Truckload brokerages

- Finding the best price
Truckload brokerages

Bid price
Driverless fleets of electric vehicles

Problems

- Assigning cars to riders
- Determining surge prices
Applications

- Biomedical research
  - How do we find the best drug to cure cancer?
  - There are millions of combinations, with laboratory budgets that cannot test everything.
  - We need a method for sequencing experiments.

*Optimal learning research by Diana Negoescu, now a professor at U. of Minn.*
Drug discovery

- Designing molecules

> X and Y are sites where we can hang substituents to change the behavior of the molecule.

Optimal learning research by Diana Negoescu, now a professor at U. of Minn.
Modeling

Before we can solve complex problems, we have to know how to think about them.

The biggest challenge when making decisions under uncertainty is modeling.
Modeling

For deterministic problems, we speak the language of mathematical programming

» Linear programming:

$$\min_x cx$$
$$Ax = b$$
$$x \geq 0$$

» For time-staged problems

$$\min_{x_0, \ldots, x_T} \sum_{t=0}^{T} c_t x_t$$
$$A_t x_t - B_{t-1} x_{t-1} = b_t$$
$$D_t x_t \leq u_t$$
$$x_t \geq 0$$
Stochastic programming
Markov decision processes
Reinforcement learning
Optimal control
Model predictive control
Robust optimization
Approximate dynamic programming
Online computation
Simulation optimization
Stochastic search
Decision analysis
Stochastic control
Simulation optimization
Dynamic Programming and Optimal Control
Introduction to Stochastic Search and Optimization
Multi-Armed Bandit Allocation Indices
Optimal Learning
Optimal Control
Optimal Learning
Markov Decision Processes
Discrete Stochastic Dynamic Programming
Introduction to Stochastic Programming
Modeling stochastic, dynamic problems

- We lack a standard language for modeling sequential, stochastic decision problems.
  - There are five dimensions to any sequential decision problem
    - State variables
    - Decision variables
    - Exogenous information
    - Transition function
    - Objective function

- This framework draws heavily from Markov decision processes and the control theory communities, but it is not the standard form used anywhere.
Modeling dynamic problems

- The system state:

  \[ S_t = (R_t, I_t, B_t) = \text{System state, where:} \]
  \[ R_t = \text{Resource state (physical state)} \]
  \[ \quad \text{Location/status of truck/train/plane} \]
  \[ \quad \text{Energy in storage} \]
  \[ I_t = \text{Information state} \]
  \[ \quad \text{Prices} \]
  \[ \quad \text{Weather} \]
  \[ B_t = \text{Belief state} \]
  \[ \quad \text{Distn. of belief about demand response} \]
  \[ \quad \text{Distn. of belief about the status of equipment} \]
  \[ \quad \text{Belief about performance of a new drug} \]
Modeling stochastic, dynamic problems

The universal objective function

\[
\max_{\pi} \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} C_t (S_t, X_\pi (S_t), W_{t+1}) \mid S_0 \right\}
\]

Expectation over all random outcomes

Finding the best policy

Decision function (policy)

State variable

Contribution function

Initial state variable

New information

Given a system model (transition function)

\[
S_{t+1} = S^M \left( S_t, x_t, W_{t+1}(\omega) \right)
\]

Now we just have to find the best policy.
Modeling

**Deterministic**

» Objective function
\[ \min \sum_{x_0, \ldots, x_T}^{T} c_t x_t \]

» Decision variables:
\( (x_0, \ldots, x_T) \)

» Constraints:
  • at time \( t \)
    \[ A_t x_t = R_t \]
    \[ x_t \geq 0 \]
  • Transition function
    \[ R_{t+1} = b_{t+1} + B_t x_t \]

**Stochastic**

» Objective function
\[ \max_{\pi} E^\pi \left\{ \sum_{t=0}^{T} \gamma^t C(S_t, X_t^\pi(S_t), W_{t+1}) \middle| S_0 \right\} \]

» Policy
\[ X^\pi : S \mapsto X \]

» Constraints at time \( t \)
\[ x_t = X_t^\pi(S_t) \in X_t \]

» Transition function
\[ S_{t+1} = S^M(S_t, x_t, W_{t+1}) \]

» Exogenous information
\( (W_1, W_2, \ldots, W_T) \)
Outline

- Applications
- Problem classes
- Designing policies
Problem classes

State independent problems

» Generic stochastic search problem:

\[
\max_{x \in X} \mathbb{E} F(x, W)
\]

» We may observe \( F(x, W) \) for a given \( x \), but we do not know (or cannot compute) \( \mathbb{E} F(x, W) \)

» Classical example: Newsvendor problem

\[
\max_{x \in X} \mathbb{E} F(x, W) = \mathbb{E} \left\{ p \min(x, W) - cx \right\}
\]

We can use various adaptive learning algorithms to either learn \( \bar{F}^n(x) \) » \( \mathbb{E} F(x, W) \) or \( x^n \) » \( x^* \)

» State \( S^n \) only captures belief \( \bar{F}^n(x) \) about \( \mathbb{E} F(x, W) \) or the belief \( x^n \) about the optimal solution \( x^* \).

» We call these learning problems.
Problem classes

- State dependent problems
  - Here the function \( F(x, W) \) or the constraints \( X \) may be state dependent.
  - For example, imagine our newsvendor with a dynamic price \( p_t \) subject to a constraint \( 0 \leq x \leq R_t \) where \( R_t \) is the inventory available at time \( t \):
    \[
    \max_{0 \leq x \leq R_t} \mathbb{E}F(x, W) = \mathbb{E}\{p_t \min(x, W) - cx\}
    \]
  - Now we have a state variable \( S_t = (R_t, I_t, B_t) \)
    \( R_t = \) Physical state (inventory)
    \( I_t = \) Information state (e.g. price)
    \( B_t = \) Belief state (e.g. about \( \mathbb{E}F(x, W) \))
Problem classes

Objective functions:

» Offline (terminal reward)
  • Classical asymptotic formulation:
    \[
    \max_{x \in X} \mathbb{E} F(x, W) \quad \text{“stochastic search”}
    \]
  • Normally we want to find a sequence
    \[
    \lim_{n \to \infty} x^n \supseteq x^*
    \]
  • Now imagine we have a policy (or algorithm) \( X^p(S^n | q) \)
    parameterized by \( q \) which gives us a decision
    \[
    x^n = X^p(S^n | q)
    \]
  • Let \( x^{p,N} \) be the solution produced after \( N \) iterations.
  • We wish to find the best policy \( X^p(S^n | q) \) that solves:
    \[
    \max_p \mathbb{E} F(x^{p,N}, W) \quad \text{“ranking and selection”}
    \]
    \[
    \quad \text{“stochastic search”}
    \]
Problem classes

Objective functions:

» Online (cumulative reward)
  • Now imagine that we want to maximize our contributions over time as we are learning.
  • Our objective function (in finite time) becomes:

\[
\max_{\rho} \mathbb{E}_{\hat{\mathbf{A}}} \sum_{n=0}^{N-1} F(X^n, S^n | q, W^{n+1})
\]

• Now we have to balance learning while maximizing rewards (earning). This is the classic exploration vs. exploitation problem.
• This is widely studied as the “multiarmed bandit problem.”
Problem classes

The problem class matrix:

<table>
<thead>
<tr>
<th>State independent problems</th>
<th>Offline Terminal reward</th>
<th>Online Cumulative reward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\max_\pi \mathbb{E}{F(x^{\pi,N}, W)</td>
<td>S_0}$</td>
</tr>
<tr>
<td></td>
<td>Stochastic search (1)</td>
<td>Multiarmed bandit problem (2)</td>
</tr>
<tr>
<td>State dependent problems</td>
<td>$\max_{\pi_{\text{trn}}} \mathbb{E}{C(S, X^{\pi_{\text{imp}}, (S</td>
<td>\theta^{\text{imp}})}, W)</td>
</tr>
<tr>
<td></td>
<td>Offline dynamic programming (4)</td>
<td>Online dynamic programming (3)</td>
</tr>
</tbody>
</table>

» Let’s take a tour of these problem classes.
Problem classes

- Group (1): Offline learning problems
  - We wish to find the best policy \( X^\rho (S^n) \) to learn the best implementation decision \( x^\rho, N \):
    \[
    \max_{\rho} \mathbb{E} F(x^\rho, N, W)
    \]
  - Policies may be:
    - Derivative-based
      \[
      x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})
      \]
    - … or derivative-free
      \[
      x^n = X^{\pi} (S^n) \in \{ x_1, \ldots, x_M \}
      \]
      where \( S^n \) is our state of belief about \( \mathbb{E} F(x, W) \).
We want a method (an algorithm) that produces the best solution by time $N$:

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

Assume that our stepsize rule is

$$\alpha_n = \frac{\theta}{\theta + N^n}$$

where $N^n$ = number of times the solution has not improved.

After $n$ iterations, our “state” is

$$S^n = (x^n, N^n)$$

Given the state $S^n$ and the parameter $\theta$, we can determine (after sampling $W^{n+1}$) the next state $S^{n+1}$.
Problem classes

- Testing different stepsize rules ("policies")

We want to optimize the rate of convergence:
- Different stepsize rules
- Different ways of computing the gradient
Problem classes

- Derivative-free stochastic search
  - Start by assuming that our set of possible decisions is finite:
    \[ x \in X = \{x_1, x_2, ..., x_M\} \]
  - Assume we have some belief about our function (say, lookup table). Using a Bayesian model, we assume we have a distribution of belief about \( f(x) = \mathbb{E}F(x, W) \) given by
    \[ \mathbb{E}F(x, W) = \mu_x \mathbb{I} N(\mu^0, \beta^0) \]
    where \( \beta^0 \) is the precision where \( \beta^0 = 1 / \sigma^{-2,0} \).
  - We refer to \( S^0 = B^0 = N(\bar{\mu}^0, \beta^0) \) as our prior state of knowledge.
Problem classes

Belief state for ranking and selection

» S is our “state of knowledge”

\[ S_5 = N\left( \mu_5^n, \sigma_5^{2,n} \right) \]

\[ S^n = (S_1^n, \ldots, S_5^n) \]
Updating beliefs

» After \( n \) experiments, our belief is \( \mu_x \sim N(\mu^n_x, \beta^n_x) \)

» Assume that based on this belief, we choose

\[
x = x^n = X^n(\mathcal{S}^n)
\]

to run for our next experiment (experiment \( n+1 \)):

\[
W_{x^n}^{n+1} = \mu_{x^n} + \epsilon^{n+1}
\]

» We update our beliefs using

\[
\mu_{x}^{n+1} = \frac{\beta^n \mu^n_{x} + \beta^W W^{n+1}_x}{\beta^n + \beta^W} \\
\beta_{x}^{n+1} = \beta^n_x + \beta^W
\]

Transition function:

\[
\mathcal{S}^{n+1} = \mathcal{S}^M(S^n, x^n, W^{n+1})
\]

» There are many belief models we can use!
Problem classes

**Designing a policy**

» We need a rule for picking which decision to try next. We call this rule our policy $X^\pi(S^n | \theta)$. Some examples are:

- **Interval estimation:**
  
  $$X^{IE}(S^n, \theta^{IE}) = \arg \max_x \left( \bar{\mu}_x^n + \theta^{IE} \sigma_x^n \right) \quad \sigma_x^n = \text{Std. dev. of } \bar{\mu}_x^n$$

- **Upper confidence bounding**
  
  $$X^{UCB}(S^n, \theta^{UCB}) = \arg \max_x \left( \bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right) \quad N_x^n = \text{No. of times } x \text{ is tested.}$$

- **Thompson sampling:**
  
  $$X^{TS}(S^n) = \arg \max_x \hat{\mu}_x^n \quad \hat{\mu}_x^n \sim N(\bar{\mu}_x^n, \beta_x^n)$$

- **Knowledge gradient (expected value of information):**
  
  $$\nu^{KG,n}(x) = E \left\{ \max_y F(y, K^{n+1}(x)) \right\} - \max_y F(y, K^n)$$
Problem classes

Group (2): Online learning problems

» In online learning, we are maximizing the cumulative reward:

\[
\max_{\rho} \mathbb{E}_{n=0}^{N-1} F(X^p(S^n), W^{n+1})
\]

» If we have a parameterized policy

\[
F(q) = \mathbb{E}_{n=0}^{N-1} F(X^p(S^n | q), W^{n+1})
\]

» … we can use derivative-based policy search

\[
\theta^{n+1} = \theta^n + \alpha_n \nabla_{\theta} F(\theta^n, W^{n+1})
\]

» … or any of the derivative-free policies.

» Although online learning is viewed as a fundamentally different problem class (“bandit problems”), it is simply stochastic search with a different objective.
Problem classes

- **Group (3): Online state-dependent problems**

  » Now consider what happens when the function or constraints depend on dynamically changing data.

  » Begin by writing our one-period problem using

  \[
  \max_{0 \leq x \leq R_t} \mathbb{E}\left\{ p_t \min\left( x_t, W_{t+1} \right) \right\} \\
  C(S_t, x_t, W_{t+1})
  \]

  » Now maximize cumulative rewards using

  \[
  \max_{\rho} \mathbb{E}\left\{ \sum_{t=0}^{T} C(S_t, X_t^\rho(S_t), W_{t+1}) \mid S_0 \right\}
  \]
Problem classes

- Group (3): Online state-dependent problems
  - The online (cumulative reward) objective function...
    \[
    \max_{\rho} \mathbb{E}_{\hat{\alpha}^t} \left[ \sum_{t=0}^{T} C(S_t, X_t^\rho(S_t), W_{t+1}) \right] | S_0
    \]
  - … is how we often write a classical dynamic program.
  - Now compare to our online learning problem (aka multiarmed bandit problem):
    \[
    \max_{\rho} \mathbb{E}^{N-1}_{n=0} F(X^n^\rho(S^n), W^{n+1})
    \]
  - When we solve our bandit problems, we focus on balancing exploration and exploration to maximize rewards over time.
  - For “dynamic programs” we tend to just focus on the quality of the policy rather than on how well it does while learning.
Problem classes

- **Group (4): Offline state-dependent problems**
  
  » Recall that in group (1), our problem was to solve
  
  \[
  \max_{\rho} \mathbb{E} F(x^{\rho,N}, W)
  \]
  
  » where the goal of policy \( \rho \) was to learn the implementation decision \( x^{\rho,N} \).
  
  » For the state dependent problem, we want to solve
  
  \[
  \max_{\rho_{lrn}} \mathbb{E}^{\rho_{imp}} C(S, X^{\rho_{imp}} (S | q^{imp}), W)
  \]
  
  » which means finding the best learning policy \( \rho_{lrn} \) to learn the best implementation policy \( X^{\rho_{imp}} (S | q^{imp}) \).
Problem classes

Group (4): Offline state-dependent problems

» Finding the expectation over states is hard (it depends on the policy), so a common approximation is to replace...

\[
\max_{p_{lrn}} \mathbb{E}^{imp} C(S, X^{imp} (S | q^{imp}), W)
\]

» ... with

\[
\max_{p_{lrn}} \mathbb{E} \sum_{t=0}^{T} \frac{1}{T} C(S_t, X_t^{imp} (S_t | q^{imp}), W_{t+1}) \mid S_0 \]

» A learning policy is typically thought of as an algorithm to find a policy, which we consider below.
The problem class matrix:

<table>
<thead>
<tr>
<th>State independent problems</th>
<th>Offline Terminal reward</th>
<th>Online Cumulative reward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \max_\pi E{F(x^{\pi,N}, W)</td>
<td>S_0} )</td>
</tr>
<tr>
<td></td>
<td>Stochastic search (1)</td>
<td>Multiarmed bandit problem (2)</td>
</tr>
<tr>
<td>State dependent problems</td>
<td>( \max_{\pi, l, n} E{C(S, X^{\pi, imp}(S</td>
<td>\theta^{imp}), W)</td>
</tr>
<tr>
<td></td>
<td>Offline dynamic programming (4)</td>
<td>Online dynamic programming (3)</td>
</tr>
</tbody>
</table>

» The distinction between offline stochastic search and online bandit problems is fairly clear.

» The distinction between offline and online “dynamic programming” has received virtually no attention.
Outline

- Applications
- Problem classes
- Designing policies
Designing policies

- We have to start by describing what we mean by a policy.
  - Definition:
    
    *A policy is a mapping from a state to an action.*
    
    ... *any mapping.*

- How do we search over an arbitrary space of policies?
Designing policies

“Policies” and the English language

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Manner</th>
<th>Ritual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belief</td>
<td>Method</td>
<td>Rule</td>
</tr>
<tr>
<td>Bias</td>
<td>Mode</td>
<td>Style</td>
</tr>
<tr>
<td>Commandment</td>
<td>Mores</td>
<td>Technique</td>
</tr>
<tr>
<td>Conduct</td>
<td>Patterns</td>
<td>Tenet</td>
</tr>
<tr>
<td>Convention</td>
<td>Plans</td>
<td>Tradition</td>
</tr>
<tr>
<td>Culture</td>
<td>Policies</td>
<td>Way of life</td>
</tr>
<tr>
<td>Customs</td>
<td>Practice</td>
<td></td>
</tr>
<tr>
<td>Dogma</td>
<td>Prejudice</td>
<td></td>
</tr>
<tr>
<td>Etiquette</td>
<td>Principle</td>
<td></td>
</tr>
<tr>
<td>Fashion</td>
<td>Procedure</td>
<td></td>
</tr>
<tr>
<td>Formula</td>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Habit</td>
<td>Protocols</td>
<td></td>
</tr>
<tr>
<td>Laws/bylaws</td>
<td>Recipe</td>
<td></td>
</tr>
</tbody>
</table>
Designing policies

Two fundamental strategies:

1) Policy search – Search over a class of functions for making decisions to optimize some metric.

\[
\max_{\pi = (f \in F, \theta^f \in \Theta^f)} E \left\{ \sum_{t=0}^{T} C\left(S_t, X_{t}^{\pi}(S_t | \theta)\right) \mid S_0 \right\}
\]

2) Lookahead approximations – Approximate the impact of a decision now on the future.

\[
X_t^*(S_t) = \arg \max_{x_t} \left\{ C(S_t, x_t) + E \left\{ \max_{\pi \in \Pi} \left\{ E \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right\}
\]
Designing policies

Policy search:

1a) Analytical functions that directly map states to actions ("policy function approximations" or PFAs) \( x_t = X^{PFA}(S_t \mid \theta) \)
   - Lookup tables
     - "when in this state, take this action"
   - Parametric functions
     - Order-up-to policies: if inventory is less than \( s \), order up to \( S \).
     - Affine policies - \( x_t = X^{PFA}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t) \)
     - Neural networks
   - Locally/semi/non parametric
     - Requires optimizing over local regions

1b) Maximizing analytical approximations of costs and/or constraints ("cost function approximations" or CFAs)
   - Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)

\[
X^{CFA}(S_t \mid \theta) = \arg \max_{x_t \in \overline{X}_\pi(\theta)} \overline{C}^{\pi}(S_t, x_t \mid \theta)
\]
Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  
  » An optimal policy (based on looking ahead):

\[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^\pi (S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right) \]
Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  » An optimal policy (based on looking ahead):

\[
X^*_t(S_t) = \arg\max_{x_t} \left[ C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X^\pi_{t'}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right]
\]

2b) Approximate the problem:

\[
X^{LA}_t(S_t) = \arg\max_{x_t} \left[ C(S_t, x_t) + \tilde{\mathbb{E}}\left\{ \max_{\tilde{\pi} \in \tilde{\Pi}} \left\{ \tilde{\mathbb{E}}\sum_{t'=t+1}^{T} C(\tilde{S}_{t'}, \tilde{X}^{\tilde{\pi}}(\tilde{S}_{t'})) \mid \tilde{S}_{t+1} \right\} \mid S_t, x_t \right\} \right]
\]
Four (meta)classes of policies

1) Policy function approximations (PFAs)
   » Lookup tables, rules, parametric/nonparametric functions

2) Cost function approximation (CFAs)
   » \( X^{CFA}(S_t | \theta) = \arg \max_{x_t \in \pi_t(\theta)} C^\pi(S_t, x_t | \theta) \)

3) Policies based on value function approximations (VFAs)
   » \( X^{VFA}_t(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \tilde{V}^x_t \left( S^x_t(S_t, x_t) \right) \right) \)

4) Direct lookahead policies (DLAs)
   » Deterministic lookahead/rolling horizon proc./model predictive control
     \( X^{LA-D}_t(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} C(\tilde{S}_t, \tilde{x}_t) + \sum_{t'=t+1} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \)
   » Chance constrained programming
     \( P[A_t x_t \leq f(W)] \leq 1 - \delta \)
   » Stochastic lookahead /stochastic prog/ Monte Carlo tree search
     \( X^{LA-S}_t(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+T}} C(\tilde{S}_t, \tilde{x}_t) + \sum_{\tilde{\omega} \in \Omega_t} p(\tilde{\omega}) \sum_{t'=t+1} C(\tilde{S}_{t'}, (\tilde{\omega}), \tilde{x}_{t'}(\tilde{\omega})) \)
   » “Robust optimization”
     \( X^{LA-RO}_t(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_t, \tilde{x}_t) + \sum_{t'=t+1} C(\tilde{S}_{t'}, (w), \tilde{x}_{t'}(w)) \)
Four (meta)classes of policies

1) Policy function approximations (PFAs)
   » Lookup tables, rules, parametric/nonparametric functions

2) Cost function approximation (CFAs)
   » \( X^{CFA}(S_t | \theta) = \arg \max_{x_t \in \bar{x}_t^\pi(\theta)} \bar{C}^\pi(S_t, x_t | \theta) \)

3) Policies based on value function approximations (VFAs)
   » \( X^{VFA}(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \bar{V}_t^x(S_t^x(S_t, x_t)) \right) \)

4) Direct lookahead policies (DLAs)
   » Deterministic lookahead/rolling horizon proc./model predictive control
     \( X^{LA-D}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} C(\tilde{S}_t, \tilde{x}_t) + \sum_{t'=t+1} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \)
   » Chance constrained programming
     \( P[A_t x_t \leq f(W)] \leq 1 - \delta \)
   » Stochastic lookahead/stochastic prog/Monte Carlo tree search
     \( X^{LA-S}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} C(\tilde{S}_t, \tilde{x}_t) + \sum_{\tilde{c} \in \Omega_t} p(\tilde{c}) \sum_{t'=t+1} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \)
   » “Robust optimization”
     \( X^{LA-RO}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} \min_{w \in W(S_t)} C(\tilde{S}_t, \tilde{x}_t) + \sum_{t'=t+1} C(\tilde{S}_{t'}, (w), \tilde{x}_{t'}) \)
An energy storage problem

Consider a basic energy storage problem:

- We are going to show that with minor variations in the characteristics of this problem, we can make each class of policy work best.
An energy storage problem

We can create distinct flavors of this problem:

» Problem class 1 – Best for PFAs
  • Highly stochastic (heavy tailed) electricity prices
  • Stationary data

» Problem class 2 – Best for CFAs
  • Stochastic prices and wind (but not heavy tailed)
  • Stationary data

» Problem class 3 - Best for VFAs
  • Stochastic wind and prices (but not too random)
  • Time varying loads, but inaccurate wind forecasts

» Problem class 4 – Best for deterministic lookaheads
  • Relatively low noise problem with accurate forecasts

» Problem class 5 – A hybrid policy worked best here
  • Stochastic prices and wind, nonstationary data, noisy forecasts.
An energy storage problem

The policies

» The PFA:
  • Charge battery when price is below p1
  • Discharge when price is above p2

» The CFA
  • Optimize over a horizon H; maintain upper and lower bounds (u, l) for every time period except the first (note that this is a hybrid with a lookahead).

» The VFA
  • Piecewise linear, concave value function in terms of energy, indexed by time.

» The lookahead (deterministic)
  • Optimize over a horizon H (only tunable parameter) using forecasts of demand, prices and wind energy

» The lookahead CFA
  • Use a lookahead policy (deterministic), but with a tunable parameter that improves robustness.
An energy storage problem

Each policy is best on certain problems

» Results are percent of *posterior* optimal solution

<table>
<thead>
<tr>
<th>Problem:</th>
<th>Problem description</th>
<th>PFA</th>
<th>CFA Error correction</th>
<th>VFA</th>
<th>Deterministic Lookahead</th>
<th>CFA Lookahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A stationary problem with heavy-tailed prices, relatively low noise, moderately accurate forecasts.</td>
<td>0.959</td>
<td>0.839</td>
<td>0.936</td>
<td>0.887</td>
<td>0.887</td>
</tr>
<tr>
<td>B</td>
<td>A time-dependent problem with daily load patterns, no seasonalties in energy and price, relatively low noise, less accurate forecasts.</td>
<td>0.714</td>
<td>0.752</td>
<td>0.712</td>
<td>0.746</td>
<td>0.746</td>
</tr>
<tr>
<td>C</td>
<td>A time-dependent problem with daily load, energy and price patterns, relatively high noise, forecast errors increase over horizon.</td>
<td>0.865</td>
<td>0.590</td>
<td>0.914</td>
<td>0.886</td>
<td>0.886</td>
</tr>
<tr>
<td>D</td>
<td>A time-dependent problem, relatively low noise, very accurate forecasts.</td>
<td>0.962</td>
<td>0.749</td>
<td>0.971</td>
<td>0.997</td>
<td>0.997</td>
</tr>
<tr>
<td>E</td>
<td>Same as (C), but the forecast errors are stationary over the planning horizon.</td>
<td>0.865</td>
<td>0.590</td>
<td>0.914</td>
<td>0.922</td>
<td>0.934</td>
</tr>
</tbody>
</table>

» … any policy might be best depending on the data.

*Joint research with Prof. Stephan Meisel, University of Muenster, Germany.*
Optimization under Uncertainty: A Unified Framework

Warren B Powell