Tutorial on Stochastic Optimization in Energy I: Modeling and Policies

Warren B. Powell, *Member, IEEE,* Stephan Meisel

***Abstract*—There is a wide range of problems in energy systems that require making decisions in the presence of different forms of uncertainty. The fields that address sequential, stochastic decision problems lack a standard canonical modeling framework, with fragmented, competing solution strategies. Recognizing that we will never agree on a single notational system, this two-part tutorial proposes a simple, straightforward canonical model (that is most familiar to people with a control theory background), and introduces four fundamental classes of policies which integrate the competing strategies that have been proposed under names such as control theory, dynamic programming, stochastic programming and robust optimization. Part II of the tutorial illustrates the modeling framework using a simple energy storage problem, where we show that, depending on the problem characteristics, *each* of the four classes of policies may be best.**

***Index Terms*—Stochastic optimization, dynamic programming, approximate dynamic programming, reinforcement learning, optimal control, stochastic programming, robust optimization, energy systems**

I. Introduction

THE modeling of deterministic optimization problems has long enjoyed relatively standard modeling frameworks developed within the fields of mathematical programming and optimal control. However, the introduction of uncertainty has produced a fragmented set of competing communities, typically with different notational systems and modeling conventions with the result that there is often an astonishing lack of precision in the description of problems. Even more problematic is the range of solution approaches that are frequently presented as competing methods, hiding what can be complementary strategies. Our goal is to synthesize, in an energy setting, communities that work under names such as dynamic programming, stochastic programming, robust optimization, and optimal control (to name the most important for our setting).

Uncertainty pervades optimization problems in the power sector. A few examples include

- Unit commitment - ISOs have to handle uncertainty from outages, weather variations, and the growing integration of wind and solar energy which can be highly unpredictable [1], [2], [3]. Unit commitment problems may be made a day ahead (for steam generators) or shorter horizons (15-60 minutes) for gas turbines. Even economic dispatch, which involves the near real-time adjustment of generators, has make adjustments which anticipate future events.
- Energy storage - Storage devices can be used for frequency regulation and energy shifting, and have to be managed in the presence of stochastic frequency regulation signals, volatile electricity prices (LMPs), energy variations from wind and solar, all in the presence of predictable variations in both load and generation.
- Bidding energy resources - It is often necessary to place bids for generation or storage into day-ahead or hour-ahead markets, managing uncertain electricity prices.
- Equipment replacement - Utilities need to plan the inspection and replacement of utility poles and transformers in the presence of uncertainty in the state of the resources and unpredictable failures.
- Pricing electricity contracts - A utility may have to sign contracts to deliver electricity for 3-5 years into the future while managing uncertainty in commodity prices, grid congestion as well as future energy investments.
- Investment planning - Utilities have to make decisions about future investments in generation and transmission, in the presence of uncertainty about future loads, commodity prices, and the decisions of other generators.

These examples illustrate the wide variation in the types of stochastic optimization problems in terms of the nature of the decisions (discrete or continuous, scalar or vector), the uncertainties (binomial failures, Gaussian noise in weather, loads and generation, heavy-tailed electricity prices) and the dynamics (we may have known models of storage processes, but unknown models of climate change, commodity prices and the behavior of competing utilities). Making decisions under uncertainty pervades the planning and operation of our energy system, which can be seen in the wide array of research articles addressing these problems.

Research in this area, however, is plagued by a frustrating lack of the kind of common canonical framework that is enjoyed by deterministic optimization. Authors draw guidance from competing fields under names such as stochastic control [4], dynamic programming (Markov decision processes) [5], stochastic programming [2] or robust optimization [6], with a variety of related fields using names such as reinforcement learning [7], approximate dynamic programming [8], and stochastic search [9] (and this is an incomplete list). Recently, we have begun referring to this as the "jungle of stochastic optimization" [10].

This article is the first of a two-part tutorial designed to clarify the modeling of sequential, stochastic optimization (control) problems. This first part focuses on two objectives:

Warren B. Powell is with the Department of Operations Research and Financial Engineering, Princeton University, e-mail: (powell@princeton.edu).

Stephan Meisel is with the Department of Information Systems, University of Muenster, Germany, e-mail: (stephan.meisel@uni-muenster.de)

Manuscript received xxx, 2014

- We propose a simple, canonical modeling framework that handles uncertainty in a general way. Our framework is closest to that used by control theory, but with some key differences.
- We describe four fundamental classes of policies that encompass all the competing solution approaches that we have encountered in our tour through the literature.

One point of departure of our approach to modeling is to completely separate the design of the model from the design of policies. We find that the vast majority of papers in sequential stochastic optimization problems blend modeling with the design of policies, such that different communities (stochastic programming, dynamic programming, optimal control, robust optimization) all become closely associated with a particular class of policy (sometimes two). Our approach breaks down these artificial walls. In Part II of this tutorial, we construct variations of an energy storage problem that show that all four classes of policies may be optimal for a particular problem. Below, we also illustrate instances where each of the four classes of policies are used in the research literature or engineering practice.

Our tutorial begins in section II with a discussion of different notational systems, and a presentation of the system we use in our framework. Section III then provides our canonical model for sequential stochastic optimization (control) problems. In our presentation, we separate the presentation of the model from the design of policies, which is given in section IV, where we organize the competing approaches into four fundamental classes of policies. Section V provides a series of examples where each of the four classes of policies have been used in energy applications. Section VI concludes part I of the tutorial.

Part II of the tutorial illustrates these ideas in the context of a relatively simple energy storage problem. We describe variations of this basic problem that illustrate the strengths of *each* of the four classes of policies.

II. Notation

Developing a standard canonical framework for sequential stochastic optimization problems requires adopting a single notational system to cover a broad class of problems, where multiple notational systems are already established (and not about to change). There are three primary communities that have addressed sequential, stochastic optimization problems, each of which has introduced its own notation. These are Markov decision processes, stochastic programming, and optimal control.

The three most important pieces of notation in any sequential decision problem are 1) state variables, 2) decision (or control or action) variables, and 3) exogenous information (random variables or noise). The standard notation used in each of our three communities is:

- Markov decision processes, which uses:
  - State variable - S_t is the "state" at time t.
  - Actions - a_t (typically, but not always, discrete; the number of actions may range for 2 to 100, possibly larger, but generally not thousands or millions).
  - Exogenous information - No standard notation. Uncertainty is buried in a one-step transition matrix that is usually denoted $p(s'|s,a)$ which is the probability of transitioning to state s' given that we are in state s and take action a.
- Stochastic programming. This is a field that evolved out of deterministic mathematical programming. Standard notation is
  - State variable - This term is generally not used. They often refer to a *history* (all the information up to some time t), or a *scenario tree* where a node in the scenario tree is equivalent to the state of an exogenous information process.
  - Decisions - x_t, which is usually a vector, which may be continuous, or a mixture of integer and continuous variables; dimensionality may be hundreds to hundreds of thousands.
  - Exogenous information - No standard notation. Leading authors have used ξ_t for the new information at time t, or $\xi_{[t]}$ for the history up through time t, but others use ω_t for information learned at time t; nothing is standard. Exogenous information is represented as a scenario tree, which captures the evolution of new information that may depend on the history. A scenario might be denoted s or ω.
- Optimal control. Most optimal control problems are deterministic, possibly with uncertain parameters, and possibly with observational noise (but not always). Standard notation is
  - State variable - x_t.
  - Controls - u_t, which are almost always continuous, and may be scalar or vectors with a relatively small number of dimensions (10 or 20 dimensions is considered large).
  - Exogenous information - w_t, which is random at time t. w_t is often interpreted as measurement noise (e.g., the error in the estimate of the position or speed of a piece of equipment).

Papers in energy (appearing in IEEE Transactions on Power Systems) use all three fields, and all three notational systems. In our own work, we universally use S_t for the state at time t, we usually use x_t for decisions (which helps build bridges to the math programming community), and we use W_t for the new information that first became known at time t (unlike w_t, which is unknown at time t). We use ω to represent a sample path of $W_1, W_2, \ldots, W_T$.

We hope this brief overview helps authors accustomed to one notational system to build a bridge to our presentation.

III. A canonical model

We begin by reviewing classical models that have been used for sequential decision problems, and then propose a canonical model that draws most heavily on the style used in the controls community (but geared toward stochastic problems).

A. Existing canonical models

Mathematical programming

The mathematical programming community (typically associated with operations research) will universally write an optimization problem using

$$\min_x cx, \tag{1}$$

subject to

$$\begin{aligned} Ax &= b, & (2)\\ x &\leq u, & (3)\\ x &\geq 0, & (4) \end{aligned}$$

where x is a vector (possibly integer), c is a vector of cost coefficients, and A and b are suitably dimensioned matrices and vectors. If we wish to solve a sequential (deterministic) problem, this might be written

$$\min_{x_0,\ldots,x_T} \sum_{t=0}^{T} c_t x_t \tag{5}$$

subject to

$$\begin{aligned} A_t x_t &= R_t, \quad t = 0,\ldots,T, & (6)\\ R_{t+1} &= R_t + B_t x_t, \quad t = 0,\ldots,T-1, & (7)\\ x_t &\leq u_t, \quad t = 0,\ldots,T, & (8)\\ x_t &\geq 0, \quad t = 0,\ldots,T. & (9) \end{aligned}$$

Stochastic programming

If we introduce randomness (say, in the resource vector R_t), the stochastic community will often form a two-stage stochastic programming problem which would be written (see [2], [3])

$$\min_{x_t,(x_{t'}(\omega),t<t'\leq t+H),\forall\omega\in\Omega_t} \big(c_t x_t + \sum_{\omega\in\Omega_t} p(\omega) \sum_{t'=t+1}^{t+H} c_{t'}(\omega) x_{t'}(\omega)\big). \tag{10}$$

Here, ω is called a *scenario* drawn from a sampled set Ω_t generated for the problem we are solving at time t (many authors prefer "s" for scenario, but ω is widely used and avoids the conflict with standard notation for state). If we have a random sample, the probability of scenario ω might be $p(\omega) = 1/|\Omega_t|$. We make one decision x_t for time t, but then we have to make a decision $x_{t'}(\omega)$ for each scenario ω in the future. Note that we are writing the problem as if we are sitting at time t, to be consistent with our dynamic program above.

The constraints are tricky to write out. We start with the time t constraints which we might write

$$\begin{aligned} A_t x_t &= b_t, & (11)\\ x_t &\geq 0. & (12) \end{aligned}$$

We then have to write out constraints for time periods $t' > t$, which have to be written for each scenario ω. These might be written $\forall\omega \in \Omega_t$:

$$\begin{aligned} A_{t'}(\omega)x_{t'}(\omega) - B_{t'-1}(\omega)x_{t'-1}(\omega) &= b_{t'}(\omega), & (13)\\ x_{t'}(\omega) &\geq 0. & (14) \end{aligned}$$

Note that we require that $x_t = x_t(\omega)$, a condition that is often imposed as a *nonanticipativity constraint*. We have intentionally modeled this as a problem we are solving at time t, extending over some horizon H into the future.

Equations (10)-(14) represent a widely used two-stage approximation for a multiperiod problem. A common strategy in the planning of hydroelectric reservoirs is to use a multistage model, which can be done with scenario trees [11] or with Benders cuts [12], although the latter reference is solving a specialized problem that enables the use of a technique called stochastic dual decomposition procedure (SDDP) [13].

Dynamic programming

The most common way of expressing a dynamic program is to write Bellman's equation as [5]:

$$V_t(S_t) = \min_{a\in\mathcal{A}} \big(C(S_t,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|S_t,a)V_{t+1}(s')\big), \tag{15}$$

where:

$$\begin{aligned} S_t &= \text{the state at time } t,\\ a &= \text{the (typically discrete) action in set } \mathcal{A},\\ C(S_t,a) &= \text{the cost of being in state } S_t \text{ and taking action } a,\\ \gamma &= \text{fixed discount factor},\\ p(s'|s,a) &= \text{the probability of transitioning to state } S_{t+1} = s' \text{ if we are in state } S_t = s \text{ and take action } a,\\ V_t(s) &= \text{the value of being in state } S_t = s \text{ at time } t \text{ and following the optimal policy from } t \text{ onward.} \end{aligned}$$

This style of writing a dynamic program was introduced by [14] and laid the foundation for an entire body of research within operations research, which was then picked up by computer scientists working under the name of *reinforcement learning* (see [7], [15]). A problem with this fundamental model is that the one-step transition matrix $p(s'|s,a)$ is either completely unknown or, for all but a small class of toy problems, computationally intractable. The one-step transition matrix is dimensioned by the number of states times the number of states by the number of actions. In addition, each element of this very large matrix requires taking an expectation over the random variables, which themselves may be a vector (this is the three curses of dimensionality in [8]).

Optimal control

Optimal control texts generally focus on deterministic problems [16], [17], [4], [18] but mathematically sophisticated texts dealing with stochastic optimal control are available [19]; a particularly clear treatment of stochastic control is given in [20], using a style similar to that used for Markov decision processes [5]. Uncertainty tends to arise in optimal control settings in two forms: uncertainty in basic parameters, and (typically additive) noise when observing the state of the system (e.g., the energy stored in a battery, wind speed). While the field of optimal control has evolved in the context of more specialized problems, the notational system has been adopted by other communities (including both operations research and computer science), and as a result has been applied to a wide range of problems (indeed, it is the foundation of the framework we adopt in our work).

The most standard formulation of an optimal control problem used in the controls community assumes a deterministic model (see [4][p. 24] or [17][Chapter 1]). Using the standard notation of the controls community (state x_t, control u_t), the problem would be written as

$$min_u J(x_0, u),$$

where $J(x_0, u) = \sum_{t=0}^{T} C(x_t, u_t)$ is the cost-to-go function and where $x_{t+1} = f(x_t, u_t)$ is the transition function. Deterministic models avoid the need to deal with the uncertainty of the state. A stochastic model is given by [20](Chapter 1) where the transition function is written

$$x_{t+1} = f(x_t, u_t, w_t),$$

where w_t is a noise term that is random at time t. The value of a policy π that maps a state x_t to an action u_t has value (over a finite horizon T)

$$J^{\pi}(x_0) = \mathbb{E} \sum_{t=0}^{T} C(x_t, \pi_t(x_t), w_t).$$

The optimization problem is then written [20][p. 5]

$$J^*(x_0) = \min_{\pi \in \Pi} J^{\pi}(x_0). \tag{16}$$

This basic model underlies the entire field of Markov decision processes, but the objective function in (16) is often not stated explicitly; see, for example, the presentation of a Markov decision model given in chapters 2 and 3 of [5], where the objective function is noticeably absent (it does not appear until chapter 4, completely separated from the original statement of the model).

Comments

The formulations given above are not equivalent. The deterministic model in equations (1)-(4) (or (5)-(9)) is a deterministic *base model*. Equations (10)-(14) represent a stochastic *lookahead model*, which is a problem we are solving at time t using an approximation of the future, with the goal of finding x_t. Equations (5)-(9) could also be a deterministic lookahead model, but we would have written it as being solved at time t. The dynamic program in equation (15) is an optimality condition that serves as the basis for solving a dynamic program (which is a stochastic base model).

The only formulation above that is actually a statement of a sequential stochastic optimization problem is (16), but as written, this is computationally intractable. Our approach builds on this fundamental framework, but provides more guidance in terms of both modeling the problem (with some notational tweaks) and provides an explicit, practical roadmap to the problem of searching over policies.

B. A canonical model for sequential stochastic problem

There are five elements to any sequential, stochastic decision problem (that is, a dynamic program). These are (see Chapter 5 of [8], available at http://adp.princeton.edu, or [10]):

- State S_t - The state variable is a minimally dimensioned function of history that, along with the exogenous information, is necessary and sufficient to compute the cost function, the decision function, and the transition function. It is useful to create three sets of state variables: the physical state R_t, the information state I_t (think of this as information not in R_t), and the state of knowledge K_t (or belief state) which is a set of probability distributions describing parameters we do not know perfectly. Examples of a knowledge state include:
  - When planning replacement of transformers, we might have a distribution of belief about the deterioration of the transformer (which is updated after the transformer is inspected).
  - Sending a request to reduce load in a demand response market will elicit an uncertain response; our knowledge state could be a probability distribution of the possible response (which changes over time).
  - Increasing the price of charging an electric vehicle at a location will reduce the demand, but this relationship is uncertain, and we need to experiment with different prices to learn this function.
- Actions/decisions/controls - We use a for discrete action, u for continuous controls, or x for (possibly high dimensional) decision vectors. We explicitly avoid (when writing our model) any discussion of how to make a decision, but postulate the existence of a decision function, known as a *policy* (known in control theory as a control law) that we designate $X_t^{\pi}(S_t)$ (if we are using x as our decision). We would use $A_t^{\pi}(S_t)$ if we are using action a, or $U_t^{\pi}(S_t)$ to determine the control u. We write $X_t^{\pi}(S_t)$ if our policy explicitly depends on time; otherwise we write $X^{\pi}(S_t)$ for stationary policies. We assume that our policy produces feasible actions (say, $x_t \in \mathcal{X}_t$), where the feasible region $\mathcal{X}_t$ might be a system of linear equations that depends on the state S_t. Integrality constraints have to be captured by the policy.
  There are many problems in energy where decisions have to be made in advance, producing a *lagged* resource commitment problem. These can be modeled by letting $x_{tt'}$ be the resources committed at time t to be used at time t' (t captures the information content of the decision).
- Exogenous information W_t - Starting at time 0, we observe exogenous information (prices, loads, generator failures, rainfall) as the sequence $W_1, W_2, \ldots$. We use the convention that any variable indexed by t is known at time t (in contrast with the convention for w_t used in control theory). This means that states, actions (decisions) and information evolve as follows:

  $$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, S_t, x_t, W_{t+1}, \ldots, S_T).$$

  An important concept is the *post-decision state* which we write as S_t^x (when we use x_t as our control), which is the information in the system immediately *after* we make a decision (see [8][Chapter 4] for a thorough discussion of post-decision state variables). Introducing the post-decision state in our sequence gives us

  $$(S_0, x_0, S_0^x, W_1, S_1, x_1, S_1^x, W_2, \ldots, S_t, x_t, S_t^x, W_{t+1}, \ldots, S_T).$$

If the information process evolves exogenously, independently of current states and actions (true for many, but not all, problems), it is convenient to let ω be a sample realization of the random variables $W_1, W_2, \ldots, W_T$. We note that W_t may depend on the state S_t, and even the action x_t (selling a large amount of energy into the grid may depress random electricity prices) but not always (e.g., wind, cloud cover).

- The transition function - We write this as

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

where $S^M(\cdot)$ has been referred to as the system model, plant model, transfer function or just model, but we refer to it as the transition function, consisting of the equations that describe the evolution of the system from t to $t+1$. The controls community will often write $S_{t+1} = f(S_t, x_t, w_t)$, where w_t is random at time t; this notation has been inherited from continuous time problems, but it can create considerable confusion when used in a discrete time setting. We use $S^M(\cdot)$ for our transition function since $f(\cdot)$ is useful notation for many other purposes.

- The objective function - Let $C(S_t, x_t)$ be the (presumably deterministic) cost when we are in state S_t and make decision x_t. Some applications depend on the exogenous information which we can write $C(S_t, x_t, W_{t+1})$ or the next state which we would write $C(S_t, x_t, S_{t+1})$; in these cases, the cost is random at time t. The objective requires finding the policy that minimizes expected costs, which is written

$$\min_{\pi \in \Pi} \mathbb{E}^\pi \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)), \tag{17}$$

where $S_{t+1} = S^M(S_t, x_t, W_{t+1})$, and where the expectation is over all possible sequences $W_1, W_2, \ldots, W_T$, which may depend on the actions taken. The notation $\mathbb{E}^\pi$ allows for the possibility that the exogenous information might depend on prior actions (something that is not allowed in traditional stochastic programming models such as that given in (10)-(14)). The goal here is to find the best policy, which means that we are looking for the best *function* for making a decision.

For a nice example of this modeling framework, please see [21]. While the paper uses a classic VFA-based policy, their model is formulated in a way that is independent of any particular policy.

In practice, we cannot compute the expectation in (17). Instead, we can approximate the expectation by simulating the policy over a long simulation period T:

$$\hat{F}^\pi = \sum_{t=0}^{T} C(S_t(\omega), X_t^\pi(S_t(\omega))), \tag{18}$$

where $S_{t+1}(\omega) = S^M(S_t(\omega), X_t^\pi(S_t(\omega)), W_{t+1}(\omega))$. Or we might take an average over a series of simulations, as in

$$\bar{F}^\pi = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} C(S_t(\omega^n), X_t^\pi(S_t(\omega^n))). \tag{19}$$

Equations (18) or (19) are often referred to as simulations, often without realizing that these are approximations of equation (17).

It is very important not to confuse the *problem* (basically, equation 17) and the *policy*, which itself may be an optimization problem such as (5)-(9) or (10)-(14). In fact, our experience is that this confusion is pervasive in papers that work on sequential problems. The various communities that work on sequential decision problems (stochastic programming, dynamic programming, robust optimization) are often tightly associated with one or two classes of policies.

We argue that it is important to first formulate the problem and then address the problem of finding a policy. We sometimes refer to equation (17) as the *base model* to avoid confusion with a lookahead model (which we discuss below). In section IV, we address the challenge of finding a policy.

C. Objectives

We close our discussion of models by touching on different types of objective functions. While the expectation in (17) is by far the most common way of handling uncertainty in the objective, there has been growing interest in objectives that recognize the issue of risk. Three examples are:

- Quantile optimization - Here, we minimize the α quantile of the optimization, given by

$$\min_{\pi \in \Pi} Q_\alpha \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)), \tag{20}$$

where $Q_\alpha(W)$ is the α-quantile of the random variable W.

- Risk measures - We might use various risk measures that balance expected costs with measures of variability, such as

$$\min_{\pi \in \Pi} \varrho \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)). \tag{21}$$

Alternatively, we may need to use a nested risk measure

$$\min_{\pi \in \Pi} \varrho\big(C(S_0, X_0^\pi(S_0)) + \varrho(C(S_1, X_1^\pi(S_1)) + \varrho(\cdots))\big).$$

An example of a risk measure is $\varrho(F(x)) = \mathbb{E}F(x) + \eta\mathbb{E}(F(x) - \mathbb{E}F(x))^2$, where $F(x)$ is a random variable. See [3], [22], [23] for thorough discussions, especially on the topic of coherent risk measures.

- Alternatively, we might use a "robust" objective which uses

$$\min_{\pi \in \Pi} \max_{w \in \mathcal{W}(\theta)} \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)), \tag{22}$$

where $S_{t+1} = S^M(S_t, x_t, W_{t+1}(w))$, and where w represents a sequence $W_1, \ldots, W_T$. The maximum over $w \in \mathcal{W}(\theta)$ refers to a search over random variables within an uncertainty set $\mathcal{W}(\theta)$ parameterized by θ [24] that defines a set of "worst case" values (in its simplest form, the uncertainty set might be a set of $(1-\theta)$-confidence intervals). We note that the robust optimization community does not necessarily use (22) to evaluate policies;

instead, they may formulate a robust policy (described in section IV) which is then evaluated in an expectation-based objective (equation (17)); see [25], section 4.3. See [26], [6], [27] and [28] for further discussions of robust optimization. From numerous conversations, we feel it is safe to say that the robust optimization community has not adopted a standard methodology for evaluating a robust policy for sequential problems.

Equation (17) (or any of (20)-(22)) represents the fundamental objective function, although it is quite rare to see a paper that writes out this objective function explicitly. The choice of the objective function is part of the modeling process, and it should provide a basis for comparing policies. We strongly urge all authors to write the objective explicitly, *before* addressing the problem of designing a policy.

The challenge is now to find the best policy. We address this challenge next.

IV. Policies

A policy (known as a control law in the optimal control community) is a mapping (*any mapping*) from a state S_t to a feasible decision/control x_t. By construction, any policy that depends on the state S_t is an *admissible policy* (also referred to as "nonanticipative" or "$\mathcal{F}_t$-measurable" in the stochastic optimization literature), which simply means that the policy can only use information that is available at time t. These properties are trivially satisfied by how we have constructed our state variable and policies, so the concepts are not actually important in practice.

Our experience is that most books/papers that cover some form of stochastic optimization/stochastic control tend to quickly focus on one or possibly two classes of policy. From our experience, these policies can be organized into four fundamental classes:

- Policy function approximations (PFAs) - These are analytical functions that map states to actions, without using an embedded optimization problem. These might be discrete lookup tables (if temperature = 95, then increase generating reserves), parametric functions (charge battery if price is below θ_1, discharge if above θ_2), or statistical models, where

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2. \tag{23}$$

  Neural networks have been particularly popular in engineering communities (see [29], [30], [31]).
- Policies based on robust cost function approximations (CFAs) - To illustrate a CFA, imagine starting with a simple myopic policy that we can write

$$X_t^\pi(S_t) = \arg\min_{x\in\mathcal{X}_t} C(S_t, x).$$

  Not surprisingly, this would rarely work well in practice (but occasionally is optimal). However, there are problems where a slightly modified cost function might work quite well. One class of approximations looks like

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t} \big(C(S_t, x) + \sum_{f\in\mathcal{F}} \theta_f \phi_f(S_t, x)\big). \tag{24}$$

  Here, $(\phi_f(S_t, x))_{f\in\mathcal{F}}$, is a set of *basis functions* (as they are known in the approximate dynamic programming community) which might be of the form $S_t x$, $S_t x^2$, x, x^2, which serves as a type of correction term. However, there are other problems where we make direct changes to the cost function itself, or perhaps the constraints (e.g., requiring generating reserves, or slack when turning on a machine). We can represent this class of policies more broadly by writing

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t^\pi(\theta)} \bar{C}^\pi(S_t, x|\theta)$$

  where $\bar{C}_t^\pi(S_t, x|\theta)$ is some parametric approximation of the costs, while $\mathcal{X}_t^\pi(\theta)$ might be a modified set of constraints. Here, we would let π carry the information about the structure of the approximation (such as the basis functions in equation (24)) and we let θ capture all tunable parameters.
- Policies based on value function approximations (VFAs) - Known as the cost-to-go function in control theory, value functions capture the future value (cost or reward) of being in a state at a point in time. Ideally, the value function is capturing the value of following an optimal policy, but this is rarely possible, so we replace it with a value function approximation (VFA), which gives us a policy of the form

$$X_t^\pi(S_t|\theta) \;=\; \arg\min_{x\in\mathcal{X}_t} \big(C(S_t, x) + \mathbb{E}\{\overline{V}_{t+1}(S_{t+1}|\theta)|S_t\}\big),$$

  where $S_{t+1} = S^M(S_t, x_t, W_{t+1})$. This is what is generally known as approximate dynamic programming. A popular (if risky) strategy is to approximate the value function using a linear model, giving us

$$X_t^\pi(S_t|\theta) \;=\; \arg\min_{x\in\mathcal{X}_t} \big(C(S_t, x) + \mathbb{E}\{\sum_{f\in\mathcal{F}} \theta_f \phi_f(S_{t+1})|S_t\}\big). \tag{25}$$

  We could replace S_{t+1} with the post-decision state S_t^x and eliminate the expectation, producing a policy that looks cosmetically like our cost function approximation given in (24). However, the basis functions and regression parameters θ in (24) are completely different from those in (25), where they are being used to approximate the value of being in state S_{t+1}, while in (24), the basis functions and θ bear no relation to a value function.
- Lookahead policies - The simplest lookahead policy involves optimizing over a horizon H deterministically (this is what is usually meant as model predictive control, see [32]). Let $\tilde{x}_{tt'}$ represent the decision variables (this might be a vector) for time t' in the lookahead model that is being solved at time t (which determines the information content). Variables with tildes represent the lookahead model, so we do not confuse them with the base model. A deterministic lookahead policy might be written

$$X_t^\pi(S_t|\theta) = \arg\min_{\tilde{x}_{tt},\ldots,\tilde{x}_{t,t+H}} \sum_{t'=t}^{t+H} C(\tilde{S}_{tt'}, \tilde{x}_{tt'}), \tag{26}$$

where θ captures the horizon and perhaps the discretization (or other approximations) used in forming the lookahead model.

The stochastic programming community approximates the future by using a sampled representation of the exogenous information process, producing a policy that might look like

$$X_t^\pi(S_t|\theta) = \arg \min_{\tilde{x}_{tt},(\tilde{x}_{tt'}(\tilde{\omega}),t<t'\leq t+H),\tilde{\omega}\in\tilde{\Omega}_t} \tilde{c}_{tt}\tilde{x}_{tt} + \sum_{\tilde{\omega}\in\tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\tilde{\omega})\tilde{x}_{tt'}(\tilde{\omega}). \quad (27)$$

In this case, θ captures parameters such as the number of information stages, the number of scenarios per stage, temporal aggregation and discretization.

The robust optimization community, when working on sequential problems, would formulate a "robust" policy using [6], [25]:

$$X_t^\pi(S_t|\theta) = \arg \min_{\tilde{x}_{tt},\ldots,\tilde{x}_{t,t+H}} \max_{(w_{tt},\ldots,w_{t,t+H})\in\mathcal{W}(\theta)} \sum_{t'=t}^{t+H} \tilde{c}_{tt'}(w_{tt'})\tilde{x}_{tt'}, \quad (28)$$

where $\mathcal{W}(\theta)$ is an *uncertainty set* parameterized by θ. The uncertainty set determines costs as well as the constraints (which we have not shown), but the decision vector $\tilde{x}_t$ is deterministic.

In addition to these four major classes, it is possible to create hybrids by mixing and matching policies from different classes. Examples include using value function approximations at the end of a lookahead model, combining low-dimensional PFAs as penalty terms in any cost-based model, or using parametric cost function approximations in a lookahead model.

We note that three of the four classes involve an embedded minimization problem, which may use a linear, nonlinear or integer programming algorithm to handle high-dimensional problems. Unit commitment problems, for example, may involve tens of thousands of integer variables. We also note that the solution of these minimization problems may not be unique; we simply assume that any tie-breaking logic is embedded in the policy (as would be the case with any algorithm).

The vast majority of papers in engineering that solve a (sequential) stochastic optimization problem of some form formulate the problem around a particular type of policy. Dynamic programming/optimal control papers often start by writing the Hamilton-Jacobi-Bellman equations, which imply a policy based on an approximation of the value function (cost-to-go function), although PFAs (often equated with control laws) are widely used. Cost function approximations represent a class of policy that is widely used in practice, but has received very little attention from the academic literature.

We note that every policy can be described by some set of parameters that we refer to as θ. This might be the parameters of a PFA, CFA or VFA; the horizon of a deterministic lookahead; the horizon, number of scenarios and structure of stages in a stochastic lookahead; or the parameters guiding the choice of uncertainty set in a robust policy. The correct way to tune these parameters is in our stochastic base model given by (17) (or (20), (21) or (22)). This step opens up multiple fields that fall under names such as stochastic search [9], simulation-optimization [33], and ranking and selection.

V. Illustrating the four classes in energy applications

All four of these classes are actually used on various problems in energy systems analysis. Below we briefly illustrate examples of each.

- Policy function approximations - Examples of PFAs include:
  - A common battery storage policy is to charge the battery when the electricity price (the LMP) is below a threshold θ^{charge}, and discharge when it is above $\theta^{discharge}$. The policy is tuned by finding the right values of $\theta = (\theta^{charge}, \theta^{discharge})$.
  - A pumped hydro reservoir might operate under the schedule to charge between midnight and 4am, and to discharge between noon and 4pm (these times can be tuned).

  PFAs are familiar to the engineering community as lookup table (or parameterized rules) such as those above, or as neural networks (see e.g., [31] for many examples), where they are often called "actor nets."
- Cost function approximations - All the ISOs perform unit commitment at a time t by solving a deterministic model over some horizon $(t, t + H)$, where the model is programmed to produce a set level of spinning and nonspinnning reserves θ (which may have to be spread around the network). This is a parameterized cost function approximation where the reserve parameters θ are tuned in an online fashion (that is, the real world), although they could be tuned in a simulator (offline).
- Value function approximations - Often referred to as dynamic programming (or approximate dynamic programming), value functions are particularly useful in the control of storage problems (see [34], [35], [36]). Value functions are widely approximated in the controls community using neural networks where they are often referred to as "critic nets."
- Lookahead policies - These are typically used whenever problems are nonstationary and a good forecast is available. They are universally used in unit commitment by the ISOs who use a deterministic forecast with reserve adjustments [37]. Many papers have appeared in the research literature for the unit commitment problem studying the use of stochastic lookahead models using the stochastic programming framework [38], [39] or robust optimization [40], [41], although these are limited to the research literature at this stage. Deterministic lookahead policies have also long been used as a standard engineering heuristic in the planning of natural gas storage [34]. Stochastic lookahead models have long been used for hydroelectric power plannning [11].

Typically, problem structure will suggest that one or two of the policies are a natural fit. Most papers focus on one class of policy, often because even analyzing a single policy can be fairly challenging. It is easy to overlook the fact that hybrids can be quite effective. For example, there are many papers addressing the "stochastic unit commmitment problem" where they assume that industry practice is to use a deterministic model, whereas industry practice is to use a modified deterministic model (which we refer to as a hybrid cost function approximation/lookahead policy) which is optimized in a stochastic base model. PFAs can also appear as low-dimensional patterns (e.g., pump water into the hydro facility between midnight and 4pm, release the water between noon and 4pm) that are embedded within high-dimensional unit commitment models.

In part II of the tutorial, we use a single energy storage problem to show that each of the four classes might work best depending on the data.

VI. Conclusions

The goal of this tutorial is to provide a canonical framework for modeling a wide range of stochastic, dynamic optimization problems, using simple but precise notation, with a clear path to computation. At the heart of the framework is an objective function that is expressed in terms of optimizing over a broad class of policies. We then argue that the vast literature on sequential stochastic optimization problems can be organized into four fundamental classes of policies which, we believe, covers every computationally implementable approach used in the literature.

Three of our four classes of policies have received considerable attention in the research literature: PFAs, VFAs and the various forms of lookaheads (the term "policy function approximation" is a recent addition, but analytical policies that do not involve an embedded optimization problem are widely used). Lookahead policies often appear under names such as model predictive control and rolling horizon procedures (for deterministic lookahead models), stochastic programming for stochastic lookahead models, or robust optimization (adapted to sequential problems).

Policies based on cost function approximations have received virtually no attention in the stochastic optimization literature, but are widely used in engineering practice as a heuristic. However, we feel that a parametric approximation of a cost function is qualitatively similar to a policy function approximation or a policy based on a value function approximation.

We close our discussion with some simple guidelines that we feel should be followed when modeling any stochastic, dynamic system:

- *Model* your problem *before* you solve it (that is, design a policy). Assume a policy is given, and make sure that anyone can simulate your system given a policy.
- Acknowledge that there are different classes of policies, even if you want to focus on just one class. Make a case for why you have settled on a particular class, and be sure to simulate variations of your policy to ensure that you have the best policy within a class. Comparisons against a standard benchmark such as a deterministic lookahead should always be included.

Part II of this tutorial illustrates these ideas in the context of an energy storage problem. We create variations that demonstrate that *each* policy class may be best for specific problem characteristics.

Acknowledgment

The work of the first author was supported by the National Science Foundation under grant ECCS-1127975 and the SAP initiative for energy systems research. The work of the second author was supported by the German Research Foundation.

References

[1] P. Kall and S. Wallace, *Stochastic Programming*. John Wiley & Sons, 1994.
[2] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, 2nd ed. New York: Springer, 2011.
[3] A. Shapiro, D. Dentcheva, and A. Ruszczyski, *Lectures on Stochastic Programming: Modeling and theory*, 2nd ed. Philadelphia: SIAM, 2014.
[4] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*, 3rd ed. Hoboken, NJ: John Wiley & Sons, 2012.
[5] M. L. Puterman, *Markov Decision Processes*, 2nd ed. Hoboken, NJ: John Wiley and Sons, 2005.
[6] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton NJ: Princeton University Press, 2009.
[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA: MIT Press, 1998, vol. 35, no. 3.
[8] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2011.
[9] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, simulation and control*. Hoboken, NJ: John Wiley & Sons, 2003.
[10] W. B. Powell, "Clearing the Jungle of Stochastic Optimization Clearing the Jungle of Stochastic Optimization," *Informs TutORials in Operations Research 2014*, no. October, 2014.
[11] J. Jacobs, G. Freeman, J. Grygier, D. P. Morton, G. Schultz, K. Staschus, and J. Stedinger, "SOCRATES: A system for scheduling hydroelectric generation under uncertainty," *Annals of Operations Research*, vol. 59, no. 1, pp. 99–133, Dec. 1995.
[12] A. Shapiro, "Minimax and risk averse multistage stochastic programming," *European Journal of Operational Research*, vol. 219, no. 3, pp. 719–726, Jun. 2012.
[13] M. F. Pereira and L. M. V. G. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, pp. 359–375, 1991.
[14] R. E. Bellman, *Dynamic Programming*. Princeton, N.J.: Princeton University Press, 1957.
[15] C. Szepesvári, *Algorithms for Reinforcement Learning*. Morgan and Claypool, Jan. 2010, vol. 4, no. 1.
[16] R. F. Stengel, *Optimal Control and Estimation*. Dover Publications, New York, 1994.
[17] D. E. Kirk, *Optimal Control Theory: An introduction*. New York: Dover, 2004.
[18] E. D. Sontag, *Mathematical Control Theory: Deterministic finite dimensional systems*, 2nd ed. New York: Springer, 2014.
[19] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The discrete time case*. New York: Academic Press, 1978, vol. 0.
[20] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II: Approximate Dynamic Programming*, 4th ed. Belmont, MA: Athena Scientific, 2012.
[21] X. Xi and R. Sioshansi, "A stochastic dynamic programming model for co-optimization of distributed energy storage," *Energy Systems*, vol. 5, no. 3, pp. 474–505, 2014.
[22] A. Ruszczyski, "Risk-averse dynamic programming for Markov decision processes," *Mathematical Programming*, vol. 125, pp. 235–261, 2010.
[23] R. T. Rockafellar and S. Uryasev, "The fundamental risk quadrangle in risk management, optimization, and statistical estimation," *Surveys in Operations Research and Management Science*, vol. 18, no. 1, pp. 33–53, 2013.

[24] C. Bandi and D. Bertsimas, "Tractable stochastic analysis in high dimensions via robust optimization," *Mathematical Programming, Series B*, vol. 134, pp. 23–70, 2012.
[25] A. Ben-Tal, B. Golany, A. Nemirovski, and J.-p. Vial, "Retailer-Supplier Flexible Commitments Contracts: A Robust Optimization Approach," *Manufacturing & Service Operations Management*, vol. 7, no. 3, pp. 248–271, 2005.
[26] H. Beyer and B. Sendhoff, "Robust optimization - A comprehensive survey," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 33-34, pp. 3190–3218, Jul. 2007.
[27] J. Goh and M. Sim, "Distributionally robust optimization and its tractable approximations," *Operations Research*, vol. 58, no. 4, Part 1 of 2, pp. 902–917, 2010.
[28] W. Wiesemann, D. Kuhn, and M. Sim, "Distributionally robust convex optimization," *Operations Research*, vol. 62, no. 6, pp. 1358–1376, 2014.
[29] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[30] S. Haykin, *Neural Networks: A comprehensive foundation*. Englewood Cliffs, N.J.: Prentice Hall, 1999.
[31] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, "Handbook of Learning and Approximate Dynamic Programming," *Wiley-IEEE Press*, 2004.
[32] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.
[33] M. C. Fu, "Optimization for simulation: Theory vs. practice," *Informs Journal on Computing*, vol. 14, no. 3, pp. 192–215, 2002.
[34] G. Lai, F. Margot, and N. Secomandi, "An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation," *Operations Research*, vol. 58, no. 3, pp. 564–582, Jun. 2010.
[35] X. Xi, R. Sioshansi, and V. Marano, "A stochastic dynamic programming model for co-optimization of distributed energy storage," *Energy Systems*, vol. 5, no. 3, pp. 475–505, 2013.
[36] D. R. Jiang, T. V. Pham, W. B. Powell, D. F. Salas, and W. R. Scott, "A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work?" in *IEEE Conference on Approximate Dynamic Programming and Reinforcement Learning*. Orlando, FL: IEEE, 2014, pp. 1–8.
[37] N. Padhy, "Unit commitment-a bibliographical survey," *IEEE Transactions on power systems*, vol. 19, no. 2, pp. 1196–1205, 2004.
[38] S. Takriti, J. R. Birge, and E. Long, "A stochastic model for the unit commitment problem," *IEEE Trans. on Power Systems*, vol. 11, no. 3, pp. 1497–1508, 1996.
[39] S. Ryan, R. J.-B. Wets, D. L. Woodruff, C. Silva-Monroy, and J.-P. Watson, "Toward scalable, parallel progressive hedging for stochastic unit commitment," in *Power and Energy Society General Meeting (PES)*. IEEE, 2013, pp. 1–5.
[40] Street, A., F. Oliveira, and J. M. Arroyo, "Contingency-constrained unit commitment with security criterion: A robust optimization approach," *IEEE Trans. on Power Systems*, vol. 26, no. 3, pp. 1581–1590, 2011.
[41] D. Bertsimas, E. Litvinov, X. Sun, J. Zhao, and T. Zheng, "Adaptive robust optimization for the security constrained unit commitment problem," *IEEE Trans. on Power Systems*, vol. 28, no. 1, pp. 52–63, 2013.

Warren B. Powell is a professor in the Department of Operations Research and Financial Engineering at Princeton University, where he has taught since 1981. His research specializes in computational stochastic optimization, with applications in energy, transportation, health and finance. He has authored/coauthored over 200 publications and two books. He founded and directs CASTLE Labs (www.castlelab.princeton.edu), specializing in fundamental contributions to computational stochastic optimization with a wide range of applications.

Stephan Meisel is an assistant professor in the Department of Information Systems at the University of Muenster, Germany. Before joining the department in 2013, he was a postdoctoral research associate in the Department of Operations Research and Financial Engineering at Princeton University. The focus of his research is on computational stochastic optimization with applications in energy and transportation.