Energy and Uncertainty: Models and Algorithms for Complex Energy Systems

Warren B. Powell

November 16, 2013

To appear in AI Magazine

Abstract

The problem of controlling energy systems (generation, transmission, storage, investment) introduces a number of optimization problems which need to be solved in the presence of different types of uncertainty. We highlight several of these applications, using a simple energy storage problem as a case application. Using this setting, we describe a modeling framework based around five fundamental dimensions which is more natural than the standard canonical form widely used in the reinforcement learning community. The framework focuses on finding the best policy, where we identify four fundamental classes of policies consisting of policy function approximations (PFAs), cost function approximations (CFAs), policies based on value function approximations (VFAs), and lookahead policies. This organization unifies a number of competing strategies under a common umbrella.

The challenge of creating an efficient, sustainable energy system requires solving design and control problems in the presence of different sources of uncertainty. There is the familiar array of decisions: discrete actions, continuous controls, and vector-valued (and possibly integer) decisions. The tools for these problems are drawn from computer science, engineering, applied math, and operations research.

The problems of energy planning, however, introduce a fresh dimension in terms of the range of different types of uncertainty: familiar Gaussian noise, heavy-tailed distributions, bursts, rare events, temporal uncertainty, lagged information processes, and model uncertainty. Randomness arises in supplies and demands, costs and prices, and in the underlying dynamics (the efficiency of gas turbines, and evaporation rates from reservoirs). A byproduct of this variability is the need to use different types of objective functions. We can minimize the expected costs, but there is growing interest in drawing on different types of risk measures as we try to capture our attitudes toward uncertainty.

The design and control of energy systems is proving to be a rich playground for specialists in the science of making decisions under uncertainty (also known as stochastic optimization). The purpose of this article is to identify some of the modeling and algorithmic challenges that arise in energy systems, and to demonstrate a way of modeling them that represents a slight departure from the classical modeling frameworks used in computer science or operations research.

The proper handling of information is arguably one of the most subtle challenges in the correct modeling of complex systems. We have well-developed notation for modeling physical processes: the operation of generators, the flow of electricity, the storage of power, and the logistics of people, equipment and fuels. Deterministic optimization models follow a small number of standard modeling

conventions, as evidenced by a substantial body of papers with models that are clear, correct and implementable.

By contrast, the communities which attempt to combine decisions and uncertainty are fragmented into a jungle of subcommunities with names such as reinforcement learning (Sutton and Barto, 1998), stochastic search (Spall, 2003), approximate/adaptive dynamic programming (Werbos 1974, Powell 2011, Bertsekas 2013), stochastic programming (Birge and Louveaux, 1997), Markov decision processes (Puterman, 2005), decision trees, optimal control (Stengel, 1994), and simulation-optimization (Swisher et al. 2003, Fu 2008). Notation for the modeling of information can be nonstandard even within a community.

Ignoring for the moment our inability to agree on the right way to write down these problems, we then face the challenge of designing algorithms to deal with the different types of uncertainty. Reliable algorithms exist only for an astonishingly narrow set of problems. The sorry state of affairs is difficult to discern from the ocean of research "proving" that algorithms work and experimental work where we are all heroes in our own papers. We focus on the challenge of finding good policies, and identify four fundamental classes of policies which help to integrate what have often been presented as competing algorithmic strategies.

Sample problems in energy systems

The process of delivering electricity requires a system comprised of a series of components, spanning generation, transmission, storage, and systems for managing loads (demands). Since electricity is a critical but difficult-to-store commodity, utilities use complex contracts to protect both consumers and themselves from the risk of high electricity prices. All of these processes interact in a complex way that benefit from the tools of stochastic optimization.

Energy storage

The problem of storing energy now to be used later arises in such a wide range of settings that it has become a foundational problem, as ubiquitous in energy systems as inventory theory is in operations management. Storage is particularly important in the context of the electric power grid, because electricity is extremely difficult to store, resulting in a variety of strategies to compensate for different types of variability (generator failures, transmission failures, and unexpected increases in temperatures on a hot day). There is tremendous interest in direct methods of storing energy from the grid, spanning pumped hydro (pumping water uphill), grid level battery storage, compressed air (pumping air under high pressure into underground cavities), flywheels and vehicle-to-grid.

Some settings include:

Battery arbitrage – Real-time spot prices of electricity on the grid will typically average around $50/MWhr (megawatt-hour), but it may range down to the 30's (and can even go negative!), but will routinely spike above $200 /MWhr, and occasionally over $500/MWhr. These prices vary by node (they are known as locational marginal prices, or LMPs), and they may be updated every 5-15

minutes. Battery arbitrage involves drawing energy from the grid when prices are low, and selling energy back when prices are high. The problem is to determine these upper and lower limits.

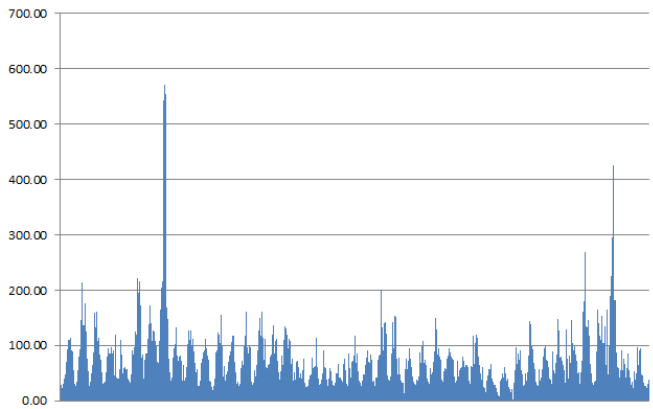


Figure 1-Electricity spot prices on the PJM grid

Battery arbitrage (energy shifting) – If your battery is large enough, you are required to bid your energy into the electricity market. Electricity markets are managed primarily by *independent system operators (ISOs)* such as PJM Interconnections, which manages the grid that covers a triangle from the Chicago area to Virginia up through New Jersey, and all the states in between (13 in total). The rules differ between ISOs, but a battery operator might have to quote lower (charge) and upper (discharge) prices either an hour ahead or even a day ahead. The optimization problem is to choose these prices to maximize net revenues while managing the energy level in the battery.

Pairing storage with wind or solar – Wind and solar are both very popular as a completely green source of energy, but they suffer from our inability to control them (a property known as "dispatchability" in the electricity community). The sun only shines during the day, and both wind and solar suffer from intermittency, creating a need for backup sources. Storage offers a way of smoothing both the intermittency as well as daily cycles.

Hydroelectric storage – Some regions are served by networks of water reservoirs, which introduces the problem of optimizing the flows between reservoirs, a problem that has long attracted the attention of the stochastic optimization communities.

Storage problems have to be solved in the presence of different types of uncertainty: uncertain generation from wind turbines and solar panels, stochastic (and heavy-tailed) real-time prices, infrequent failures of generators or transmission lines, as well as forecasts of temperatures and loads.

Commodity contracts

Utilities need to sign contracts, typically 3-5 years into the future (but sometimes longer), to supply a customer with energy at a guaranteed price. The utility needs to then purchase hedges to protect against potential spikes in prices. While spikes are generally of short duration, a lengthy heat wave in Texas in 2012 cost a utility tens of millions of dollars. Contracts can be purchased over different horizons. For this problem, we might introduce the notation $a_{tt'}$ giving us the amount we wish to purchase at time t for delivery during time period t'. Forecasts $f_{tt'}$ (e.g. of prices or demands) and decisions $a_{tt'}$ (e.g. purchasing forward contracts) are known as *lagged information and decision processes*.

Negotiating these advance contracts requires that we deal with different sources of uncertainty: the long term growth in demand, improvements in energy efficiency, structural changes in commodity prices such as gas and oil, as well as the usual spikes in prices shown in Figure 1.

Demand response

"Demand response," in the terminology of the energy community (sometimes called "demand side management" or "demand side response"), refers to the ability to reduce demand using various incentives. This might be through a real-time price or, more often, a previously arranged contract where customers are paid in return for agreeing to reduce their load on demand. These contracts may offer a consumer a lower price in return for their agreement to reduce their load on demand (possibly under direct control of the grid operator or a load serving entity).

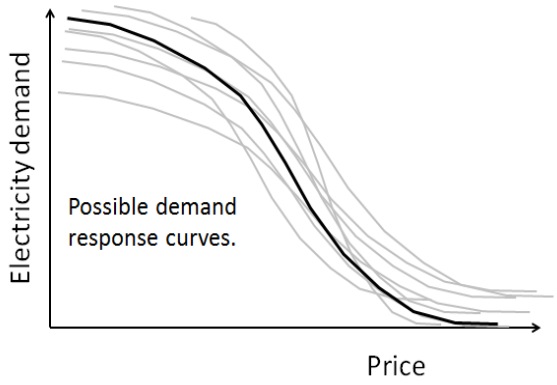


Figure 2-Uncertainty in demand response models

Demand response providers have to deal with a number of issues. First, customers often want some advance notice (two hours is typical) so they can take actions to minimize the impact of the reduction. Commercial buildings might want to air condition a lobby so the cool air can be used later. A manufacturing enterprise might want to make extra product which is then held in inventory. Both of these examples represent a form of advance storage. In other cases, demand is pushed off to later in the day (again, a different form of storage).

Second, customers may not always respond in a predictable way. Increasing the cost of charging an electric vehicle may have a bigger response on a warm and sunny day, when customers are more willing to walk, than a cold and rainy day. It may be necessary to learn a function that changes depending on conditions.

The challenge we face with demand response is that we are often dealing with people, not equipment. Not only do we not know how people (including organizations) may respond to load curtailment incentives, the response may depend on local conditions that change over time. Well known is the problem of *fatigue*, where responses diminish as repeated demands for curtailment are made. These behaviors introduce the dimension of *model uncertainty.*

Unit commitment

The unit commitment problem involves planning the power generating units (nuclear, steam plants run by coal or gas, and fast-ramping gas turbines) which are to be turned on or off, and the level at which they should be operated. Most grid operators follow PJM, which runs a day-ahead market to determine the steam generators that should be used tomorrow, and an hour-ahead market that plans which gas

turbines should be online. There is also a real-time, economic dispatch problem which can ramp a generator up and down within limits, without turning it off or on.

Unit commitment problems are hard largely because they might involve up to a hundred thousand integer variables. Fortunately, such problems can be solved using commercial solvers such as Cplex or Gurobi. However, these solvers can only be used if we view the future deterministically. ISOs have learned how to insert reserve capacity to handle the uncertainty in demand forecasts and the infrequent generator failure, but face a much more difficult challenge in designing a schedule that can handle the much higher level of uncertainty as generation from wind and solar increases.

As an example, it is not hard to recognize that if we are going to count on a significant contribution from wind or solar, we are going to have to arrange enough backup reserve to provide power when the wind drops or clouds come in. But this is not enough, especially when we are drawing energy from wind, which can exceed the forecast. If this happens, we also need to be able to ramp the generators down. If we cannot do this (which might happen if gas turbines are running at their minimum), then we cannot use the wind energy, forcing us to replace free energy which generates no CO2 with more expensive energy from natural gas which has a substantial CO2 footprint.

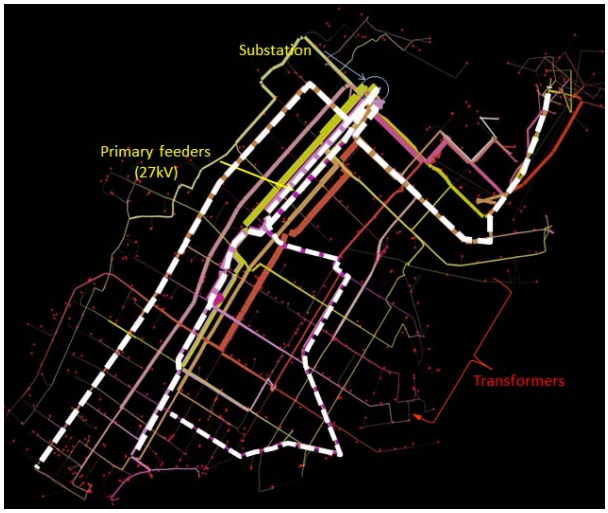


Figure 3-Primary feeders in New York, with potential failures.

A significant part of unit commitment involves protecting against generation and network failures. Figure 3 shows the primary transformers in New York City, which fail from time to time. It is important to plan generation capacity that will meet demands even in the event of these infrequent, but not rare, network failures.

Replacement of transformers

Transformers represent a critical technology in a grid. A large number of transformers were installed in the 1950's and 1960's, and are still operating today. As a result, utilities do not actually know the true failure curve. Overestimating the failure rate curve (which is common if they believed the curves provided by manufacturers were accurate) increases costs by replacing transformers unnecessarily. However, underestimating the failure rate curve exposes the utility to the risk of a burst of failures. It can take up to 12 months to order and install a replacement, and utilities would be exposed to considerable financial risk if they had to order, say, 20 transformers in a single year.

Information, forecasting and uncertainty

The examples above help to illustrate a range of different types of uncertainty. Before highlighting the different types of uncertainty, it is useful to make a few remarks about the nature of information and uncertainty.

First, all of the applications above involve some sort of manageable resources (generators, energy in storage, commodities, spare transformers, and even customers). Exogenous information can enter in the form of energy from the sun, uncertain temperatures, uncertainty in the response of customers to curtailment decisions, infrequent generator failures, and heavy-tailed spot prices.

Second, we often have access to some sort of forecast. We might let $f_{tt'}$ be the forecast of solar energy during time period t', using information available up to time t. If $E_{t'}$ is the actual energy generated at time t', it is reasonable to expect that the forecast of the demand is unbiased, which is to say

$$f_{tt'} = \mathbb{E}_t[E_{t'}] \tag{1}$$

where $\mathbb{E}_t[...]$ represents the expected value of a quantity made at time *t*. It is common to think of the energy $E_{t'}$ as random if we are at time $t < t'$, but the real uncertainty is not in $E_{t'}$, but rather the difference between $E_{t'}$ and its forecast, which we might write as

$$E_{t'} = f_{tt'} + \varepsilon_{t'}. \tag{2}$$

Figure 4 illustrates a forecast of wind (bold solid line) produced by a meteorological model called WRF (Weather Research and Forecasting). Also shown is the actual observations (dashed line), along with a number of sample realizations of the random variable $\varepsilon_{t'}$ drawn from a stochastic model estimated specifically to capture the difference between actual and forecasted wind. Note that in this plot, the variation in the forecasted wind is much more variable than the error in the forecast. It is important to distinguish predictable variability from unpredictable variability.

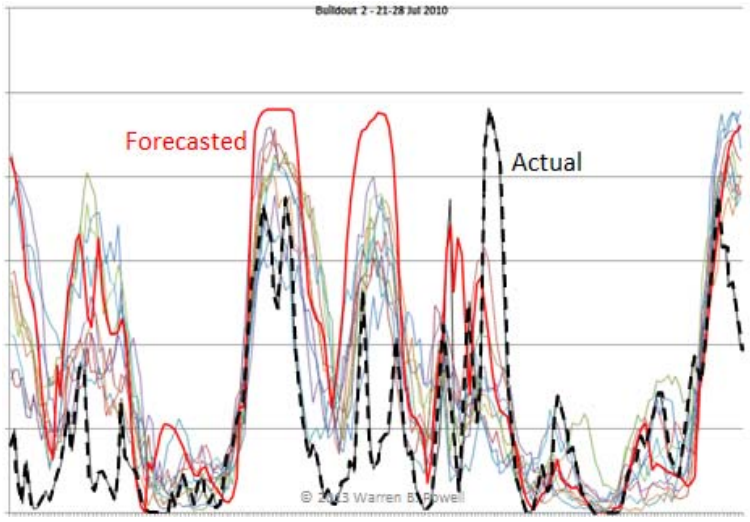


Figure 4-Actual and forecasted wind, along with sample paths representing other potential outcomes.

This application illustrates a different type of uncertainty. In the graph, the meteorological model seems to do a good job of forecasting sudden increases in wind speed, but in addition to amplitude errors (errors in the amount of wind), there also appear to be temporal errors, representing errors in predicting the *timing* of changes in the wind speed.

This discussion highlights the different types of uncertainty that arise in different applications in energy systems:

- Gaussian noise, which is used to model the familiar bell-shaped curve of the normal distribution.

- Heavy-tailed distributions, as arise in models of real-time electricity prices. In one study, electricity prices were well predicted by a Cauchy distribution with infinite variance.
- Binomial random variables with small probabilities, which would be used to model the event that a generator fails.
- Poisson distributions, which would be used to model the number of failures over a large network.
- Temporal errors, which capture errors in the *timing* of an event.
- Model uncertainty, where we may not know the structure of a model or, more often, we do not know the parameters of a model.
- Unobservable processes. We will never know the elasticity of demand in response to electricity prices, or the true load on the network which is influenced by activities that are outside of the range of our grid (but still connected). Instead, we will have to represent our understanding of these parameters through a probability distribution.

We also have to distinguish between problems where we have a reasonable estimate of the probability distribution of the uncertainty, and problems where we do not. If we do not know the probability distribution, we may assume that we are working in a setting where we can observe the random events. The second case is one way that "model-free" applications arise, although this is a term that is used differently in different communities.

Case application: Analysis of a solar-storage system

There is growing interest in pairing solar arrays with battery storage on the grid, where the battery can be used to smooth the variations in the solar energy (which can be quite spikey due to clouds). The battery can be used for multiple purposes. A major use is *frequency regulation* (smoothing out the high frequency deviations from the standard 60 Hz signal for electricity), for which there is an established market which pays about $30 per hour, per megawatt of power (a megawatt is the *rate* at which power moves over the network). Another use is known as *battery arbitrage* (also referred to as *energy shifting*). Battery arbitrage arises because the price of electricity (known as a *locational marginal price*, or LMP), changes every five minutes (on the PJM grid – this frequency of update varies from one ISO to another). LMPs typically range between $30/MWhr (but can go lower), up to $1000/MWhr, averaging about $50/MWhr. A battery can be used to store electricity when prices are low, and sell it back when prices are high (hence the name "energy shifting").

Managing energy storage requires a policy that can react to price spikes, while thinking about the timing of when solar energy will be generated, and when it is needed. A sample configuration is illustrated in figure 5. Forecasts are often available for both generation and loads (which tends to depend heavily on temperature). We have to decide whether our solar array should meet current demands of a building, sell to the grid, or store in a battery.

Figure 5-Illustration of storage device balancing energy from wind and grid to meet a time-varying load.

We are going to use this setting to illustrate some principles of modeling and the design of control policies.

The challenge of modeling

It is very common in the reinforcement learning community to represent stochastic dynamic problems as consisting of a state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r(s,a)$, a discount factor γ, and a one-step transition matrix $\mathcal{P}$ with element $p(s'|s,a)$. Implicit in the representation is that the problem is stationary. Such a representation would be called "model-based" in computer science because we assume we know the transition matrix $\mathcal{P}$ (known in this community as the "model"). It is widely recognized that the transition matrix is not known for many applications. Such models are known as "model-free" and it is assumed that if we are in a state s^n and take action a^n, that it is possible to observe from an external source the next state s^{n+1} and an associated reward $r(s^n,a^n,s^{n+1})$. This line of thinking overlooks the many applications where the dynamics are known, but the transition matrix is not computable.

Such problems are often "modeled" as dynamic programs by writing Bellman's optimality equation, given by

$$V(s) = \min_{a\in\mathcal{A}}\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)V(s')\right). \quad (3)$$

Of course, Bellman's optimality equation is not a model at all, but rather a characterization of an optimal policy. The problem is that it is also well known to be computationally intractable for all but the smallest problems. The problem is most commonly described as *the curse of dimensionality*, which refers to the explosion of the state space as additional dimensions are added. In reality, for many problems there may be up to three "curses" of dimensionality, which we discuss below.

As problems become more complex, we have to put more attention into how we *model* these problems. There is a tendency to quickly put problems into a canonical framework and then focus on algorithms. However, the standard form can prevent us from addressing important dimensions of the problem.

Below we offer a simple framework that divides modeling into five components: states, actions, exogenous information, the transition function and the objective function.

The state of the system

Perhaps one of the most important quantities in a dynamic model is a state variable, but try to find a formal definition. Classic texts such as Bellman [1957] and Puterman [2005] avoid a definition with statements such as "there is a *state* of the system." Wikipedia offers "State commonly refers to either the present condition of a system or entity," which is true but hopelessly vague, or "A state variable is one of the set of variables that are used to describe the mathematical 'state' of a dynamical system" (using the word you are trying to define in the definition is a red flag that you do not have a definition). Our own definition (from Powell [2011]) is that a state variable is the "minimally dimensioned function of history that is necessary and sufficient to calculate the decision function, cost/reward function, and (if available) the transition function." In other words, all the information you need to model the information from time *t* onward, and only the information that is needed.

Even this definition is so general that a student modeling a complex system might overlook important elements. It helps to describe three increasing sets of variables:

- The physical (or resource) state R_t. This might be the energy in a battery, the state of a generator, or the status of a component such as a transformer that might fail.
- The information state I_t, which (in addition to including R_t) might include the price of electricity, the current temperature, the wind speed, or the price of natural gas.
- The knowledge (or belief) state K_t. In addition to known information captured in I_t, the knowledge state captures distributional information about any unknown parameters.

A common mistake is to assume that the state of the system is given by the physical state which is often (but not always) the only controllable dimension(s). In some applications, we need to include information that was first revealed in the past, leading some authors to describe these as *history-dependent* (that is, non-Markovian) systems. For example, we might need to know the wind speed over the past three hours to predict the wind speed in the next hour. Just because we first learned the wind speed three hours ago does not make it any different (as a piece of information) from the wind speed that we just learned now. But if we do not need the wind speed four hours ago, then it is not part of the (information) state.

The most subtle part of a state variable is the knowledge (belief) state, which is how we describe unobservable parameters. We might learn about how the market responds to a price signal, but only by varying the price and learning (imperfectly) the price response coefficient. In this setting, we may wish to try different prices to learn the price more efficiently, but this would involve a classic exploration/ exploitation tradeoff familiar to the bandit community.

To illustrate using our solar-storage case application, let R_t be the energy stored in the battery, and let E_t be the energy generated from the solar array over *t-1* to *t*. Let p_t be the price of electricity on the

grid (known as a *locational marginal price* or LMP) at time *t,* which depends on the temperature Q_t at time *t* (LMPs tend to be both higher, and a lot more volatile, at very low and very high temperatures). The flow of energy into and out of the battery is governed by *bids* $b_t = (g_{t-1,t}, h_{t-1,t})$, made at time *t-1*, to be implemented between *t* and *t+1.* The battery charges (buys from the grid) if $p_t < g_{t-1,t}$, and discharges (sells back to the grid) if $p_t > h_{t-1,t}$. According to market rules, the bids $b_t = (g_{t,t+1}, h_{t,t+1})$ were set at an earlier time (for simplicity, assume they were set at *t-1*), which means that at time *t*, b_{t-1} is part of our (information) state. Finally, we recognize that temperature can be forecasted with some degree of precision. Let $f_{tt'}$ be the forecast of the temperature $Q_{t'}$ at time *t'* given what we know at time *t* (note that $f_{tt} = Q_t$). We can roll this into a single variable $f_t = \left(f_{tt'} \right)_{t' \geq t}$. Pulling all this together gives us our state variable $S_t = \left(R_t, E_t, p_t, f_t, b_t \right)$.

The recognition that a forecast is part of the state variable seems to be quite recent. However, we can also argue that a forecast is really a part of our probability model of future events. Technically, if this probability model is changing over time (as forecasts are updated), then this probability model should also be part of the state variable. But the probability model can also be viewed as a *latent variable*, one that is used in calculating the expectation, but which is not explicitly represented in the state variable.

Decisions

Decisions come in different shapes and sizes: discrete actions, continuous scalars, and continuous or discrete vectors of arbitrary size. The reinforcement learning community works primarily with discrete actions, denoted *a*. The engineering controls community tends to work with low-dimensional continuous vectors denoted *u*. Finally, the operations research community often works with very high dimensional vectors denoted by *x*, typically in the context of convex resource allocation problems.

An important complication that arises in many applications involves *lagged* decisions. For example, we might have to purchase electricity contracts in month *t* to deliver electricity in month *t'*. Alternatively, we might have to request at hour *t* that a commercial building operator reduce its air conditioning in hour *t'* (perhaps four hours in the future). We can represent these decisions as $a_{tt'}$, or as a vector $a_t = \left(a_{tt'} \right)_{t' \geq t}$. Using this notation, we see that these lagged decisions are very similar to our forecasts (and work in a similar way, except that we control them).

When writing the model, it is important to define decisions but defer until later how a decision will be made. Standard modeling practice is to define an arbitrary function $\pi(s)$, referred to as a policy, that returns an action *a* (or control *u* or decision *x*) given a state *s*. In practice, researchers tend to quickly adopt a particular class of functions to represent the policy, which can artificially narrow their search for a solution. For reasons that will become clear, we are going to let π represent the structure of the function used. Then, we let $A^\pi(s)$ be the function (policy) that returns an action *a* if we are in state *s*. We can use $X^\pi(s)$ or $U^\pi(s)$ for our policies if we are using decision *x* or control *u*.

For our solar-storage case application, the decision variable would be $a_t = \left(x_t, b_{t,t+1}\right)$, where

$$x_t = \left(x_t^{GB}, x_t^{SB}, x_t^{SG}\right)$$

is the energy flows from grid to battery (or back), solar panel to battery, and solar panel to grid, respectively, while $b_{t,t+1} = (g_{t,t+1}, h_{t,t+1})$ represents the bids we make at time *t* that will be used at time *t+1*. Below, we let $A^{\pi}(S_t)$ be the function (policy) that returns the action a_t when we are in state S_t.

Exogenous information

The reinforcement learning community often avoids explicit models of exogenous information by either assuming that the one-step transition matrix is given (the exogenous information is buried in this matrix), or by assuming the application is "model free" which means that exogenous information is not observable.

The exogenous information process provides the information we need to describe the changes in the state variables. There is very little in the way of standard notation for modeling exogenous information, we have adopted the style of putting hats on exogenous information variables. We illustrate our style using our solar-storage application. For this system, the exogenous information comes in the form of changes in LMPs, which we denote by $\delta p_t = p_t - p_{t-1}$, changes in the energy from the solar panel, δE_t, and changes in the forecasts of future temperature, δf_t. We let W_t be our generic exogenous information variable, which would be $W_t = \left(\delta p_t, \delta E_t, \delta f_t\right)$ for this application. If we were modeling storage over the grid, $\delta p_t = \left(\delta p_{ti}\right)_{i \in \mathcal{I}}$ where δp_{ti} is the change in the LMP at node ("bus") *i*. Thus, W_t can have thousands of dimensions.

Transition function

The concept of a transition function is widely used in the engineering controls community, but it is a term that is rarely heard in either the computer science or operations research communities. The transition function consists of a series of equations that describe how the state variables change over time.

For our solar-storage application, let η be the round trip efficiency (typically around 0.80). If we let $A = (\eta, \eta, 0)^T$, then $A^T x_t$ is the net flow into or out of the battery. Also, between *t* and *t+1* there will be exogenous changes to the storage due to the interaction between the available energy from the solar panel E_{t+1}, the LMPs p_t, and the decision b_t. To keep the model simple, we represent this change using the random variable δR_{t+1} which depends on the bid b_t as well as the observations of the change in the LMP δp_{t+1} and change in the solar energy δE_{t+1}. Our transition function is then

$$R_{t+1} = R_t + A^T x_t + \delta R_{t+1}, \tag{5}$$

$$p_{t+1} = p_t + \delta p_{t+1}, \tag{6}$$

$$E_{t+1} = E_t + \delta E_{t+1}, \tag{7}$$

$$f_{t+1,t'} = f_{t,t'} + \delta f_{t+1,t'}. \tag{8}$$

These equations capture, respectively, the evolution of storage (the only controllable state), followed by the stochastic processes describing energy from wind, the loads (demands) on the grid, and the updating of forecasts. The variables with hats represent exogenous information that first became known at time *t+1.*

The transition function goes by many names in the literature: transfer function, system model, plant model, or simply "model." While the function $f(...)$ is often used for the transition function, we use the notation

$$S_{t+1} = S^M(S_t, a_t, W_{t+1})$$

to represent our transition function (or "system model"). Note that at time *t*, the state S_t is known, the action a_t is computable (using our policy $A^\pi(S_t)$), but the new information W_{t+1} is random at time *t* (it will be known at time *t+1*).

This notation allows us to describe the second curse of dimensionality that we introduced earlier. Let's consider what is involved in computing the one-step transition matrix. While it is convenient to assume that the transition matrix might simply be given, computing it requires solving the equation

$$p(s' \mid s, a) = \mathbb{E}1_{\left\{s'=S^M(s,a,W)\right\}}, \tag{9}$$

where $1_{\{E\}}$ =1 if the event E is true, and the expectation is over the random variable W (or W_{t+1}). If we are modeling a network with 100 storage devices, W_{t+1} would have 300 dimensions. Calculating equation (9), then, means computing a 300-dimensional integral (or summation). It is easy to overlook the need to compute an expectation when calculating the one-step transition function. In fact, a more natural way of writing Bellman's equation is to use its expectation form

$$V(S_t) = \min_{a \in \mathcal{A}} \left(r(S_t, a) + \gamma \mathbb{E}\left\{ V(S^M(S_t, a, W_{t+1})) \mid S_t \right\} \right). \tag{10}$$

Now we see the expectation explicitly, and we can understand that if our exogenous information process is multidimensional, then we are going to have a problem calculating this expectation. This is the second curse of dimensionality. Finally, if a_t is a vector, then enumerating a set of discrete actions to solve the maximization problem introduces the third curse of dimensionality.

The objective function

Our last step involves writing the objective function. Let $C(s,a)$ be the cost (contribution if we are maximizing) if we are in state *s* and take action *a*. If we fix our policy with the function $A^{\pi}(S_t)$ when we are in state S_t, we can write our objective function over a finite horizon 0, …, T as

$$\min_{\pi\in\Pi} \mathbb{E}F^{\pi}(S_0,W) = \mathbb{E}^{\pi}\sum_{t=0}^{T}\gamma^t C(S_t,A^{\pi}(S_t)), \tag{11}$$

where $S_{t+1} = S^M(S_t,A^{\pi}(S_t),W_{t+1})$ governs how we evolve from state S_t to S_{t+1}. In virtually all practical applications, equation (11) is calculated using simulation. We let $(W_1,W_2,...,W_t,...)$ represent a Monte Carlo sample realization of all the exogenous information. There are, of course, many possible sample realizations. We can either use one very long simulation, or take an average over a number of simulations.

For some applications, the policy $A^{\pi}(S_t)$ is stationary. While stationary policies (and infinite horizon problems) are quite popular in the reinforcement learning community, applications in energy systems (in our experience) seem to usually require some sort of time-dependent policy (the policy itself is a function of time, and not just a function of a time-dependent state). Time-of-day, day-of-week and seasonal patterns tend to be present across applications, and these tend to require the use of time-dependent policies.

The use of expectations is so widespread that we tend to use them blindly as a kind of default objective function. Expectations represent the correct operator when we want to minimize (or maximize) our average performance. However, there are many applications where we have to pay serious attention to risk. For example, in the unit commitment problem, we have to worry about the risk that we do not have enough generating capacity to cover the network load. Also, a supplier signing a contract to deliver energy at a fixed price in the future has to worry about the risk that electricity might cost more than the contract. If we want to depend on high penetrations of wind and solar, we have to worry about the risk that there will not be enough energy available from these intermittent sources. And if we are interested in investing in solar while counting on the value of solar renewable energy certificates (a market-driven form of subsidy), investors will worry about the possibility that the value of these certificates might drop from year to year.

A relatively simple approach to handling risk is to replace the expectation with a quantile. If *W* is a random variable, let $Q_{\alpha}(W)$ be the α quantile of *W*. Equation (11) now becomes

$$\min_{\pi\in\Pi} Q_{\alpha}F^{\pi}(S_0,W) = Q_{\alpha}\sum_{t=0}^{T}\gamma^t C(S_t,A_t^{\pi}(S_t)). \tag{12}$$

This simple change is actually not so simple. First, the quantile operator is no longer additive, which prevents us from using the Bellman optimality equation. Second, it is much harder to estimate the quantile of a function using standard Monte Carlo methods than it is to estimate the mean.

The finance community has developed a rich theory around risk measures, especially a class known as *coherent risk measures* which enjoy certain properties that are important in finance. Coherent risk measures enjoy properties such as convexity (with respect to the costs), monotonicity (higher costs mean higher risk), translationary invariance (adding a constant does not change the risk), and positive homogeneity (multiplying the costs times a scaling factor is the same as multiplying the risk times the same scaling factor). For more on this topic, see Shapiro et al. (2009) and Ruszczynski (2010). In our work, we have found that we might worry about the risk of running out of energy or losing money on a contract; such risk measures are not coherent, but rather fall in a category we are calling *threshold risk measures* (see Collado and Powell 2013). Considerable attention is also being given to worst-case performance, sometimes referred to as *robust optimization* (for an excellent introduction to the field of robust optimization, see Ben-Tal et al. 2009). The general field of stochastic optimization using risk measures or "robust" objectives is quite young, but with tremendous applications in energy.

The biggest challenge with finding the best policy, whether we use equation (11), (12) or any other risk measure, is determining what is meant by a search over policies. There is a fairly wide range of communities working in the general area of making effective decisions over time under uncertainty. Part of the diversity is explained by differences in problems addressed by different communities. But a contributing factor is the lack of a standard vocabulary for describing problems, and there seems to be a surprising misunderstanding in terms of what is meant by the word "policy." We address this issue next.

Policies

If we were solving a deterministic problem, our challenge would be to find an optimal set of *decisions*. When we work on stochastic problems, the challenge is to find a set of *functions* which are known in the literature as policies. Since Bellman's seminal work in the 1950's on dynamic programming, there has been a widely shared presumption that this means solving Bellman's optimality equation (equations (3) or (10)). While this works well on a small class of problems, it has proven to be astonishingly difficult for most real applications, and certainly those that we have encountered in energy applications. As a result, different communities have identified a variety of strategies to solve different problem classes, a situation we have come to refer as the *jungle of stochastic optimization*, illustrated in figure 6. Our conclusion, however, is that the range of strategies is not as diverse as it seems, since similar policies are often disguised by differences in presentation style and notation.

There seems to be some confusion about the meaning of the term "dynamic program." Dynamic programming is often equated with Bellman's optimality equation. Actually, a dynamic program is a sequential decision problem, as depicted in equation (11). Bellman's optimality equation is a) a characterization of an optimal policy, and b) a foundation for finding a policy. A policy, on the other hand, is a mapping from a state to an action…. *any mapping.* A policy, for example, may require solving a decision tree, or a linear (or integer) program.

The search for an optimal (or good) policy can seem like an impossible task. How in the world are we supposed to search over some arbitrary class of functions? This seems even more hopeless when our decisions involve hundreds or thousands of dimensions (as occurs in the stochastic unit commitment problem) and have to obey a complex set of constraints. It turns out that a good guide is to look at the range of tools that people are already using.

Figure 6-The jungle of stochastic optimization

The four classes of policies

Our own experience stumbling around the jungle of stochastic optimization for several decades has led to the conclusion that all the different algorithmic strategies can be boiled down into four fundamental classes:

1) Policy function approximations (PFAs) – These are functions that return a state given an action, without solving an imbedded minimization or maximization problem.
2) Myopic cost function approximations (CFAs) – This covers policies where we minimize a single period cost function, usually modified in some way to achieve better long term performance. In rare cases, minimizing a single period cost may be optimal.
3) Policies based on value function approximations (VFAs). This is the class of policies most often associated with dynamic programming, where we optimize a one-period cost/contribution plus an approximation of the value of future costs/contributions given the downstream state.
4) Lookahead policies – This covers the entire class of policies where we optimize over a finite horizon *H*, but then just implement the first period decision. Lookahead policies include rolling/receding horizon procedures, decision trees, rollout heuristics, Monte Carlo tree search, model predictive control, and stochastic programming.

The four classes of policies can be briefly summarized as PFAs, CFAs, VFAs, and lookahead policies. A more detailed summary of these policies, illustrated using energy applications, follow.

Policy function approximations represent the class of policies that are described by an analytic function of some form. PFAs come in a variety of forms:

- Lookup tables (wear a coat if it is raining).
- Parametric models, such as $A^{\pi}\left(S_t \mid \theta\right) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2$.
- Neural networks.

- Nonparametric functional approximations.

In our energy application, the decision to charge or discharge a battery is governed by the rule: "charge if $p_t < g_t$ or discharge if $p_t > h_t$" which is a form of PFA. PFAs are the only class of policy that do not require solving a minimization problem.

A policy based on a **myopic cost function approximation** might be written

$$A_t^\pi(S_t) = \arg\min_{a \in \mathcal{A}_t(\theta)} C^\pi(S_t, a \mid \theta) \tag{13}$$

where $C^\pi(S_t, a \mid \theta)$ is a modification of the original cost function $C(S_t, a)$ designed to achieve better long term performance. The vector θ represents parameters that can be tuned to improve the performance of the policy, and may affect the cost function itself or the constraint set $\mathcal{A}_t(\theta)$. For our energy application, the set $\mathcal{A}_t(\theta)$ would capture the flow conservation constraints on energy flows as well as logical limits on bids. Cost functions are widely used in engineering practice, but they have not received the respect they deserve. In the energy setting, a cost function is often used for the *economic dispatch problem*, which is a linear program that is allowed to adjust generators up or down (but cannot turn them on or off) to meet the current set of demands in real-time.

Policies based on **value function approximations** would be written

$$A_t^\pi(S_t \mid \theta) = \arg\min_{x_t} \left(C(S_t, a_t) + \gamma \mathbb{E}\left\{ \bar{V}_{t+1}(S^M(S_t, a_t, W_{t+1})) \mid S_t \right\} \right), \tag{14}$$

where we have replaced the value function with some sort of approximation. For many problems, it is useful to define the *post-decision state variable* S_t^a which is the state at time *t* immediately after a decision has been made (see Powell(2011)[Chapter 4]). For example, in our energy storage problem we introduced earlier, the post-decision state would be

$$S_t^a = \left(R_t^a, p_t, E_t, f_t, b_{t-1}, b_t \right)$$

where

$$R_t^a = R_t + A^T x_t .$$

We have to augment the state variable to reflect that we not only have the bids b_{t-1} made at time *t-1*, but also the bids b_t that we just chose (to be used at time *t+1*). Note that the post-decision state is a deterministic function of S_t and a_t, allowing us to write our VFA-based policy as

$$A_t^\pi(S_t \mid \theta) = \arg\min_{a_t} \left(C(S_t, a_t) + \bar{V}_t^a(S_t^a) \right). \tag{15}$$

The policy π determines the structure of the value function approximation. For example, it is very popular to use a linear model with a predefined set of basis functions, which we would write as

$$A_t^\pi(S_t \mid \theta) = \arg\min_a \left(C(S_t, a_t) + \sum_{f \in \mathcal{F}} \theta_f \phi_f(S_t^a) \right), \tag{16}$$

where the basis functions $\phi_f(s)$ represent the type of policy, while the regression parameters θ_f are the tunable parameters.

Lookahead policies are widely used in practice. In their most popular form, a deterministic approximation of the future would be used, producing a policy of the form

$$A_t^\pi(S_t \mid \theta) = \arg\min_{(a_t, a_{t+1}, \ldots, a_{t+H}) \in \mathcal{A}(\theta^A)} \sum_{t'=t}^{t+\theta^H} C(S_{t'}, a_{t'} \mid \theta^C) \tag{17}$$

where the lookahead horizon is represented as one tunable parameter θ^H, but where the cost function or constraints can also depend on tunable parameters (θ^C and θ^A, respectively). There is a large community which attempts to build uncertainty into the lookahead model. In computer science, this falls under the umbrella of Monte Carlo tree search, designed for discrete action spaces. The operations research community, with its focus on vector-valued decisions, uses the framework of *stochastic programming* which approximates future uncertainties by building a scenario tree of sampled outcomes (not unlike Monte Carlo tree search); see Sen and Higle (1999) for an easy-to-read tutorial. The difference is that the scenario tree is constructed in advance. In this setting, θ would be the set of parameters that govern how the scenario tree is generated. Lookahead policies are the only class of policy that does not require some sort of functional approximation. While this can be quite attractive, the cost is that this is a relatively brute force approach, and is computationally the most difficult.

In addition to this list of policies, it is possible to create a wide range of hybrids. For example, we might combine a deterministic lookahead policy with a value function approximation, or a lookahead policy with a cost function approximations (with modified costs and constraints). Lookahead policies or policies based on value function approximations can also be combined with low-dimensional policy function approximations.

We note that three of our policies involve some sort of function approximation, whether it be the cost function, the policy function approximation, or the value function approximation. Functions can be approximated using three broad strategies: lookup tables, parametric functions (these may be linear or nonlinear in the parameters), and nonparametric functions. In practice, lookahead policies are often solved using a form of (parametric) cost function approximation, which typically enter the problem in the form of bonuses, penalties and modified constraints (such as reserve capacity).

Lookahead policies combined with cost function approximations are popular in planning electricity markets. Grid operators use large optimization models to determine which generators to turn off or on (the "unit commitment problem") by planning over perhaps a 48 hour horizon (the lookahead part). Built into this model are constraints to ensure that we have sufficient reserve, where we might require that total generation is $(1+\theta)$ times the forecasted load (this is the cost function approximation part).

We note that *all* of the policies above depend on some type of tunable parameters, which we represented using θ. Once we fix the class of policy, we can tune the parameters using

$$\min_{\theta} \mathbb{E}^{\pi} \sum_{t=0}^{T} C(S_t, A_t^{\pi}(S_t \mid \theta)). \tag{18}$$

In general, we cannot compute the expectation exactly. As a result, we have to turn to the large field of stochastic search algorithms.

Choosing a policy

The list of policies above transforms the vague search over policies in equation (11) into a much more concrete process of identifying specific classes of policies, and then optimizing the tunable parameters. We have used *all* of these policies in our energy systems research.

So how do we choose between these? Not surprisingly, it all depends on the structure of your policy. Some guidelines from our own empirical work include:

- Policy function approximations are best when the structure of the policy is obvious. For example, we might want to change a battery when prices are below one number, and discharge when it is above a higher number. This structure is simply obvious (or may even be specified by regulations). When this is the case, we only have to search for the best values of the tunable parameters.
- There are problems where minimizing single-period costs are optimal, and others where it is a good initial heuristic. For example, our utility may want to assign repair resources to jobs at least cost, but this might ignore a job that requires moving a long distance. We can introduce a bonus for covering jobs based on how long they have been waiting.
- Value function approximations work well when you feel that approximating the value of being in a future state is relatively simple. This does not mean the problem is small. We have successfully used VFAs to optimize large transportation systems. VFAs seem to work especially well for energy storage problems, and problems where the value of being in a state can be approximated in a straightforward way using standard statistical tools. High dimensionality is not an issue if you are willing to live with separability.
- If you have some type of forecast (prices, loads, weather), then a lookahead policy is going to be a natural choice. The question is: can you use a deterministic approximation of the future, or do you need to model uncertainty explicitly in the lookahead model? This is not an obvious choice,

and solving a stochastic lookahead model can be quite hard (basically, it is a stochastic, dynamic program, but typically on a somewhat simplified problem). Deterministic lookahead models can sometimes provide good solutions for stochastic models.

- Often, a deterministic lookahead policy may work well, but we are concerned about robustness. However, achieving a more robust solution may be fairly obvious. For example, standard industry practice with the unit commitment problem is to require reserve capacity to account for possible network failures. We would call this a hybrid lookahead/CFA policy.

It is generally best to use a policy with a min or max operator (that is, anything but PFAs) if your decision is a vector that has to satisfy a set of constraints. These problems can generally be solved using some sort of math programming solver such as Cplex, Gurobi, CVX or LOQO which is designed to handle constraints. PFAs work well when we have a good understanding of the structure of the policy, but this generally only happens for very low dimensional decisions. However, it is possible to combine a PFA with an optimization-based policy by including the PFA in the objective function in the form of a penalty term for deviating from what the PFA feels would be best for a particular variable. For example, our economic dispatch model (which minimizes cost to meet demand, a form of CFA) may wish to discharge energy to the grid because of a high price, but our battery may be close to the point where further discharge will degrade its lifetime, requiring that we deviate from our guidelines for discharging the battery (a form of PFA).

Closing notes

The applications in energy systems open up a wide array of challenges that require making decisions in the presence of different sources of uncertainty. The thoughts in this article are designed to bring together the different communities working in stochastic optimization. The contributions of the reinforcement learning community in the solution of problems with discrete action spaces with nonconvex objective functions would benefit from the skills of the operations research communities (approximate dynamic programming, stochastic programming, simulation-optimization) that are familiar with vector-valued decisions with convex objective functions.

References

Ben-Tal, A.; Ghaoui, L. El; and Nemirovski, A. 2009. *Robust Optimization*. Princeton NJ: Princeton University Press.

Bertsekas, D. P. 2012. *Dynamic Programming and Optimal Control, Vol. II, 4th Edition: Approximate Dynamic Programming*. Belmont, MA: Athena Scientific.

Birge, J. R.; and Louveaux, F. 1997. *Introduction to Stochastic Programming*. *Springer*. New York: Springer Verlag.

Collado, R.; and Powell, W.B. 2013. "Threshold Risk Measures for Dynamic Optimization," Working paper, Department of Operations Research and Financial Engineering, Princeton University, Princeton NJ.

Fu, M. C. 2008. What You Should Know About Simulation and Derivatives. *Naval Research Logistics*, *55*(8), 723–736.

Lagoudakis, M. G.; and Parr, R. 2003. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Pham, V. 2013. *Experimental Analysis of Machine Learning Models for Approximate Policy Iteration with Optimal Benchmarks*, Master's Thesis, Department of Operations Research and Financial Engineering, Princeton University.

Powell, W. B. 2011. *Approximate Dynamic Programming: Solving the curses of dimensionality* (2nd ed.). Hoboken, NJ: John Wiley & Sons.

Powell, W. B.; Simao, H. P.; and Bouzaiene-Ayari, B. 2012. Approximate Dynamic Programming in Transportation and Logistics: A Unified Framework. *EURO Journal on Transportation and Logistics*, *1*(3), 237–284. doi:10.1007/s13676-012-0015-8

Puterman, M. L. 2005. *Markov Decision Processes* (2nd ed., Vol. 7pp). Hoboken, NJ: John Wiley and Sons.

Ruszczynski, A. 2010. Risk-Averse Dynamic Programming for Markov Decision Processes. *Mathematical Programming*, *125*(2), 235–261. doi:10.1007/s10107-010-0393-3

Salas, D. F.; and Powell, W. B. 2013. Benchmarking a Scalable Approximate Dynamic Programming Algorithm for Stochastic Control of Multidimensional Energy Storage Problems. Working Paper, Department of Operations Research and Financial Engineering, Princeton, N.J.

Scott, W. R.; Powell, W.B.; and Moazeni, S. 2013. Least Squares Policy Iteration with Instrumental Variables vs . Direct Policy Search : Comparison Against Optimal Benchmarks Using Energy Storage. Working Paper, Department of Operations Research and Financial Engineering, Princeton, N.J.

Sen, S.; and Higle, J. L. 1999. An Introductory Tutorial on Stochastic Linear Programming Models. *Interfaces*, *29*(2), 33–6152.

Shapiro, A.; Dentcheva, D.; and Ruszczynski, A. 2009. *Lectures on Stochastic Programming: Modeling and Theory*. *SIAM, Philadelphia, PA*. Philadelphia: SIAM.

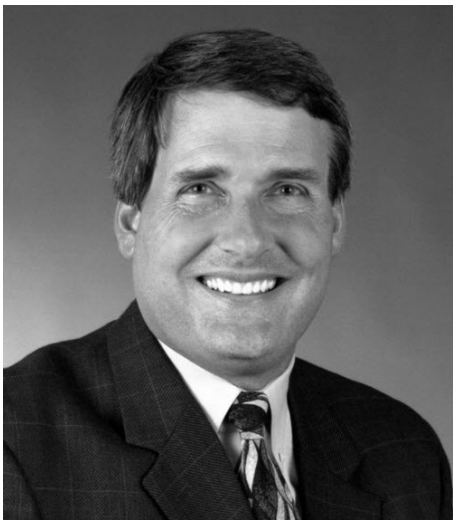
Stengel, R. F. 1994. *Optimal Control and Estimation*. Dover Publications, New York.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement Learning*. *An Introduction. MIT Press, Cambridge, MA. Tsitsiklis, J. N* (Vol. 35). Cambridge, MA: MIT Press.

Swisher, J. R.; Jacobson, S. H.; Yücesan, E.; and Yucesan, E. 2003. Discrete-Event Simulation Optimization Using Ranking, Selection, and Multiple Comparison Procedures: A Survey. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *13*(2), 134–154.

Tsitsiklis, J. N.; and Van Roy, B. 1997. An Analysis of Temporal-Difference Learning with Function Approximation. *IEEE Transactions on Automatic Control*, *42*(5), 674–690.

Werbos, P. J. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University.



Warren Powell is a professor in the Department of Operations Research and Financial Engineering at Princeton University, where he has taught since 1981. He is director of CASTLE Labs (http://www.castlelab.princeton.edu) which specializes in computational stochastic optimization and learning, with applications in energy, transportation, health and finance. He has over 190 refereed publications, and is the author of *Approximate Dynamic Programming: Solving the curses of dimensionality* and (with Ilya Ryzhov) *Optimal Learning*. He is a Fellow of Informs, where he has served in numerous leadership and service roles.