A Unified Framework for Stochastic Optimization

Warren B. Powell

Princeton University
January 30, 2018

Abstract

Stochastic optimization is an umbrella term that includes over a dozen fragmented communities, using a patchwork of often overlapping notational systems with algorithmic strategies that are suited to specific classes of problems. This paper reviews the canonical models of these communities, and proposes a universal modeling framework that encompasses all of these competing approaches. At the heart is an objective function that optimizes over policies which is standard in some approaches, but foreign to others. We then identify four meta-classes of policies that encompasses all of the approaches that we have identified in the research literature or industry practice. In the process, we observe that any adaptive learning algorithm, whether it is derivative-based or derivative-free, is a form of policy that can be tuned to optimize either the cumulative reward (similar to multi-armed bandit problems) or final reward (as is used in ranking and selection or stochastic search). We argue that the principles of bandit problems, long a niche community, should become a core dimension of mainstream stochastic optimization.

Keywords: Dynamic programming, stochastic programming, stochastic search, bandit problems, optimal control, approximate dynamic programming, reinforcement learning, robust optimization, Markov decision processes, ranking and selection, simulation optimization

Contents

1. Introduction

There are many communities that contribute to the problem of making decisions in the presence of different forms of uncertainty, motivated by a vast range of applications spanning business, science, engineering, economics and finance, health and transportation. Decisions may be binary, discrete, continuous or categorical, and may be scalar or vector. Even richer are the different ways that uncertainty arises. The combination of the two creates a virtually unlimited range of problems.

A byproduct of this diversity has been the evolution of different mathematical modeling styles and solution approaches. In some cases communities developed a new notational system followed by an evolution of solution strategies. In other cases, a community might adopt existing notation, and then adapt a modeling framework to a new problem setting, producing new algorithms and new research questions.

Our point of departure from deterministic optimization, where the goal is to find the best decision, is to address the problem of finding the best *policy*, which is a function for making decisions given what we know (sometimes called a "decision rule"). Throughout, we capture what we know at time t by a state variable S_t (we may sometimes write this as S^n to capture what we know after n iterations). We *always* assume that the state S_t has *all* the information we need to know at time t from history to model our system from time t onward, even if we know some parameters probabilistically (more on this later).

We will then define a function $X^\pi(S_t)$ to represent our policy that returns a decision $x_t = X^\pi(S_t)$ given our state of knowledge S_t about our system. Stated compactly, a policy is a mapping (*any mapping*) from state to a feasible action. We let $C_t(S_t, x_t, W_{t+1})$ be our performance metric (e.g. a cost or contribution) that tells us how the decision performs (this metric may or may not depend on S_t or W_{t+1}). Once we make our decision x_t, we then observe new information W_{t+1} that takes us to a new state S_{t+1} using a *transition function*

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

Our optimization challenge is to solve the problem

$$\max_\pi \mathbb{E}\left\{\sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t), W_{t+1})|S_0\right\}, \tag{2}$$

where S_t evolves according to equation (1), and where we have to specify an exogenous information process that consists of the sequence

$$(S_0, W_1, W_2, \ldots, W_T). \tag{3}$$

Given the already broad scope of this article, we will restrict our attention to problems that maximize or minimize expected performance, but we could substitute a nonlinear risk metric (introducing substantial computational complexity). The objective in (2) expresses the goal of maximizing the *cumulative reward* (summing rewards over time), but there are many problems where we are only interested in the *final reward*, which we can express by letting $C_t(\ldots) = 0$ for $t = 0, \ldots, T-1$.

This article will argue that (1)-(3) forms the basis for a universal model that can be used to represent virtually every expectation-based stochastic optimization problem. At this same time, this framework disguises the richness of stochastic optimization problems. This framework introduces two types of challenges:

- Modeling - Modeling sequential decision problems is often the most difficult task, and requires a strong understanding of state variables, the different types of decisions and information, and the dynamics of how the system evolves over time (which may not be known).
- Designing policies - Given a model, we have to design a policy that maximizes (or minimizes) our objective in (2).

Different communities in stochastic optimization differ in both how they approach modeling, and most approach the problem of searching over policies by working within one or two classes of policies.

This review extends the thinking of two previous tutorial articles. Powell (2014) was our first effort at articulating four classes of policies which we first hinted at in Powell (2011)[Chapter 6]. Powell (2016) extended this thinking, recognizing for the first time that these four classes fell into two important categories: policy search (a term used in computer science), which requires searching over a class of (typically parametric) functions, and policies based on lookahead approximations, where we approximate in different ways the downstream value of a decision made now. Each category can be further divided into two classes, producing what we refer to as the four (meta)classes of policies. While different communities have embraced each of these four classes of policies, we have shown (Powell & Meisel (2016*a*)) that each of the four classes may work best depending on the data, although choices are often guided by the characteristics of the problem.

The process of developing a single framework that bridges between all the different communities is already identifying opportunities for cross-fertilization. This review makes the following observations which the reader might keep in mind while progressing through the article:

- The stochastic optimization communities have treated optimization of the final reward (often under terms such as "ranking and selection" or stochastic search) as distinctly different from optimization of the cumulative reward (commonly done in dynamic programming and multiarmed bandit problems), but these are just different objective functions. While the choice of the best policy will depend on the objective, the process of finding the best policy does not.
- The multiarmed bandit problem can be viewed as a derivative-free stochastic search problem using a cumulative reward objective function. Maximizing cumulative rewards is often overlooked in stochastic optimization, while some communities (notably dynamic programming) use a cumulative reward objective when the real interest is in the final reward. While the process of optimizing over policies may be the same, it is still important to use the correct formulation (later in the article we argue that the newsvendor is an example of a misformulated problem).

- This article identifies (for the first time) two important problem classes:

  State-independent problems - In this class, the state variable captures only our belief about an unknown function, but where the problem itself does not depend on the state variable.

  State-dependent problems - Here, the contributions, constraints, and/or transition function depends on dynamically varying information.

  Both of these problems can be modeled as dynamic programs, but are classically treated using different approaches. We argue that both can be approached using the same framework (1) - (3), and solved using the same four classes of policies.

- Classical algorithms such as stochastic gradients methods can be viewed as dynamic programs, opening the door to addressing the challenge of designing optimal *algorithms*.

- Most communities in stochastic optimization focus on a particular approach for designing a policy. We claim that all four classes of policies should at least be considered. In particular, the approach using policy search and the approach based on lookahead approximations each offer unique strengths and weaknesses that should be considered when designing practical solution strategies.

We also demonstrate that our framework will open up new questions by taking the perspective of one problem class into a new problem domain.

2. The communities of stochastic optimization

Deterministic optimization can be organized along two major lines of investigation: math programming (linear, nonlinear, integer), and deterministic optimal control. Each of these fields has well-defined notational systems that are widely used around the world.

Stochastic optimization, on the other hand, covers a much wider class of problems, and as a result has evolved along much more diverse lines of investigation. Complicating the organization of these contributions is the observation that over time, research communities which started with an original, core problem and modeling framework have evolved to address new challenges which require new algorithmic strategies. This has resulted in different communities doing research on similar problems with similar strategies, but with different notation, and asking different research questions.

Below we provide a summary of the most important communities, using the notation most familiar to each community. Later, we are going to introduce a single notational system which strikes a balance between using notation that is most familiar and which provides the greatest transparency. All of these fields are quite mature, so we try to highlight some of the early papers as well as recent contributions in addition to some of the major books and review articles that do a better job of summarizing the literature than we can, given the scope of our treatment. However, since our focus is integrating across fields, we simply cannot do justice to the depth of the research taking place within each field.

Readers may wish to just skim this section on a first pass so they can have a quick sense of the diverse modeling frameworks, but then move to the rest of the paper. However, if you choose to give it a careful read, please pay attention not just to the differences in notation, but the different ways each community approaches the process of modeling. Some key modeling characteristics are

- Problem statement - Deterministic math programs are represented as objective functions subject to constraints. Stochastic optimization problems might similarly be represented as optimizing an objective (although they vary in terms of how they state what they are optimizing over), but other communities will state an optimality condition (Bellman's equation) or a policy (such as the lookahead policies in stochastic programming). Differences in how problems are stated easily introduces the greatest confusion.

- State variables - In operations research, many equate "state" with physical state such as inventory or the location of a vehicle. In engineering controls, "state" might be estimates of parameters. In stochastic search, the "state" might capture the state of an algorithm (for derivative-based algorithms) or the belief about a function (for derivative-free algorithms). For bandit problems, "state" is the belief (in the form of a statistical model) about an unknown function.

- Decisions under uncertainty - A decision x_t (or action a_t or control u_t) has to be made with the information available at that time. This is represented as an action at a node (in a tree), a "measurable function" (common in optimal stopping and control theory), "nonanticipativity constraints" (in stochastic programming), an action chosen by solving Bellman's optimality equation, or a policy $X^\pi(S_t)$ that chooses an action x_t that depends on a state S_t (which is the most general).

- Representing uncertainty - Stochastic programming will represent future events as scenarios, Markov decision processes bury uncertainty in a one-step transition matrix, robust optimization models uncertainty in terms of uncertainty sets, reinforcement learning (and many papers in optimal control for engineering) use a data-driven approach by assuming that uncertainty can be observed but not modeled.

- Modeling system dynamics - Stochastic programming will capture dynamics in systems of linear equations, Markov decision problems use a one-step transition matrix, optimal control uses a transition function ("state equation"), while several communities (engineering controls, reinforcement learning) will often assume that transitions can only be observed.

- Objective functions - We may wish to minimize costs, regret, losses, errors, risk, volatility, or we may maximize rewards, profits, gains, utility, strength, conductivity, diffusivity and effectiveness. Often, we want to optimize over multiple objectives, although we assume that these can be rolled into a utility function.

These differences are subtle, and may be difficult to identify on a first read.

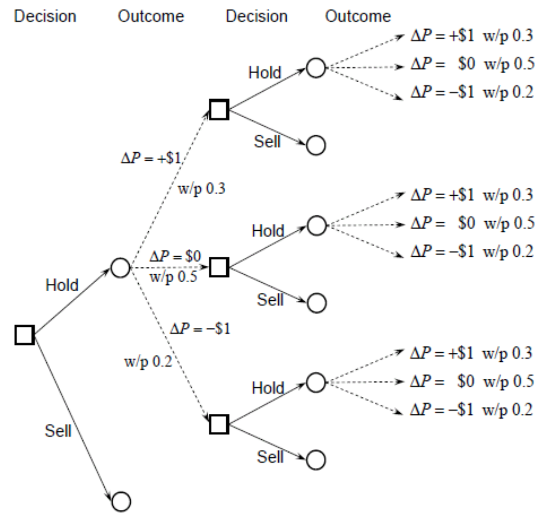

Figure 1: Illustration of a simple decision tree for an asset selling problem.

2.1. Decision trees

Arguably the simplest stochastic optimization problem is a decision tree, illustrated in figure 1, where squares represent decision nodes (from which we choose an action), and circles represent outcome nodes (from which a random event occurs). Decision trees are typically presented without mathematics and therefore are very easy to communicate. However, they explode in size with the decision horizon, and are not at all useful for vector-valued decisions.

Decision trees have proven useful in a variety of complex decision problems in health, business and policy (Skinner, 1999). There are literally dozens of survey articles addressing the use of decision trees in different application areas.

2.2. Stochastic search

Derivative-based stochastic optimization began with the seminal paper of Robbins & Monro (1951) which launched an entire field. The canonical stochastic search problem is written

$$\max_x \mathbb{E} F(x, W), \tag{4}$$

where W is a random variable, while x is a continuous scalar or vector (in the earliest work). We assume that we can compute gradients (or subgradients) $\nabla_x F(x, W)$ for a sample W. The classical stochastic gradient algorithm of Robbins & Monro (1951) is given by

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}), \tag{5}$$

where α_n is a stepsize that has to satisfy

$$\alpha_n > 0, \tag{6}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \tag{7}$$

$$\sum_{n=0}^{\infty} \alpha_n^2 < \infty. \tag{8}$$

Stepsizes may be deterministic, such as $\alpha_n = 1/n$ or $\alpha_n = \theta/(\theta + n)$, where θ is a tunable parameter. Also popular are stochastic stepsizes that adapt to the behavior of the algorithm (see Powell & George (2006) for a review of stepsize rules). Easily the biggest challenge of these rules is the need to tune parameters. Important recent developments which address this problem to varying degrees include AdaGrad (Duchi et al., 2011), Adam (Kingma & Ba, 2015) and PiSTOL (Orabona, 2014).

Stochastic gradient algorithms are used almost universally in Monte Carlo-based learning algorithms. A small sample of papers includes the early work on unconstrained stochastic search including Wolfowitz (1952) (using numerical derivatives), Blum (1954) (extending to multidimensional problems), and Dvoretzky (1956). A separate line of research focused on constrained problems under the umbrella of "stochastic quasi-gradient" methods, with seminal contributions from Ermoliev (1968), Shor (1979), Pflug (1988*b*), Kushner & Clark (1978), Shapiro & Wardi (1996), and Kushner & Yin (2003). As with other fields, this field broadened over the years. The best recent review of the field (under this name) is Spall (2003). Bartlett et al. (2007) approaches this topic from the perspective of online algorithms, which refers to stochastic gradient methods where samples are provided by an exogenous source. Broadie et al. (2011) revisits the stepsize conditions (6)-(8).

We note that there is a different line of research on deterministic problems using randomized algorithms that is sometimes called "stochastic search" which is outside the scope of this article.

2.3. Optimal stopping

Optimal stopping is a niche problem that has attracted significant attention in part because of its simple elegance, but largely because of its wide range of applications in the study of financial options (Karatzas (1988), Longstaff & Schwartz (2001), Tsitsiklis & Van Roy (2001)), equipment replacement (Sherif & Smith, 1981) and change detection (Poor & Hadjiliadis, 2009).

Let $W_1, W_2, \ldots, W_t, \ldots$ represent a stochastic process that might describe stock prices, the state of a machine or the blood sugar of a patient. For simplicity, assume that $f(W_t)$ is the reward we receive if we stop at time t (e.g. selling the asset at price W_t). Let ω refer to a particular sample path of $W_1, \ldots, W_T$ (assume we are working with finite horizon problems). Now let

$$X_t(\omega) = \begin{cases} 1 & \text{if we stop at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\tau(\omega)$ be the first time that $X_t = 1$ on sample path ω. The problem here is that ω specifies the entire sample path, so writing $\tau(\omega)$ makes it seem as if we can decide when to stop based on the *entire* sample path. This notation is hardly unique to the optimal stopping literature as we see below when we introduce stochastic programming.

We can fix this by constructing the function X_t so that it only depends on the history $W_1, \ldots, W_t$. When this is done, τ is called a *stopping time*. In this case, we call X_t an *admissible policy*, or we would say that "X_t is $\mathcal{F}_t$-measurable" or *nonanticipative* (these terms are all equivalent). We would then write our optimization problem as

$$\max_{\tau} \mathbb{E} X_\tau f(W_\tau), \tag{9}$$

where we require τ to be a stopping time, or we would require the function X_τ to be $\mathcal{F}_t$-measurable or an admissible policy.

There are different ways to construct admissible policies. The simplest is to define a state variable S_t which only depends on the history $W_1, \ldots, W_t$. For example, define a physical state $R_t = 1$ if we are still holding our asset (that is, we have not stopped). Further assume that the W_t process is a set of prices $p_1, \ldots, p_t$, and define a smoothed price process $\bar{p}_t$ using

$$\bar{p}_t = (1-\alpha)\bar{p}_{t-1} + \alpha p_t.$$

At time t, our state variable is $S_t = (R_t, \bar{p}_t, p_t)$. A policy for stopping might be written

$$X^\pi(S_t|\theta) = \begin{cases} 1 & \text{if } \bar{p}_t > \theta^{max} \text{ or } \bar{p}_t < \theta^{min} \text{ and } R_t = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Finding the best policy means finding the best $\theta = (\theta^{min}, \theta^{max})$ by solving

$$\max_{\theta} \mathbb{E} \sum_{t=0}^{T} p_t X^\pi(S_t|\theta). \tag{10}$$

So, now our search over admissible stopping times τ becomes a search over the parameters θ of a policy $X^\pi(S_t|\theta)$ that only depend on the state. This transition hints at the style that we are going to use in this paper.

Optimal stopping is an old and classic topic. An elegant presentation is given in Cinlar (1975) with a more recent discussion in Cinlar (2011) where it is used to illustrate filtrations. DeGroot (1970) provides a nice summary of the early literature. One of the earliest books dedicated to the topic is Shiryaev (1978) (originally in Russian). Moustakides (1986) describes an application to identifying when a stochastic process has changed, such as the increase of incidence in a disease or a drop in quality on a production line. Feng & Gallego (1995) uses optimal stopping to determine when to start end-of-season sales on seasonal items. There are numerous uses of optimal stopping in finance (Azevedo & Paxson, 2014), energy (Boomsma et al., 2012) and technology adoption (Hagspiel et al., 2015), to name just a few.

2.4. Optimal control

The canonical stochastic control problem is typically written

$$\min_{u_0,\ldots,u_T} \mathbb{E}\left\{\sum_{t=0}^{T-1} L_t(x_t,u_t) + L_T(x_T)\right\}, \tag{11}$$

where $L_t(x_t,u_t)$ is a loss function with terminal loss $L_T(x_T)$, and where the state x_t evolves according to

$$x_{t+1} = f(x_t,u_t) + w_t, \tag{12}$$

where $f(x_t,u_t)$ is variously known as the transition function, system model, plant model (as in chemical or power plant), plant equation, and transition law. Here, w_t is a random variable representing exogenous noise, such as wind blowing an aircraft off course. A more general formulation is to use $x_{t+1} = f(x_t,u_t,w_t)$, which allows w_t to affect the dynamics in a nonlinear way.

It is typically the case in engineering control problems that (12) is linear in the state x_t and control u_t. In addition, it is common to assume that the true state $\hat{x}_t$ (for example, the location and speed of an aircraft) can only be observed up to an additive noise, as in $x_t = \hat{x}_t + \epsilon_t$.

The engineering controls community primarily focuses on deterministic problems where $w_t = 0$, in which case we are optimizing over deterministic controls $u_0,\ldots,u_T$. For the stochastic version, we follow a sample path $w_0(\omega), w_1(\omega),\ldots,w_T(\omega)$, with a corresponding set of controls $u_t(\omega)$ for $t = 0,\ldots,T$. Here, ω represents an entire sample path, so writing $u_t(\omega)$ makes it seem as if u_t gets to "see" the entire trajectory. As with the optimal stopping problem, we can fix this by insisting that u_t is "$\mathcal{F}_t$-measurable," or by saying that u_t is an "admissible policy" which recognizes that u_t is actually a function rather than a decision variable. Alternatively, we can handle this by writing $u_t = \pi_t(x_t)$ where $\pi_t(x_t)$ is a policy that determines u_t given the state x_t, which by construction is a function of information available up to time t. The challenge then is to find a good policy that only depends on the state x_t.

For the control problem in (11), it is typically the case in engineering applications that the objective function will have the quadratic form

$$L_t(x_t,u_t) = (x_t)^T Q_t x_t + (u_t)^T R_t u_t.$$

When the transition function (12) (typically referred to as the "state equations") is linear in the state x_t and control u_t, and the control u_t is unconstrained, the problem is referred to as "linear quadratic regulation" (LQR).

This problem is typically solved using the Hamilton-Jacobi equation, given by

$$J_t(x_t) = \min_{u_t}\left(L(x_t,u_t) + \int_w J_{t+1}(f(x_t,u_t,w))g^W(w)dw\right), \tag{13}$$

where $g^W(w)$ is the density of the random variable w_t and where $J_t(x_t)$ is known as the "cost to go." When we exploit the linear structure of the transition function and the quadratic structure of the loss function, it is possible to find the cost-to-go function $J_t(x_t)$ analytically, which allows us to show that the optimal policy has the form

$$\pi_t(x_t) = K_t x_t,$$

where K_t is a complex matrix that depends on Q_t and R_t. This is a rare instance of a problem where we can actually compute an optimal policy.

There is a long history in the development of optimal control, summarized by many books including Kirk (2004), Stengel (1986), Sontag (1998), Sethi & Thompson (2000), and Lewis et al. (2012). The canonical control problem is continuous, low-dimensional and unconstrained, which leads to an analytical solution. Of course, applications evolved past this canonical problem, leading to the use of numerical methods. Deterministic optimal control is widely used in engineering, whereas stochastic optimal control has tended to involve much more sophisticated mathematics. Some of the most prominent books include Astrom (1970), Kushner & Kleinman (1971), Bertsekas & Shreve (1978), Yong & Zhou (1999), Nisio (2014) (note that some of the books on deterministic controls touch on the stochastic case).

As a general problem, stochastic control covers any sequential decision problem, so the separation between stochastic control and other forms of sequential stochastic optimization tends to be more one of vocabulary and notation (Bertsekas (2011) is a good example of a book that bridges these vocabularies). Control-theoretic thinking has been widely adopted in inventory theory and supply chain management (e.g. Ivanov & Sokolov (2013) and Protopappa-Sieke & Seifert (2010)), finance (Yu et al., 2010), and health services (Ramirez-Nafarrate et al., 2014), to name a few.

2.5. Markov decision processes

Richard Bellman initiated the study of sequential, stochastic, decision problems in the setting of discrete states and actions. We assume that there is a set of discrete states $\mathcal{S}$, where we have to choose an action $a \in \mathcal{A}_s$ when we are in state $s \in \mathcal{S}$ after which we receive a reward $r(s,a)$. The challenge is to choose actions (or more precisely, a policy for choosing actions), that maximizes expected rewards over time.

The most famous equation in this work (known as "Bellman's optimality equation") writes the value of being in a discrete state s as

$$V_t(s) = \max_{a \in \mathcal{A}_s} \left(r(s,a) + \sum_{s' \in \mathcal{S}} P(s'|s,a) V_{t+1}(s') \right). \tag{14}$$

where the matrix $P(s'|s,a)$ is the one-step transition matrix defined by

$P(s'|s,a) =$ The probability that state $S_{t+1} = s'$ given that we are in state $S_t = s$ and take action a.

This community often treats the one-step transition matrix as data, but it can be notoriously hard to compute. In fact, buried in the one-step transition matrix is an expectation which can be written

$$P(s'|s,a) \quad = \quad \mathbb{E}_W\{\mathbb{1}_{\{s'=S^M(s,a,W)\}}\} \tag{15}$$

where $s' = S^M(s,a,W)$ is the transition function with random input W. Note that any of s, a and/or W may be vector-valued, highlighting what are known as the three curses of dimensionality in dynamic programming.

Equation (14) is the discrete analog of the Hamilton-Jacobi equations used in the optimal control literature (given in equation (13)), leading many authors to refer to these as Hamilton-Jacobi-Bellman equations (or HJB for short). This work was initially reported in his classic reference (Bellman, 1957) (see also (Bellman, 1954) and (Bellman et al., 1955)), but this work was continued by a long stream of books including Howard (1960) (another classic), Nemhauser (1966), Denardo (1982), Heyman & Sobel (1984), leading up to Puterman (2005) (this first appeared in 1994). Puterman's book represents the last but best in a long series of books on Markov decision processes, and now represents the major reference in the field.

If we could compute equation (14) for all states $s \in \mathcal{S}$, stochastic optimization would not exist as a field. This highlights the consistent message that the central issue of stochastic optimization is computation.

2.6. Approximate/adaptive/neuro-dynamic programming

Bellman's equation (14) requires enumerating all states (assumed to be discrete), which is problematic if the state variable is a vector, a condition known widely as the curse of dimensionality. Actually, there are three curses of dimensionality which all arise when computing the one-step transition matrix $p(s'|s,a)$: the state variable s, the action a (which can be a vector), and the random information, which is hidden in the calculation of the probability (see equation (15)).

Bellman recognized this and began experimenting with methods for approximating value functions (see Bellman & Dreyfus (1959) and Bellman et al. (1963)), but the operations research community then seemed to drop any further research in approximation methods until the 1980's. As computers improved, researchers began tackling Bellman's equation using numerical approximation methods, with the most comprehensive presentation in Judd (1998) which summarized almost a decade of research (see also Chen et al. (1999)).

A completely separate line of research in approximations evolved in the control theory community with the work of Paul Werbos (Werbos (1974)) who recognized that the "cost-to-go function" (the same as the value function in dynamic programming, written as $J_t(x_t)$ in equation (13)) could be approximated using various techniques. Werbos helped develop this area through a series of papers (examples include Werbos (1989), Werbos (1990), Werbos (1992) and Werbos (1994)). Important references are the edited volumes (White & Sofge, 1992) and (Si et al., 2004) which highlighted what had already become a popular approach using neural networks to approximate both policies ("actor nets") and value functions ("critic nets").

Building on work developing in computer science under the umbrella of "reinforcement learning" (reviewed below), Tsitsiklis (1994) and Jaakkola et al. (1994) were the first to recognize that the basic algorithms being developed under the umbrella of reinforcement learning represented generalizations of the early stochastic gradient algorithms of Robbins & Monro (1951). Bertsekas & Tsitsiklis (1996) laid the foundation for adaptive learning algorithms in dynamic programming, using the name "neuro-dynamic programming." Werbos, (e.g. Werbos (1992)), had been using the term "approximate dynamic programming," which became the title of Powell (2007) (with a major update in Powell (2011)), a book that also merged math programming and value function approximations to solve high-dimensional, convex stochastic optimization problems (but, see the developments under stochastic programming below). Later, the engineering controls community reverted to "adaptive dynamic programming" as the operations research community adopted "approximate dynamic programming."

There are many variations of approximate dynamic programming, but one of the simplest involves using some policy $\pi(S_t)$ to simulate from a starting state S_0 until an ending period T. Assume we do this repeatedly, and let S_t^n be the state we visit at time t during iteration n. Assume our policy returns action $a_t^n = \pi(S_t^n)$, and let $S_t^{a,n}$ be the state immediately after we implement action a_t^n, known as the *post-decision* state (an example of the post-decision state is the outcome node in a decision tree). Finally let $\overline{V}_t^{a,n-1}(S_t^a)$ be an approximation of the value of being in post-decision state based on information from the first $n-1$ iterations. We can compute a sampled estimate of the value $\hat{v}_t^n$ of being in pre-decision state S_t^n using

$$\hat{v}_t^n = C(S_t^n, a_t^n) + \overline{V}_t^{a,n-1}(S_t^{a,n}). \tag{16}$$

Now update the value function approximation using

$$\overline{V}^{a,n}(S_{t-1}^{a,n}) = (1-\alpha_{n-1})\overline{V}^{a,n-1}(S_{t-1}^{a,n}) + \alpha_{n-1}\hat{v}_t^n, \tag{17}$$

where $S_{t-1}^{a,n}$ is the post-decision state we visited before arriving to the next pre-decision state S_t^n. We then compute $S_t^{a,n}$ from S_t^n and a_t^n, after which we simulate our way to S_{t+1}^n and repeat the process.

Using equations (16)-(17) requires a policy to guide the choice of action. One we might use is a greedy policy where (16) is replaced with

$$\hat{v}_t^n = \max_a \left(C(S_t^n, a_t^n) + \overline{V}_t^{a,n-1}(S_t^{a,n})\right).$$

While a pure exploitation policy can work quite poorly, there are special cases where it can produce an optimal policy.

Equations (16)-(17) are best described as "forward approximate dynamic programming" since they involve stepping forward through states. This is attractive because it works for very high dimensional applications. In fact, the idea has been applied to optimizing major trucking companies and railroads (Simao et al. (2009), Bouzaiene-Ayari et al. (2016)), but these applications exploit linearity and convexity. More recently researchers have applied the idea of approximating value functions from a sampled set of states in a method described as "backward approximate dynamic programming" (Senn et al. (2014), Cheng et al. (2017), Durante et al. (2017)).

2.7. Reinforcement learning

Independently from the work in operations research (with Bellman) or control theory (the work of Werbos), computer scientists Andy Barto and his student Rich Sutton were working on describing the behavior of mice moving through a maze in the early 1980's. They developed a basic algorithmic strategy called Q-learning, which iteratively estimates the value of being in a state s and taking an action a, given by $Q(s, a)$ (the "Q factors"). These estimates are computing using

$$\hat{q}^n(s^n, a^n) = r(s^n, a^n) + \gamma \max_{a'} Q^{n-1}(s^{n+1}, a'), \tag{18}$$

$$Q^n(s^n, a^n) = (1 - \alpha_{n-1})Q^{n-1}(s^n, a^n) + \alpha_{n-1}\hat{q}^n(s^n, a^n), \tag{19}$$

where $\hat{q}^n(s^n, a^n)$ is a sampled estimate of the value of being in state $s = s^n$ and taking action $a = a^n$, and where γ is a discount factor. The sampled estimates "bootstrap" the downstream value $Q^{n-1}(s', a')$. The parameter α_n is a "stepsize" or "learning rate" which has to satisfy (6)-(8). The state s^{n+1} is a sampled version of the next state we would visit given that we are in state s^n and take action a^n. This is sometimes written as being sampled from the one-step transition matrix $P(s'|s^n, a^n)$ (if this is available), although it is more natural to write $s^{n+1} = f(s^n, a^n, w^n)$ where $f(s^n, a^n, w^n)$ is the transition function and w^n is a sample of exogenous noise.

The reinforcement learning community traditionally estimates Q-factors that depend on state and action, whereas Bellman's equation (and approximate dynamic programming) focus on developing estimates of the value of being in a state. These are related using

$$V(s) = \max_a Q(s, a).$$

We emphasize that equations (16)-(17) are computed given a policy $\pi(s)$, which means that the action is implicit when we specify the policy.

These basic equations became widely adopted for solving a number of problems. The field of reinforcement learning took off with the appearance of their now widely cited book (Sutton & Barto, 1998), although by this time the field was quite active (see the review Kaelbling et al. (1996)). Research under the umbrella of "reinforcement learning" has evolved to include other algorithmic strategies under names such as policy search and Monte Carlo tree search. Other references from the reinforcement learning community include Busoniu et al. (2010) and Szepesvári (2010) (a second edition of Sutton & Barto (1998) is in preparation).

2.8. Online algorithms

Online algorithms technically refer to methods that respond to data sequentially without any knowledge of the future. Technically, this would refer to *any* policy that depends on a properly formulated state variable which could include a forecast of the future, possibly in the form of a value function. In practice, the field of online algorithms refer to procedures that do not even attempt to approximate the future, which means they are some form of myopic policy (see Borodin & El-Yanniv (1998) for a nice introduction and Albers (2003) for a survey).

Online algorithms were originally motivated by the need to make decisions in a computationally constrained setting such as a robot or device in the field with limited communication or energy sources. This motivated models that made no assumptions about what might happen in the future, producing myopic policies. This in turn produced a body of research known as competitive analysis that develops bounds on the performance compared to a perfectly clairvoyant policy.

Online algorithms have attracted considerable attention in complex scheduling problems such as those that arise in transportation (Jaillet & Wagner (2006), Berbeglia et al. (2010), Pillac et al. (2013)) and machine scheduling (Ma et al. (2010), Slotnick (2011)).

2.9. Model predictive control

This is a subfield of optimal control, but it became so popular that it evolved into a field of its own, with popular books such as Camacho & Bordons (2003) and hundreds of articles (see Lee (2011) for a 30-year review). MPC is a method where a decision is made at time t by solving a typically approximate model over a horizon $(t, t+H)$. The need for a model, even if approximate, is the basis of the name "model predictive control"; there are many settings in engineering where a model is not available. MPC is typically used to solve a problem that is modeled as deterministic, but it can be applied to stochastic settings by using a deterministic approximation of the future to make a decision now, after which we experience a stochastic outcome. MPC can also use a stochastic model of the future, although these are typically quite hard to solve.

Model predictive control is better known as a rolling horizon procedure in operations research, or a receding horizon procedure in computer science. Most often it is associated with deterministic models of the future, but this is primarily because most of the optimal control literature in engineering is deterministic. MPC could use a stochastic model of the future which might be a Markov decision process (often simplified) which is solved (at each time period) using backward dynamic programming. Alternatively, it may use a sampled approximation of the future, which is the standard strategy of stochastic programming which some authors will refer to as model predictive control (Schildbach & Morari, 2016).

2.10. Stochastic programming

The field of stochastic programming evolved from deterministic linear programming, with the introduction of random variables. The first paper in stochastic programming was Dantzig (1955), which introduced what came to be called the "two-stage stochastic programming problem" which is written as

$$\min_{x_0} \left(c_0 x_0 + \sum_{\omega \in \Omega} p(\omega) \min_{x_1(\omega) \in \mathcal{X}_1(\omega)} c_1(\omega) x_1(\omega) \right). \tag{20}$$

Here, x_0 is the first-stage decision (imagine allocating inventory to warehouses), which is subject to first stage constraints

$$A_0 x_0 \leq b_0, \tag{21}$$

$$x_0 \geq 0. \tag{22}$$

Then, the demands D_1 are revealed. These are random, with a set of possible realizations $D_1(\omega)$ for $\omega \in \Omega$ (these are often referred to as "scenarios"). For each scenario ω, we have to obey the following constraints in the second stage for all $\omega \in \Omega$:

$$A_1(\omega)x_1(\omega) \leq x_0, \tag{23}$$

$$B_1(\omega)x_1(\omega) \leq D_1(\omega). \tag{24}$$

There are two-stage stochastic programming *problems*, but in most applications it is used as an approximation of a fully sequential ("multistage") problem. In these settings, the first-stage decision x_0 is really a decision x_t at time t, while the second stage can represent decisions $x_{t+1}(\omega), \ldots, x_{t+H}(\omega)$ which are solved for a sample realization of all random variables over the horizon $(t, t+H)$. In this context, two-stage stochastic programming is a stochastic form of model predictive control.

Stochastic programs are often computationally quite difficult, since they are basically deterministic optimization problems that are $|\Omega|$ times larger than the deterministic problem. Rockafellar & Wets (1991) present a powerful decomposition procedure called *progressive hedging* that decomposes (20)-(24) into a series of problems, one per scenario, that are coordinated through Lagrangian relaxation.

Whether it is for a two-stage problem, or an approximation in a rolling horizon environment, two-stage stochastic programming has evolved into a mature field within the math programming community. A number of books have been written on stochastic programming (two stage, and its much harder extension, multistage), including Pflug (1988*a*), Kall & Wallace (2009), Birge & Louveaux (2011) and Shapiro et al. (2014).

Since stochastic programs can become quite large, a community has evolved that focuses on how to generate the set of scenarios Ω. Initial efforts focused on ensuring that scenarios were not too close to each other (Dupacova et al. (2003), Heitsch & Romisch (2009), Löhndorf (2016)); more recent research focuses on identifying scenarios that actually impact decisions (Bayraksan & Love, 2015). Of considerable interest is work on sampling that directly addresses solution quality and decisions (Bayraksan & Morton, 2009).

A parallel literature has evolved for the study of stochastic linear programs that exploits the natural convexity of the problem. The objective function (20) is often written

$$\min_{x_0} \left(c_0 x_0 + \mathbb{E}Q(x_0, W_1) \right), \tag{25}$$

subject to (21)-(22). The function $Q(x_0, W_1)$ is known as the recourse function where W_1 captures all sources of randomness. For example, we might write $W_1 = (A_1, B_1, c_1, D_1)$, with sample realization $W_1(\omega)$. The recourse function is given by

$$Q(x_0, W_1(\omega)) = \min_{x_1(\omega) \in \mathcal{X}_t(\omega)} c_1(\omega)x_1(\omega) \tag{26}$$

where the feasible region $\mathcal{X}_t(\omega)$ is defined by equations (23) - (24).

There is an extensive literature exploiting the natural convexity of $Q(x_0, W_1)$ in x_0, starting with Van Slyke & Wets (1969), followed by the seminal papers on stochastic decomposition (Higle & Sen,

1991) and the stochastic dual decomposition procedure (SDDP) (Pereira & Pinto, 1991). A substantial literature has unfolded around this work, including Shapiro (2011) who provides a careful analysis of SDDP, and its extension to handle risk measures (Shapiro et al. (2013), Philpott et al. (2013)). A number of papers have been written on convergence proofs for Benders-based solution methods, but the best is Girardeau et al. (2014). A modern overview of the field is given by Shapiro et al. (2014).

2.11. Robust optimization

Robust optimization first emerged in engineering problems, where the goal was to find the best design x that worked for the worst possible outcome of an uncertain parameter $w \in W$ (the robust optimization community uses $u \in \mathcal{U}$, but this conflicts with control theory notation). The robust optimization problem is formulated as

$$\min_{x \in \mathcal{X}} \max_{w \in \mathcal{W}} F(x, w). \tag{27}$$

Here, the set $\mathcal{W}$ is known as the uncertainty set, which may be a box where each dimension of w is limited to minimum and maximum values. The problem with using a box is that it might allow, for example, each dimension w_i of w to be equal to its minimum or maximum, which is unlikely to occur in practice. For this reason, $\mathcal{W}$ is sometimes represented as an ellipse, although this is more complex to create and solve.

Equation (27) is the robust analog of our original stochastic search problem in equation (4). Robust optimization was originally motivated by the need in engineering to design for a "worst-case" scenario (defined by the uncertainty set $\mathcal{W}$). It then evolved as a method for doing stochastic optimization without having to specify the underlying probability distribution. However, this has been replaced by the need to create an uncertainty set.

A thorough review of the field of robust optimization is contained in Ben-Tal et al. (2009) and Bertsimas et al. (2011), with a more recent review given in Gabrel et al. (2014). Bertsimas & Sim (2004) studies the price of robustness and describes a number of important properties. Robust optimization is attracting interest in a variety of application areas including supply chain management (Bertsimas & Thiele (2006), Keyvanshokooh et al. (2016)), energy (Zugno & Conejo, 2015). and finance (Fliege & Werner, 2014).

2.12. Ranking and selection

Assume we are trying to find the best choice x in a set $\mathcal{X} = \{x_1, \ldots, x_M\}$, where x might be the choice a diabetes treatment, the price of a product, the color for a website, or the path through a network. Let μ_x be the true performance of x, which could be the reduction of blood sugar, the revenue from the product, the hits on a website, or the time to traverse the network.

We do not know μ_x, but we run experiments to create estimates $\bar{\mu}^n_x$. Let S^n capture what we have learned after n experiments (the estimates $\bar{\mu}^n_x$, along with statistics capturing the precision of this estimate), and let $X^\pi(S^n)$ be our rule (policy) for deciding the experiment $x^n = X^\pi(S^n)$ that we will run next, after which we observe $W^{n+1}_{x^n} = \mu_{x^n} + \varepsilon^{n+1}$.

Let $\bar{\mu}_x^N$ be our estimates after we exhaust our budget of N, and let

$$x^{\pi,N} = \arg\max_{x\in\mathcal{X}} \bar{\mu}_x^N$$

be the best choice given what we know after we have finished our experiments. The final design $x^{\pi,N}$ is a random variable, in part because the true μ is random (if we are using a Bayesian model), and also because of the noise in the observations $W^1, \ldots, W^N$.

We can express the value of our policy for a set of observations based on our estimates $\bar{\mu}_x^N$ using

$$F^\pi = \bar{\mu}^N_{x^{\pi,N}}.$$

This value depends on the true values μ_x for all x, and on the results of the experiments W^n which themselves depend on μ. We can state the optimization problem as

$$\max_\pi \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\ \bar{\mu}_{x^{\pi,N}}. \tag{28}$$

With the exception of optimal stopping (equation (10)), this is the first time we have explicitly written our optimization problem in terms of searching over policies.

Ranking and selection enjoys a long history dating back to the 1950's, with an excellent treatment of this early research given by the classic DeGroot (1970), with a more up to date review in Kim & Nelson (2007). Recent research has focused on parallel computing (Luo et al. (2015), Ni et al. (2016)) and handling unknown correlation structures (Qu et al., 2012). However, ranking and selection is just another name for derivative-free stochastic search, and has been widely studied under this umbrella (Spall, 2003). The field has attracted considerable attention from the simulation-optimization community, reviewed next.

2.13. Simulation optimization

The field known as "simulation optimization" evolved from within the community that focused on problems such as simulating the performance of the layout of a manufacturing system. The simulation-optimization community adopted the modeling framework of ranking and selection, typically using a frequentist belief model that requires doing an initial test of each design. The problem is then how to allocate computing resources over the designs given initial estimates.

Perhaps the best known method that evolved specifically for this problem class is known as optimal computing budget allocation, or OCBA, developed by Chun-Hung Chen in Chen (1995), followed by a series of articles (Chen (1996), Chen et al. (1997), Chen et al. (1998), Chen et al. (2003), Chen et al. (2008)), leading up to the book Chen & Lee (2011) that provides a thorough overview of this field. The field has focused primarily on discrete alternatives (e.g. different designs of a manufacturing system), but has also included work on continuous alternatives (e.g. Hong & Nelson (2006)). An important recent result by Ryzhov (2016) shows the asymptotic equivalence of OCBA and expected improvement policies which maximize the value of information. When the number of alternatives is much larger (say, 10,000), techniques such as simulated annealing, genetic algorithms and tabu search (adapted for stochastic environments) have been brought to bear. Swisher et al. (2000) contains a nice review of this literature. Other reviews include Andradóttir (1998*a*), Andradóttir (1998*b*), Azadivar (1999), Fu (2002), and Kim & Nelson (2007). The recent review Chau et al. (2014) focuses on gradient-based methods.

The simulation-optimization community has steadily broadened into the full range of (primarily offline) stochastic optimization problems reviewed above, just as occurred with the older stochastic search community, as summarized in Spall (2003). This evolution became complete with Fu (2014), an edited volume that covers a very similar range of topics as Spall (2003), including derivative-based stochastic search, derivative-free stochastic search, and full dynamic programs.

2.14. Multiarmed bandit problems

The multiarmed bandit problem enjoys a rich history, centered on a simple illustration. Imagine that we have M slot machines, each with expected (but unknown) winnings μ_x, $x \in \mathcal{X} = \{1, \ldots, M\}$. Let S^0 represent our prior distribution of belief about each μ_x, where we might assume that our beliefs are normally distributed with mean $\bar{\mu}^0_x$ and precision $\beta^0_x = 1/\bar{\sigma}^{2,0}_x$ for each x. Further let S^n be our beliefs about each x after n plays, and let $x^n = X^\pi(S^n)$ be the choice of the next arm to play given S^n, producing winnings $W^{n+1}_{x^n}$. The goal is to find the best policy to maximize the *total* winnings over our horizon.

For a finite time problem, this problem is almost identical to the ranking and selection problem, with the only difference that we want to maximize the cumulative rewards, rather than the final reward. Thus, the objective function would be written (assuming a Bayesian prior) as

$$\max_\pi \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} W^{n+1}_{X^\pi(S^n)}. \tag{29}$$

Research started in the 1950's with the much easier two-armed problem. DeGroot (1970) was the first to show that an optimal policy for the multiarmed bandit problem could be formulated (if not solved) using Bellman's equation (this is true of *any* learning problem, regardless of whether we are maximizing final or cumulative rewards). The first real breakthrough occurred in Gittins & Jones (1974) (the first and most famous paper), followed by Gittins (1979). This line of research introduced what became known as "Gittins indices," or more broadly, "index policies" which involve computing an index ν^n_x given by

$$\nu^n_x = \bar{\mu}^n_x + \Gamma(\bar{\mu}^n_x, \bar{\sigma}^n_x, \sigma_W, \gamma)\sigma^W,$$

where σ^W is the (assumed known) standard deviation of W, and $\Gamma(\bar{\mu}^n_x, \bar{\sigma}^n_x, \sigma_W, \gamma)$ is the Gittins index, computed by solving a particular dynamic program. The Gittins index policy is then of the form

$$X^{GI}(S^n) = \arg\max_x \nu^n_x. \tag{30}$$

While computing Gittins indices is possible, it is not easy, sparking the creation of an analytical approximation reported in Chick & Gans (2009).

The theory of Gittins indices was described thoroughly in his first book (Gittins, 1989), but the "second edition" (Gittins et al., 2011), which was a complete rewrite of the first edition, represents the best introduction to the field of Gittins indices, which now features hundreds of papers. However, the field is mathematically demanding, with index policies that are difficult to compute.

A parallel line of research started in the computer science community with the work of Lai & Robbins (1985) who showed that a simple policy known as *upper confidence bounding* possessed the property that the number of times we test the wrong arm can be bounded (although it continues to grow with n). The ease of computation, combined with these theoretical properties, made this line of research extremely attractive, and has produced an explosion of research. While no books on this topic have appeared as yet, an excellent monograph is Bubeck & Cesa-Bianchi (2012). A sample of a UCB policy (designed for normally distributed rewards) is

$$X^{UCB1}(S^n) = \arg\max_x \left(\bar{\mu}^n_x + 4\sigma_W \sqrt{\frac{\log n}{N^n_x}} \right), \tag{31}$$

where N^n_x is the number of times we have tried alternative x. The square root term can shrink to zero if we test x often enough, or it can grow large enough to virtually guarantee that x will be sampled.

UCB policies are typically used in practice with a tunable parameter, with the form

$$X^{UCB1}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}^n_x + \theta^{UCB} \sqrt{\frac{\log n}{N^n_x}} \right). \tag{32}$$

We need to tune θ^{UCB} to find the value that works best. We do this by replacing the search over policies π in equation (29) with a search over values for θ^{UCB}. In fact, once we open the door to using tuned policies, we can use any number of policies such as interval estimation

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}^n_x + \theta^{IE} \bar{\sigma}^n_x \right), \tag{33}$$

where $\bar{\sigma}^n_x$ is the standard deviation of $\bar{\mu}^n_x$, which tends toward zero if we observe x often enough. Again, the policy would have to be tuned using equation (29).

These same ideas have been applied to bandit problems using a terminal reward objective using the label the "best arm" bandit problem (see Audibert & Bubeck (2010), Kaufmann et al. (2016), Gabillon et al. (2012)). It should be apparent that any policy that can be tuned using equation (29) can be tuned using equation (28) for terminal rewards.

2.15. Partially observable Markov decision processes

An extension of the multiarmed bandit problem and generalization of the standard Markov decision process model is one where we assume that the discrete states $s \in \mathcal{S}$ are not directly observable. For example, imagine that s captures the status of a tumor in a patient, or the inventory of units of blood in a hospital with a poor inventory control system. In both cases, the state s cannot be observed directly. Let $b^n(s)$ be the belief about s after n transitions, which is to say, the probability that we are in state s, where $\sum_{s\in\mathcal{S}} b^n(s) = 1$.

Assume that we take an action a^n and then make some observation W^{n+1} (some authors denote this as O^{n+1} for "observation", but the notation is not standard). The observation W^{n+1} could be a noisy observation of the state s^n (for example, $W^{n+1} = s^n + \varepsilon^{n+1}$), or an indirect measurement from which we can make inferences about our system (e.g. the existence of marker molecules in the blood that might indicate the presence of tumors). Assume that we know the conditional distribution of W^{n+1} given by $P^W[W^{n+1} = s|s^n, a^n]$, which would be derived from the relationship between the observation W^{n+1} and the true state s^n (e.g. if the patient actually has cancer) and action a^n (which could be a particular type of medical test).

Such a problem is termed a *partially observable Markov decision process*, or POMDP, where "s" is the unobservable state (sometimes called the environment), while b is the vector of probabilities that we are in s, also known as the belief state. We can write the belief space as $\mathcal{B} = \{b| \sum_{s\in\mathcal{S}} b(s) = 1\}$. The set $\mathcal{S}$ can be quite large in many settings. If we have three continuous state variables that we discretize into 100 elements, then we have a million states, which means that b is a million-dimensional vector.

As with our Markov decision process, let $P(s'|s, a)$ be our one-step transition matrix for the unobservable states, which we assume is known. The belief state evolves according to (see, e.g. Shani et al. (2013))

$$\begin{aligned}
b^{n+1}(s') &= P[S^{n+1} = s'|b^n, a^n, W^{n+1}] \\
&= \frac{P[b^n, S^{n+1} = s', a^n, W^{n+1}]}{P[b^n, a^n, W^{n+1}]} \\
&= \frac{P^W[W^{n+1}|b^n, S^{n+1} = s', a^n] P[S^{n+1} = s'|b^n, a^n] P[b^n, a^n]}{P[W^{n+1}|b^n, a^n] P[b^n, a^n]} \\
&= \frac{P^W[W^{n+1}|b^n, S^{n+1} = s', a^n] \sum_{s\in\mathcal{S}} P[S^{n+1} = s'|b^n, S^n = s, a^n] P[S^n = s|b^n, a^n]}{P[W^{n+1}|b^n, a^n]} \\
&= \frac{P^W[W^{n+1}|b^n, S^{n+1} = s', a^n] \sum_{s\in\mathcal{S}} P[S^{n+1} = s'|b^n, S^n = s, a^n] b^n(s)}{P[W^{n+1}|b^n, a^n]} \qquad (34)
\end{aligned}$$

where we used $b^n(s) = P[S^n = s] = P[s|b^n, a^n]$, and where

$$P^W[W^{n+1}|b^n, a^n] = \sum_{s\in\mathcal{S}} b^n(s) \sum_{s'\in\mathcal{S}} P[S^{n+1} = s'|b^n, S^n = s, a^n] P^W[W^{n+1}|b^n, S^{n+1} = s', a^n].$$

POMDPs are characterized by the property that the entire history

$$h^n = (b^0, a^0, W^1, b^1, a^1, W^2, \ldots\ldots, a^{n-1}, W^n, b^n)$$

is fully summarized by the latest belief b^n. POMDPs are characterized by two transition matrices: the one-step transition matrix for the system state $P[S^{n+1} = s'|S^n = s, a^n]$, and the conditional observation distribution $P^W[W^{n+1} = w|S^n = s, b^n, a^n]$. Both of these can be derived in principle from the physics of the problem, although computing them is another matter.

POMDPs are notoriously hard to solve, and as a result the computational side has attracted considerable attention (see Lovejoy (1991) and Aberdeen (2003) for early surveys). One of the earliest breakthroughs was the dissertation (Sondik, 1971) that found that the value function can be represented as a series of cuts (see Sondik (1978) and Smallwood et al. (1973)). However, the strategy that has attracted the most attention is based on the idea of "point-based" solvers (see Pineau et al. (2003) and Smith & Simmons (2005) for examples, and Shani et al. (2013) for a survey of point-based solvers).

POMDPs can be modeled as conventional Markov decision processes where the state is just the belief state (which is generally continuous), and where equation (34) is the transition function (Sondik, 1971). This is sometimes referred to as the "belief MDP" (see, for example, Cassandra et al. (1994), Oliehoek et al. (2008), Ross et al. (2008*b*), Ross et al. (2008*a*)). Further complicating the situation is that there are many settings where the state variable consists of a mixture of observable parameters and belief states. For example, the multiarmed bandit problem is an example of a problem where the only state variables are belief states, which reflect unobservable and uncontrollable parameters that either do not change over time, or which change but not due to any decisions.

2.16. Discussion

Each of the topics above represents a distinct community, most with entire books dedicated to the topic. We note that some of these communities focus on *problems* (stochastic search, optimal stopping, optimal control, Markov decision processes, robust optimization, ranking and selection, multiarmed bandits), while others focus on *methods* (approximate dynamic programming, reinforcement learning, model predictive control, stochastic programming), although some of these could be described as methods for particular problem classes.

In the remainder of our presentation, we are going to present a single modeling framework that covers all of these problems. We begin by noting that there are problems that can be solved exactly, or approximately by using a sampled version of the different forms of uncertainty. However, most of the time we end up using some kind of adaptive search procedure which uses either Monte Carlo sampling or direct, online observations (an approach that is often called *data driven*).

We are then going to argue that *any* adaptive search strategy can be represented as a policy for solving an appropriately defined dynamic program. Solving any dynamic program involves searching over policies, which is the same as searching for the best algorithm. We then show that there are two fundamental strategies for designing policies, leading to four meta-classes of policies which cover all of the approaches used by the different communities of stochastic optimization.

3. Solution strategies

There are three core strategies for solving stochastic optimization problems:

Deterministic/special structure - These are problems that exhibit special structure that make it possible to find optimal solutions. Examples include: linear programs where costs are actually expectations of random variables; the newsvendor problem with known demand where we can use the structure to find the optimal order quantity; and Markov decision processes with a known one-step transition matrix, which represents the expectation of the event that we transition to a downstream state.

Sampled models - There are many problems where the expectation in $\max_x \mathbb{E}F(x, W)$ cannot be computed, but where we can replace the original set of outcomes Ω (which may be multidimensional and/or continuous) with a sample $\hat{\Omega}$. We can then replace our original stochastic optimization problem with

$$\max_x \sum_{\hat{\omega} \in \hat{\Omega}} \hat{p}(\hat{\omega}) F(x, \hat{\omega}). \tag{35}$$

This strategy has been pursued under different names in different communities. This is what is done in statistics when a batch dataset is used to fit a statistical model. It is used in stochastic programming (see section 2.10) when we use scenarios to approximate the future. It is also known as the sample average approximation, introduced in Kleywegt et al. (2002) with a nice summary in Shapiro et al. (2014). There is a growing literature focusing on strategies for creating effective samples so that the set $\hat{\Omega}$ does not have to be too large (Dupacova et al. (2003), Heitsch & Romisch (2009), Bayraksan & Morton (2011)). An excellent recent survey is given in Bayraksan & Love (2015).

Adaptive algorithms - While solving sampled models is a powerful strategy, by far the most widely used approaches depend on adaptive algorithms which work by sequentially sampling random information, either using Monte Carlo sampling from a stochastic model, or from field observations.

The remainder of this article focuses on adaptive algorithms, which come in derivative-based forms (e.g. the stochastic gradient algorithm in (5)) and derivative-free (such as policies for multiarmed bandit problems including upper confidence bounding in (32) and interval estimation in (33)). We note that *all* of these algorithms represent sequential decision problems, which means that they are all a form of dynamic program.

In the next section, we propose a canonical modeling framework that allows us to model all adaptive learning problems in a common framework.

4. A universal canonical model

We now provide a modeling framework with which we can create a single canonical model that describes all of the problems described in section 2. We note that in designing our notation, we had to navigate the various notational systems that have evolved across these communities. For example, the math programming community uses x for a decision, while the controls community uses x_t for the state and u_t for their control. We have chosen S_t for the state variable (widely used in dynamic programming and reinforcement learning), and x_t for the decision variable (universally used in math programming, but also used by the bandit community). We have worked to use the most common notational conventions, resolving conflicts as necessary.

There are five fundamental elements to any sequential decision problem: state variables, decision variables, exogenous information, the transition function, and the objective function. A brief summary of each of these elements is as follows:

State variables - The state S_t of the system at time t is a function of history which, combined with a policy and exogenous information, contains all the information that is necessary and sufficient to model our system from time t onward. This means it has to capture the information needed to compute costs, constraints, and (in model-based formulations) how this information evolves over time (which is the transition function).

We distinguish between the initial state S_0 and the dynamic state S_t for $t > 0$. The initial state contains all deterministic parameters, initial values of dynamic parameters, and initial probabilistic beliefs about unknown parameters. The dynamic state S_t contains information that is evolving over time.

There are three types of information in S_t:

- The physical state, R_t, which in most (but not all) applications is the state variables that are being controlled. R_t may be a scalar, or a vector with element R_{ti} where i could be a type of resource (e.g. a blood type) or the amount of inventory at location i.
- Other information, I_t, which is any information that is known deterministically not included in R_t. The information state often evolves exogenously, but may be controlled or at least influenced by decisions (e.g. selling a large number of shares may depress prices).
- The belief state B_t, which contains distributional information about unknown parameters, where we can use frequentist or Bayesian belief models. These may come in the following styles:
  - Lookup tables - Here we have a belief $\bar{\mu}^n_x$ which is our estimate of $\mu_x = \mathbb{E}F(x, W)$ after n observations for each discrete x. With a Bayesian model, we treat μ_x as a random variable that is normally distributed with $\mu_x \sim N(\bar{\mu}^n_x, \bar{\sigma}^{2,n}_x)$.
  - Parametric belief models - We might assume that $\mathbb{E}F(x, W) = f(x|\theta)$ where the function $f(x|\theta)$ is known but where θ is unknown. We would then describe θ by a probability distribution.
  - Nonparametric belief models - These approximate a function at x by smoothing local information near x.

We emphasize that the belief state carries the parameters of a distribution describing an unobservable parameter of the model. B_t might be the mean and variance of a normal distribution or the parameters of a log-normal distribution, while the distribution itself (e.g. the normal distribution) is specified in S_0.

The state S_t is sometimes referred to as the *pre-decision state* because it is the state just before we make a decision. We often find it useful to define a *post-decision state* S_t^x which is the state immediately after we make a decision, before any new information has arrived, which means that S_t^x is a deterministic function of S_t and x_t. For example, in a basic inventory problem where $R_{t+1} = \max\{0, R_t + x_t - \hat{D}_{t+1}\}$, the post-decision state would be $S_t^x = R_t^x = R_t + x_t$. Post-decision states are often simpler, because there may be information in S_t that is only needed to make the decision x_t, but there are situations where x_t becomes a part of the state.

Decision variables - Decisions are typically represented as a_t for discrete actions, u_t for continuous (typically vector-valued) controls, and x_t for general continuous or discrete vectors. We use x_t as our default, but find it useful to use a_t when decisions are categorical.

Decisions may be binary (e.g. for a stopping problem), discrete (e.g. an element of a finite set), continuous (scalar or vector), integer vectors, and categorical (e.g. the attributes of a patient). We note that entire fields of research are sometimes distinguished by the nature of the decision variable.

We assume that decisions are made with a policy, which we might denote $X^\pi(S_t)$ (if we use x_t as our decision), $A^\pi(S_t)$ (if we use a_t), or $U^\pi(S_t)$ (if we use u_t). We assume that a decision $x_t = X^\pi(S_t)$ is feasible at time t. We let "π" carry the information about the type of function $f \in \mathcal{F}$ (for example, a linear model with specific explanatory variables, or a particular nonlinear model), and any tunable parameters $\theta \in \Theta^f$. We use x_t as our default notation for decisions.

Exogenous information - We let W_t be any new information that first becomes known at time t (that is, between $t-1$ and t). When modeling specific variables, we use "hats" to indicate exogenous information. Thus, $\hat{D}_t$ could be the demand that arose between $t-1$ and t, or we could let $\hat{p}_t$ be the change in the price between $t-1$ and t.

The exogenous information process may be stationary or nonstationary, purely exogenous or state (and possibly action) dependent. We let ω represent a sample path $W_1, \ldots, W_T$, where $\omega \in \Omega$, and where $\mathcal{F}$ is the sigma-algebra on Ω. We also let $\mathcal{F}_t = \sigma(W_1, \ldots, W_t)$ be the sigma-algebra generated by $W_1, \ldots, W_t$. We adopt the style throughout that any variable indexed by t is $\mathcal{F}_t$-measurable, something we guarantee by how decisions are made and information evolves (in fact, we do not even need this vocabulary).

Transition function - We denote the transition function by

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}), \tag{36}$$

where $S^M(\cdot)$ is also known by names such as system model, plant model, plant equation and transfer function. Equation (36) is the classical form of a transition function which gives the equations from the pre-decision state S_t to pre-decision state S_{t+1}. We can also break down these equations into two steps: pre-decision to post-decision S^x_t, and then the post-decision S^x_t to the next pre-decision S_{t+1}. The transition function may be a known set of equations, or unknown, such as when we describe human behavior or the evolution of CO2 in the atmosphere. When the equations are unknown the problem is often described as "model free" or "data driven."

Transition functions may be linear, continuous nonlinear or step functions. When the state S_t includes a belief state B_t, then the transition function has to include the frequentist or Bayesian updating equations.

Given a policy $X^\pi(S_t)$, an exogenous process W_t and a transition function, we can write our sequence of states, decisions, and information as

$$(S_0, x_0, S^x_0, W_1, S_1, x_1, S^x_1, W_2, \ldots, x_{T-1}, S^x_{T-1}, W_T, S_T).$$

Below we continue to use t as our iteration counter, but we could use n if appropriate, in which case we would write states, decisions and information as S^n, x^n and W^{n+1}.

Objective functions - We assume that we have a metric that we are maximizing (our default) or minimizing, which we can write in state-independent or state-dependent forms:

State-independent We write this as $F(x, W)$, where we assume we have to fix x_t or x^n and then observe W_{t+1} or W^{n+1}. In an adaptive learning algorithm, the state S_t (or S^n) captures what we know about $\mathbb{E}F(x, W)$, but the function itself depends only on x and W, and not on the state S.

State-dependent These can be written in several ways:

- $C(S_t, x_t)$ - This is the most popular form, where $C(S_t, x_t)$ can be a contribution (for maximization) or cost (for minimization). This is written in many different ways by different communities, such as $r(s, a)$ (the reward for being in state s and taking action a), $g(x, u)$ (the gain from being in state x and using control u), or $L(x, u)$ (the loss from being in state x and using control u).
- $C(S_t, x_t, W_{t+1})$ - We might use this form when our contribution depends on the information W_{t+1} (such as the revenue from serving the demand between t and $t + 1$).
- $C(S_t, x_t, S_{t+1})$ - This form is used in model-free settings where we do not have a transition function or an ability to observe W_{t+1}, but rather just observe the downstream state S_{t+1}.

Of these, $C(S_t, x_t, W_{t+1})$ is the most general, as it can be used to represent $F(x, W)$, $C(S_t, x_t)$, or (by setting $W_{t+1} = S_{t+1}$), $C(S_t, x_t, S_{t+1})$. We can also make the contribution time-dependent, by writing $C_t(S_t, x_t, W_{t+1})$, allowing us to capture problems where the cost *function* depends on time. This is useful, for example, when the contribution in the final time period is different from all the others.

Assuming we are trying to maximize the expected sum of contributions, we may write the objective function as

$$\max_{\pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t), W_{t+1}) | S_0 \right\}, \tag{37}$$

where

$$S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1}). \tag{38}$$

We refer to equation (37) along with the state transition function (38) as the *base model.*

Equations (37)-(38) may be implemented in a simulator (offline), or by testing in an online field setting. Care has to be taken in the design of the objective function to reflect which setting is being used.

We urge the reader to be careful when interpreting the expectation operator $\mathbb{E}$ in equation (37), which is typically a set of nested expectations that may be over a Bayesian prior (if appropriate), the results of an experiment while learning a policy, and the events that may happen while testing a policy.

We note that the term "base model" is not standard, although the concept is widely used in many, but not all, communities in stochastic optimization.

There is growing interest in replacing the expectation in our base model in (37) with a risk measure ρ. The risk measure may act on the total contribution (for example, penalizing contributions that fall below some target), but the most general version operates on the entire sequence of contributions, which we can write as

$$\max_{\pi} \rho(C_0(S_0, X^\pi(S_0), W_1), \ldots, C_T(S_T, X^\pi(S_T))). \tag{39}$$

The policy $X^\pi(S_t)$ might even be a robust policy such as that given in equation (27), where we might introduce tunable parameters in the uncertainty set $\mathcal{W}_t$. For example, we might let $\mathcal{W}_t(\theta)$ be the uncertainty set where θ captures the confidence that the noise (jointly or independently) falls within the uncertainty set. We can then use (37) as the basis for simulating our robust policy. This is basically the approach used in Ben-Tal et al. (2005), which compared a robust policy to a deterministic lookahead (without tuning the robust policy) by averaging the performance over many iterations in a simulator (in effect, approximating the expectation in equation (37)).

This opens up connections with a growing literature in stochastic optimization that addresses risk measures (see Shapiro et al. (2014) and Ruszczyński (2014) for nice introductions to dynamic risk measures in stochastic optimization). This work builds on the seminal work in Ruszczyński &

Shapiro (2006), which in turn builds on what is now an extensive literature on risk measures in finance (Rockafellar & Uryasev (2000), Rockafellar & Uryasev (2002), Kupper & Schachermayer (2009) for some key articles), with a general discussion in Rockafellar & Uryasev (2013). There is active ongoing research addressing risk measures in stochastic optimization (Collado et al. (2011), Shapiro (2012), Shapiro et al. (2013), Kozmík & Morton (2014), Jiang & Powell (2016*a*)). This work has started to enter engineering practice, especially in the popular area (for stochastic programming) of the management of hydroelectric reservoirs (Philpott & de Matos (2012), Shapiro et al. (2013)) as well as other applications in energy (e.g. Jiang & Powell (2016*b*)).

We refer to the base model in equation (37) (or the risk-based version in (39)), along with the transition function in equation (38), as our *universal formulation*, since it spans all the problems presented in section 2 (but, see the discussion in section 10). With this universal formulation, we have bridged offline (terminal reward) and online (cumulative reward) stochastic optimization, as well as state-independent and state-dependent functions.

With our general definition of a state, we can handle pure learning problems (the state variable consists purely of the distribution of belief about parameters), classical dynamic programs (where the "state" often consists purely of a physical state such as inventory), partially observable Markov decision processes, problems with simple or complex interperiod dependencies of the information state, and any mixture of these. In section 10, we are going to revisit this formulation and offer some additional insights.

Central to this formulation is the idea of optimizing over policies, which is perhaps the single most significant point of departure from most of the formulations presented in section 2. In fact, our finding is that many of the fields of stochastic optimization are actually pursuing a particular class of policies. In the next section, we provide a general methodology for searching over policies.

5. Designing policies

We begin by defining a policy as

Definition 5.1. *A* **policy** *is a rule (or function) that determines a feasible decision given the available information in state S_t (or S^n).*

We emphasize that a policy is *any* function that returns a (feasible) decision given the information in the state variable. A common mistake is to assume that a policy is some analytical function such as a rule (which is a form of lookup table) or perhaps a parametric function. In fact, it is often a carefully formulated optimization problem.

There are two fundamental strategies for creating policies:

Policy search - Here we use an objective function such as (37) to search within a family of functions to find a function that works best.

Lookahead approximations - Alternatively, we can construct policies by approximating the impact of a decision now on the future.

Either of these approaches can yield optimal policies, although this is rare. Below we show that each of these approaches are the basis of the two strategies for designing policies, producing four meta-classes that cover all of the approaches that have ever been used in the literature. These are described in more detail below.

5.1. Policy search

Policy search involves tuning and comparing policies using the objective function such as (37) or (39) so that they behave well over time, under whatever sources of uncertainty that we choose to model in our simulator (which can also be the real world). Imagine that we have a class of functions $\mathcal{F}$, where for each function $f \in \mathcal{F}$, there is a parameter vector $\theta \in \Theta^f$ that controls its behavior. Let $X^f(S_t|\theta)$ be a function in class $f \in \mathcal{F}$ parameterized by $\theta \in \Theta^f$. Policy search involves finding the best policy using

$$\max_{f\in\mathcal{F},\theta\in\Theta^f} \mathbb{E}\left\{\sum_{t=0}^{T} C_t(S_t, X^f(S_t|\theta), W_{t+1})|S_0\right\}. \tag{40}$$

If $\mathcal{F}$ includes the optimal policy architecture, and Θ^f includes the optimal θ for this function, then solving equation (40) would produce the optimal policy. There are special cases where this is true (such as (s, S) inventory policies). We might also envision the ultimate function class that can approximate any function such as deep neural networks or support vector machines, although these are unlikely to ever solve high dimensional problems that arise in logistics.

Since we can rarely find optimal policies using (40), we have identified two meta-classes:

Policy function approximations (PFAs) - Policy function approximations can be lookup tables, parametric or nonparametric functions, but the most common are parametric functions. This could be a linear function such as

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1\phi_1(S_t) + \theta_2\phi_2(S_t) + \ldots,$$

or a nonlinear function such as an order-up-to inventory policy, a logistics curve, or a neural network. Typically there is no guarantee that a PFA is in the optimal class of policies. Instead, we search for the best performance within a class.

Cost function approximations (CFAs) - A CFA is

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

where $\bar{C}_t^\pi(S_t, x|\theta)$ is a parametrically modified cost function, subject to a parametrically modified set of constraints. CFAs are widely used for solving large scale problems such as scheduling an airline or planning a supply chain. For example, we might introduce slack into a scheduling problem, or buffer stocks for an inventory problem. Below we show that popular policies for learning problems such as multiarmed bandits use CFAs.

Policy search is best suited when the policy has clear structure, such as inserting slack in an airline schedule, or selling a stock when the price goes over some limit. We may believe policies are smooth, such as the relationship between the release rate from a reservoir and the level of the reservoir, but often they are discontinuous such as an order-up-to policy for inventories.

5.2. Lookahead approximations

Just as we can, in theory, find an optimal policy using policy search, we can also find an optimal policy by modeling the downstream impact of a decision made now on the future. This can be written

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_\pi \mathbb{E}\left\{\sum_{t'=t+1}^{T} C(S_{t'},X_{t'}^\pi(S_{t'}))\middle| S_{t+1}\right\}\middle| S_t,x_t\right\}\right). \tag{41}$$

Equation (41) is daunting, but can be parsed in the context of a decision tree with discrete actions and discrete random outcomes (see figure 1). The states $S_{t'}$ correspond to nodes in the decision tree. The state S_t is the initial node, and the actions x_t are the initial actions. The first expectation is over the first set of random outcomes W_{t+1} (out of the outcome nodes resulting from each decision x_t).

After this, the policy π represents the action $x_{t'}$ that would be taken from every downstream node $S_{t'}$ for $t' > t$. Thus, a policy π could be a table specifying which action is taken from each potential downstream node, over the rest of the horizon. Then, the second expectation is over all the outcomes $W_{t'}$, $t' = t+2, \ldots, T$. Solving the maximization over all policies in (41) simply moves the policy search problem one time period later.

Not surprisingly, just as we can rarely find the optimal policy by solving the policy search objective function in (40), we can only rarely solve (41) (a decision tree is one example where we can). For this reason, a wide range of approximation strategies have evolved for addressing these two problems. These can be divided (again) into two meta-classes:

Value function approximations (VFAs) - Our first approach is to replace the entire term capturing the future in (41) with an approximation known widely as a value function approximation. We can do this in two ways. The first is to replace the function starting at S_{t+1} with a value function $V_{t+1}(S_{t+1})$ giving us

$$X_t^{VFA}(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{V_{t+1}(S_{t+1})|S_t\right\}\right) \tag{42}$$

where $S_{t+1} = S^M(S_t,x_t,W_{t+1})$, and where the expectation is over W_{t+1} conditioned on S_t (some write the conditioning as dependent on S_t and x_t). Since we generally cannot compute $V_{t+1}(S_{t+1})$, we can use various strategies to replace it with some sort of approximation $\overline{V}_{t+1}(S_{t+1})$, known as a value function approximation.

The second way is to approximate the function around the post-decision state S_t^x, which eliminates the expectation (42), giving us

$$X_t^{VFA}(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + V_t^x(S_t)\right). \tag{43}$$

The post-decision formulation is popular for problems where x_t is a vector, and $V_t^x(S_t^x)$ is a convex function of S_t^x.

Direct lookahead (DLAs) There are many problems where it is just not possible to compute sufficiently accurate VFAs (dynamic problems with forecasts is a broad problem class where this happens). When all else fails, we have to resort to a direct lookahead, where we replace the lookahead expectation and optimization in (41) with an approximate *model*. The most widely used strategy is to use a deterministic lookahead, but the field of stochastic programming will use a sampled future to create a more tractable version.

5.3. Notes

The four meta-classes of policies (PFAs, CFAs, VFAs, and DLAs) cover every policy considered in all the communities covered in section 2, with the possible exception of problems that can be solved exactly or using a sampled belief model (these are actually special cases of policies). We note that as of this writing, the "cost function approximation" has been viewed as more of an industry heuristic than a formal policy, but we believe that this is an important class of policy that has been overlooked by the research community (see Perkins & Powell (2017) for an initial paper on this topic).

It is natural to ask, why do we need four approximation strategies when we already have two approaches for finding optimal policies (equations (40) and (41)), either of which can produce an optimal policy? The reasons are purely computational. Equations (40) and (41) can rarely be solved to optimality. PFAs as an approximation strategy are effective when we have an idea of the structure of a policy, and these are typically for low-dimensional problems. CFAs similarly serve a role of allowing us to solve simplified optimization problems that can be tuned to provide good results. VFAs only work when we can design a value function approximation that reasonably approximates the value of being in a state. DLAs are a brute force approach where we typically resort to solving a simplified model of the future.

Below, we revisit the four classes of policies by first addressing learning problems, which are problems where the function being optimized does not depend on the state variable, and then in the much richer class of state-dependent functions. However, we are first going to touch on the important challenge of modeling uncertainty.

6. Learning challenges

Of the four classes of policies, only direct lookaheads do not involve any form of statistical learning. Of the remaining, there are five types of statistical learning problems:

- Learning an approximation $\overline{F}(x) \approx \mathbb{E}_W F(x, W)$. This is the easiest problem because we typically assume we have access to unbiased observations of $F(x, W)$. The goal is to minimize some measure of error between $\overline{F}(x)$ and $F(x, W)$.

- Learning policies $X^\pi(s)$. Here we are learning a function that maximizes a contribution or minimizes a cost, typically in the base model in equation (37).

- Learning a cost function approximation, which means a parametrically modified cost function or set of constraints. This is similar to learning $\overline{F}(x)$, except that we are learning a function embedded within a max or min operator.

- Learning a value function approximation $\overline{V}_t(S_t) \approx V_t(S_t)$

- Learning the state transition function - There are many problems where the transition function $S^M(S_t, x_t, W_{t+1})$ is not known (it might reflect human behavior, or a complex process such as climate change). We can use observations (S_t, x_t, S_{t+1}) to fit a statistical model $\bar{S}^M(S_t, x_t|\theta)$ to this data.

These learning challenges draw heavily on the fields of statistics and machine learning. Section 8.2 gave a very brief overview of general statistical methodologies and some references. There are several twists that make statistical learning in stochastic optimization a little different, including

- Recursive learning - Almost all of the statistical challenges listed above (approximate policy iteration being an exception) involve recursive learning. This means that we need methods that evolve from low to higher dimensional representations as we acquire more data.

- Active learning - We get to choose x (or the policy), which means we have control over what experiments to run. This means we usually are balancing the classic exploration-exploitation tradeoff.

- We may be optimizing a physical process or numerical simulation rather than a mathematical model. In these settings, observations of the function may be quite expensive, which means we do not have access to the large datasets that have become so familiar in a "big data" world.

- Learning value functions is one of the most difficult challenges from a statistical perspective, because we typically have to learn $\overline{V}_t(S_t)$ from observations $\hat{v}_t^n$ that are generally biased estimates of $V_t(S_t)$ (or its derivatives). The bias arises because we learn these values using suboptimal policies, but then we have to use our approximations.

- Policies are often discontinuous, as with buy low, sell high policies, or order-up-to inventory policies.

There is an extensive literature on learning. Hastie et al. (2009) is an excellent introduction to the broad field of statistical learning, but there are many good books. Jones (2001) and Montgomery (2000) provide thorough reviews of response surface methods. Kleijnen (2017) reviews regression and kriging metamodels for simulation models, which is the foundation of most stochastic optimization.

7. Modeling uncertainty

The community of stochastic optimization has typically focused on making good (or robust) decisions in the presence of some form of uncertainty. However, we tend to put a lot more attention into making a good decision than in the modeling of uncertainty.

The first step is to identify the sources of randomness. This can include observational errors, forecasting errors, model uncertainty, control uncertainty and even goal uncertainty (different decision-makers may have different expectations).

There is a field known as "uncertainty quantification" that emerged from within science and engineering in the 1960's (Smith (2014) and Sullivan (2015) are two recent books summarizing this area). This work complements the extensive work that has been done in the Monte Carlo simulation community which is summarized in a number of excellent books (good introductions are Banks et al. (1996), Ross (2002), Rubinstein & Kroese (2017)). Asmussen & Glynn (2007) provides a strong theoretical treatment.

It is important to recognize that if we want to find an optimal policy that solves (37), then we have to use care in how we model the uncertainties. There are different ways to representing an uncertain future, including

- Stochastic modeling - By far the most attention has been given to developing an explicit stochastic model of the future, which requires capturing:
  - Properties of probability distributions, which may be described by an exponential family (e.g. normal or exponential) and their discrete counterparts (Poisson, geometric), and heavy-tailed distributions. We can also use compound distributions such as Poisson distributions with random means, or mixtures such as jump diffusion models. It is often necessary to use nonparametric distributions derived from history.
  - Behavior over time - There are many ways to capture temporal behavior, including autocorrelation, crossing times (the length of time the actual is above or below a benchmark such as a forecast), regime switching, spikes, bursts and rare events.
  - Other relationships, such as spatial patterns, behaviors at different levels of aggregation.
- Distributionally robust modeling - There is growing attention given to the idea of using other methods to represent the future that do not require specific knowledge of a distribution (see Bayraksan & Love (2015) and Gabrel et al. (2014) for good reviews). Robust optimization uses uncertainty sets which is shown in (Xu et al., 2012) to be equivalent to a distributionally robust optimization problem. We note that while uncertainty sets offers a different way of approaching uncertainty, it introduces its own computational challenges (Goh & Sim (2010),Wiesemann et al. (2014)).
- No model - There are many applications where we simply are not able to model the underlying dynamics. These can be complex systems such as climate change, production plants, or the

behavior of a human. Different communities use terms such as model-free dynamic programming, data-driven stochastic optimization, or online control.

This is a very brief summary of a rich and complex dimension of stochastic optimization, but we feel it is important to recognize that modeling uncertainty is fundamental to the process of finding optimal policies. Stochastic optimization problems can be exceptionally challenging, and as a result we feel that most of the literature has focused on designing good policies. However, a policy will not be effective unless it has been designed in the context of a proper model, which means accurately capturing uncertainty.

8. Policies for state-independent problems

An important class of problems is where the function being maximized does not depend on any dynamic information that would be in the state variable. We can write these optimization problems as

$$\max_{x \in \mathcal{X}} \mathbb{E}F(x, W). \tag{44}$$

An example is the newsvendor problem

$$\max_{x} \mathbb{E}F(x, W) = \max_{x} \mathbb{E}(p \min\{x, W\} - cx), \tag{45}$$

where we order a quantity x at a unit cost c, then observe demand W and sell the minimum of these two at a price p. We assume we cannot compute $\mathbb{E}F(x, W)$ (perhaps the distribution of W is not known), so we will iteratively develop estimates $\overline{F}^n(x)$. We might let $S^n = \overline{F}^n(x)$ be our belief about $\mathbb{E}F(x, W)$. If we make a decision x^n and observe $F^{n+1} = F(x^n, W^{n+1})$, we can use this information to update our belief about $\mathbb{E}F(x, W)$. Thus, our state S^n only captures our belief about the function.

An example of a state-dependent problem would be one where the quantity x is constrained by $x \leq R^n$ where R^n is the available resources at iteration n, or where the price is p^n which is revealed before we make the decision x. In this case, our state variable might consist of $S^n = (R^n, p^n, \overline{F}^n(x))$. In this section, we assume that the state variable consists only of the belief about the function.

Below we describe adaptive algorithms where the state S^n at iteration n captures what we need to know to make a decision (that is, to calculate our policy), but which does not affect the function itself. However, we might be solving a time-dependent problem where the price p_t is revealed before we make a decision x_t at time t. In this case, p_t would enter our state variable, and we would have a state-dependent function.

We are going to design a sequential search procedure, which we can still model as a stochastic, dynamic system, but now the state S^n (after n iterations) captures the information we need to make a decision using some policy $X^\pi(S^n)$. We refer to this problem class as *learning problems*, and make the distinction between derivative-based and derivative-free problems.

8.1. Derivative-based

Assume we can compute a gradient $\nabla_x F(x, W)$ at a point $x = x^n$ and $W = W^{n+1}$, allowing us to implement a stochastic gradient algorithm of the form

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}), \tag{46}$$

where α_n is a stepsize that may adapt to conditions as they unfold. There are many choices of stepsize rules as reviewed in Powell & George (2006), with new and powerful rules given in Duchi et al. (2011) (AdaGrad), Kingma & Ba (2015) (Adam), and Orabona (2014) (PiSTOL). To illustrate the core idea, imagine we use Kesten's stepsize rule given by

$$\alpha_n = \frac{\theta}{\theta + N^n}, \tag{47}$$

where we might let N^n be the number of times that the gradient $\nabla_x F(x^n, W^{n+1})$ changes direction.

We now have a dynamic system (the stochastic gradient algorithm) that is characterized by a gradient and a "policy" for choosing the stepsize (47). The state of our system is given by $S^n = (x^n, N^n)$, and is parameterized by θ along with the choice of how the gradient is calculated, and the choice of the stepsize policy (e.g. Kesten's rule). Our policy, then, is a rule for choosing a stepsize α_n. Given α_n (and the stochastic gradient $\nabla_x F(x^n, W^{n+1})$), we sample W^{n+1} and then compute x^{n+1}. Thus, the updating equation (46), along with the updating of N^n, constitutes our transition function.

This simple illustration shows that a derivative-based stochastic gradient algorithm can be viewed as a stochastic, dynamic system (see Kushner & Yin (2003) for an in-depth treatment of this idea). Optimizing over policies means optimizing over the choice of stepsize rule (such as Kesten's rule (Kesten (1958)), BAKF (Powell & George (2006)), AdaGrad (Duchi et al. (2011)), Adam (Kingma & Ba (2015)), PiSTOL (Orabona (2014))) and the parameters that characterize the rule (such as θ in Kesten's rule above).

8.2. Derivative-free

We make the simplifying assumption that the feasible region $\mathcal{X}$ in the optimization problem (44) is a discrete set of choices $\mathcal{X} = \{x_1, \ldots, x_M\}$, which puts us in the arena of ranking and selection (if we wish to maximize the terminal reward), or multiarmed bandit problems (if we wish to maximize the cumulative reward). The discrete set might represent a set of drugs, people, technologies, paths over a network, or colors, or it could be a discretized representation of a continuous region. Not surprisingly, this is a tremendously broad problem class. Although it has attracted attention since the 1950's (and earlier), the first major reference on the topic is DeGroot (1970), who also characterized the optimal policy using Bellman's equation, although this could not be computed. Since this time, numerous authors have worked to identify effective policies for solving the optimization problem in (28).

Central to derivative-free stochastic search is the design of a belief model. Let $\overline{F}^n(x) \approx \mathbb{E}F(x, W)$ be our approximation of $\mathbb{E}F(x, W)$ after n experiments. We can represent $\overline{F}^n(x)$ using any of the following architectures.

Lookup tables Let $\mu_x = \mathbb{E}F(x, W)$ be the true value of the function at $x \in \mathcal{X}$. A lookup table belief model would consist of estimates $\bar{\mu}_x^n$ for each $x \in \mathcal{X}$. If we are using a Bayesian belief model, we can represent the beliefs in two ways:

Independent beliefs We assume that μ_x is a random variable where a common assumption is $\mu_x \sim N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$, where $\bar{\sigma}_x^{2,n}$ is the variance in our belief about μ_x.

Correlated beliefs Here we assume we have a matrix Σ^n with element $\Sigma_{xx'}^n = Cov^n(\mu_x, \mu_{x'})$, where $Cov^n(\mu_x, \mu_{x'})$ is our estimate of the covariance after n observations.

Parametric models The simplest parametric model is linear with the form

$$f(x|\theta) = \theta_0 + \theta_1\phi_1(x) + \theta_2\phi_2(x) + \ldots$$

where $\phi_f(x),\ f \in \mathcal{F}$ is a set of features drawn from the decision x (and possibly other exogenous information). We might let $\bar{\theta}^n$ be our time n estimate of θ, and we might even have a covariance matrix $\Sigma^{\theta,n}$ that is updated as new information comes in. Parametric models might be nonlinear in the parameters (such as a logistic regression), or a basic (low dimensional) neural network.

Nonparametric models These include nearest neighborhood and kernel regression (basically smoothed estimates of observations close to x), support vector machines, and deep (high dimensional) neural networks.

If we let S^n be our belief state (such as point estimates and covariance matrix for our correlated belief model), we need a policy $X^\pi(S^n)$ to return the choice x^n of experiment to run, after which we make a noisy observation of our unknown function $\mathbb{E}f(x, W)$. We represent this noisy experiment by W^{n+1}, which we may view as returning a sampled observation $F(x^n, W^{n+1})$, or a noisy observation $W^{n+1} = f(x^n) + \varepsilon^{n+1}$ where $f(x)$ is our true function. This leaves us with the problem of identifying good policies $X^\pi(S)$.

A number of policies have been proposed in the literature. We can organize these into our four classes of policies, although the most popular are cost function approximations (CFAs) and single-period, direct lookaheads (DLAs). However, we use this setting to illustrate all four classes:

Policy function approximations - For learning problems, assume we have some policy for making a decision. Imagine that the decision is continuous, such as a price, amount to order, or the forces applied to a robot or autonomous vehicle. This policy could be a linear rule (that is, an "affine policy"), or a neural network which we denote by $Y^\pi(S)$. Assume that after making the decision, we use the resulting performance to update the rule. For this reason, it helps to introduce some exploration by introducing some randomization which we might do using

$$X^\pi(S) = Y^\pi(S) + \varepsilon.$$

The introduction of the noise $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ is referred to in the controls literature as "excitation." The variance σ_ε^2 is a tunable parameter.

Cost function approximations - This is the most popular class of policies, developed primarily in the setting of online (cumulative reward) problems known as multiarmed bandit problems. Examples include:

Pure exploitation - These policies simply choose what appears to be best, such as

$$X^{Xplt}(S^n) = \arg\max_x \bar{\mu}_x^n. \tag{48}$$

We might instead have a parametric model $f(x|\theta)$ with unknown parameters. A pure exploitation policy (also known as "simple greedy") would be

$$\begin{aligned} X^{Xplt}(S^n) &= \arg\max_x f(x, \bar{\theta}^n), \\ &= \arg\max_x f(x, \mathbb{E}(\theta|S^n)). \end{aligned}$$

This policy includes any method that involves optimizing an approximation of the function such as linear models, often referred to as response surface methods (Ginebra & Clayton (1995)).

Bayes greedy - This is basically a pure exploitation policy where the expectation is taken outside the function. For example, assume that our true function is a parametric function $f(x|\theta)$ with an unknown parameter vector θ. The Bayes greedy policy would be

$$X^{BG}(S^n) = \arg\max_x \mathbb{E}\{f(x,\theta)|S^n\}. \tag{49}$$

Interval estimation - This is given by

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x (\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n). \tag{50}$$

where $\bar{\sigma}_x^n$ is the standard deviation of the estimate $\bar{\mu}_x^n$.

Upper confidence bounding - There is a wide range of UCB policies that evolved in the computer science literature, but they all have the generic form

$$X^{UCB}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB}\sqrt{\frac{\log n}{N_x^n}}\right), \tag{51}$$

where N_x^n is the number of times we have tried alternative x. We first introduced UCB policies in equation (32) where we used $4\sigma^W$ instead of the tunable parameter θ^{UCB}. UCB policies are very popular in the research literature (see, for example, Bubeck & Cesa-Bianchi (2012)) where it is possible to prove bounds for specific forms, but in practice it is quite common to introduce tunable parameters such as θ^{UCB}.

Value functions - It is possible in principle to solve learning problems using value functions, but these are rare and seem to be very specialized. This would involve a policy of the form

$$X^{VFA}(S^n) = \arg\max_x \left(\bar{\mu}_x^n + \mathbb{E}\{V^{n+1}(S^{n+1})|S^n, x\}\right), \tag{52}$$

where S^n (as before) is our state of knowledge. There are special cases where S^n is discrete, but if S^n is, for example, a set of point estimates $\bar{\mu}_x^n$ and variances $\bar{\sigma}_x^{2,n}$, then $S^n = (\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})_{x\in\mathcal{X}}$ which is high-dimensional and continuous. Value functions are the foundation of Gittins indices (see section 2.14), which are calculated by decomposing multi-armed bandit problems into a series of single-arm problems which allows the value functions to be computed.

Direct lookahead policies - It is important to distinguish between single-period lookahead policies (which are quite popular), and multi-period lookahead policies:

Single period lookahead - Examples include

Knowledge gradient - This estimates the value of information from a single experiment. Assume we are using a parametric belief model where $\bar{\theta}^n$ is our current estimate, and $\bar{\theta}^{n+1}(x)$ is our updated estimate if we run experiment $x^n = x$. Keeping in mind that $\bar{\theta}^{n+1}(x)$ is a random variable at time n when we choose to run the experiment, the value of the experiment, measured in terms of how much better we can find the best decision, is given by

$$\nu^{KG,n}(x) = \mathbb{E}_\theta E_{W|\theta}\{\max_{x'} f(x'|\bar{\theta}^{n+1}(x))|S^n\} - \max_{x'} f(x'|\bar{\theta}^n).$$

The knowledge gradient was first studied in depth in Frazier et al. (2008) for independent beliefs, and has been extended to correlated beliefs (Frazier et al., 2009), linear beliefs (Negoescu et al., 2010), nonlinear parametric belief models (Chen et al., 2015), nonparametric beliefs (Barut & Powell (2014), Cheng et al. (2014)), and hierarchical beliefs (Mes et al., 2011). These papers all assume that the variance of measurements is known, an assumption that is relaxed in Chick et al. (2010). The knowledge gradient seems to be best suited for settings where experiments are expensive, but care has to be taken when experiments are noisy, since the value of information may become non-concave. This is addressed in Frazier & Powell (2010).

Expected improvement - Known as EI in the literature, expected improvement is a close relative of the knowledge gradient, given by the formula

$$\nu_x^{EI,n} = \mathbb{E}\left[\max\left\{0, \mu_x - \max_{x'} \bar{\mu}_{x'}^n\right\}\middle| S^n, x^n = x\right]. \tag{53}$$

Expected improvement maximizes the degree to which the current belief about the function at x might exceed the current estimate of the maximum. Like the knowledge gradient, is a form of value-of-information policy (see e.g. Chick et al. (2010)), with the difference that EI captures the improvement in the function at a point x, while the knowledge gradient captures the improvement due to a change in the decision resulting from improved estimates.

Sequential kriging - This is a methodology developed in the geosciences to guide the investigation of geological conditions, which are inherently continuous where x may have two or three dimensions (see Cressie (1990) for the history of this approach). Although the method is popular and relatively simple, for reasons of space, we refer readers to Stein (1999) and Powell & Ryzhov (2012) for introductions. This work is related to efficient global optimization (EGO) (Jones et al., 1998), and has been applied to the area of optimizing simulations (see Ankenman et al. (2010) and the survey in Kleijnen (2014)).

Thompson sampling - First introduced in Thompson (1933), Thompson sampling works by sampling from the current belief about $\mu_x \sim N(\mu_x^n, \bar{\sigma}_x^{2,n})$, which can be viewed as the prior distribution for experiment $n+1$. Let $\hat{\mu}_x^n$ be this sample. The Thompson sampling policy is then

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^n.$$

Thompson sampling can be viewed as a form of randomized interval estimation, without the tunable parameter (we could introduce a tunable parameter by sampling from $\mu_x \sim N(\mu_x^n, \theta^{TS}\bar{\sigma}_x^{2,n})$). Thompson sampling has attracted considerable recent interest from the research community (Agrawal & Goyal, 2012) and has sparked further research in posterior sampling (Russo & Van Roy, 2014).

Multiperiod lookahead - Examples include

Decision tree - Some sequential decision problems (for example, with binary outcomes) can be computed exactly for small budgets (say, up to seven experiments). Decision trees can directly model the belief state. Larger problems can be approximated using techniques such as Monte Carlo tree search.

The KG(*) - policy There are many settings where the value of information is nonconcave, such as when experiments are very noisy (experiments with Bernoulli outcomes fall in this category). For this setting, Frazier & Powell (2010) proposes to act as if alternative x is going to be tested n_x times, and then find n_x to maximize the *average* value of information.

8.3. Discussion

We note in closing that we did not provide a similar list of policies for derivative-based problems. A stochastic gradient algorithm would be classified as a policy function approximation. Wu et al. (2017) appears to be the first to consider using gradient information in a knowledge gradient policy.

9. Policies for state-dependent problems

While state-independent learning problems are an important problem class, they pale in comparison to the vast range of state-dependent functions, which includes the entire range of problems known generally as "resource allocation." Since it helps to illustrate ideas in the context of an example, we are going to use a relatively simple energy storage problem, where energy is stored in the battery for a system which can get energy from a wind farm (where the price is free), the grid (which has unlimited capacity but highly stochastic prices) to serve a predictable, time-varying load.

This example is described in more detail in Powell & Meisel (2016*b*) which shows for this problem setting that each of the four classes may work best depending on the characteristics of the system.

9.1. Policy function approximations

A basic policy for buying energy from and selling energy to the grid from a storage device is to buy when the price p_t falls below a buy price θ^{buy}, and to sell when it goes above a sell price θ^{sell}.

$$X^\pi(S_t|\theta) = \begin{cases} -1 & \text{If } p_t < \theta^{buy}, \\ 0 & \text{If } \theta^{buy} \leq p_t \leq \theta^{sell}, \\ 1 & \text{If } p_t > \theta^{sell}. \end{cases}$$

This is a policy that is nonlinear in θ. A popular PFA is one that is linear in θ, often referred to as an "affine policy" or a "linear decision rule," which might be written as

$$X^\pi(S_t|\theta) = \theta_0\phi_0(S_t) + \theta_1\phi_1(S_t) + \theta_2\phi_2(S_t). \tag{54}$$

Recently, there is growing interest in tapping the power of deep neural networks to represent a policy. In this context, the policy π would capture the structure of the neural network (the number of layers and dimensionality of each layer), while θ would represent the weights, which can be tuned using a gradient search algorithm.

These are examples of stationary policies, which is to say that while the function depends on a dynamically varying state S_t, the function itself does not depend on time. While some authors will simply add time to the state variable as a feature, in most applications (such as energy storage), the policy will not be monotone in time. It is possible to make $\theta = (\theta^{buy}, \theta^{sell})$ time dependent, in which case we would write it as θ_t, but now we have dramatically increased the number of tunable parameters (Moazeni et al. (2017) uses splines to simplify this process).

9.2. Cost function approximations

A cost function approximation is a policy that solves a modified optimization problem, where either the objective function or the constraints can be modified parametrically. A general way of writing this is

$$X^{CFA}(S_t|\theta) = \arg\max_{x \in \mathcal{X}^\pi(\theta)} \bar{C}^\pi(S_t, x|\theta). \tag{55}$$

A simple CFA uses a linear modification of the objective function which we can write as

$$X_t^{CFA}(S_t|\theta) = \arg\max_{x\in\mathcal{X}_t}\left(C(S_t,x) + \sum_{f\in\mathcal{F}}\theta_f\phi_f(S_t,x)\right), \tag{56}$$

where the term added to $C(S_t,x)$ is a "cost function correction term," which requires designing basis functions $(\phi_f(S_t,x))$, $f\in\mathcal{F}$, and tuning the coefficients θ.

A common strategy is to introduce modifications to the constraints. For example, a grid operator planning energy generation for tomorrow will introduce extra reserve by scaling up the forecast. Airlines will optimize the scheduling of aircraft, handling uncertainty in travel times due to weather by introducing schedule slack. Both of these represent modified constraints, where the extra generation reserve or schedule slack represent tunable parameters, which may be written

$$X_t^{CFA}(S_t|\theta) = \arg\max_{x\in\mathcal{X}_t^\pi(\theta)} C(S_t,x), \tag{57}$$

where $\mathcal{X}_t^\pi(\theta)$ might be the modified linear constraints

$$\begin{aligned} A_t x &= b_t + D_t\theta, \\ x &\geq 0. \end{aligned} \tag{58}$$

Here, θ is a vector of tunable parameters and D is an appropriate scaling matrix. Using the creative modeling for which the linear programming community has mastered, equation (58) can be used to introduce schedule slack into an airline schedule, spinning reserve into the plan for energy generation, and even buffer stocks for managing a supply chain.

9.3. Value function approximations

We begin by recalling the optimal policy based on calculating the impact of a decision now on the future (originally given in equation (41)),

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_\pi \mathbb{E}\left\{\sum_{t'=t+1}^{T} C(S_{t'},X_{t'}^\pi(S_{t'}))\,\middle|\, S_{t+1}\right\}\middle|\, S_t,x_t\right\}\right). \tag{59}$$

We let $V_{t+1}(S_{t+1})$ be the expected optimal value of being in state S_{t+1}, allowing us to write equation (59) as

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{V_{t+1}(S_{t+1})|S_t,x_t\right\}\right). \tag{60}$$

The problem with equation (60) is that we typically cannot compute the value function $V_{t+1}(S_{t+1})$. Section 2.6 provided a brief introduction of how to replace the exact value function with an approximation $\overline{V}_{t+1}(S_{t+1})$ which would give us the policy

$$X_t^{VFA}(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\overline{V}_{t+1}(S_{t+1})|S_t,x_t\right\}\right).$$

There are many problems where we cannot compute the expectation, so we might instead compute the value function around the post-decision state S_t^x, giving us

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \overline{V}_t(S_t^x)\right).$$

A substantial field has grown up around approximating value functions, typically under the umbrella of approximate dynamic programming (Powell, 2011), or reinforcement learning (Sutton & Barto, 1998) (see also Szepesvári (2010)). Beyond the brief introduction we provided in section 2.6, we refer the reader to these references as a starting point.

There is an entire literature that focuses on settings where x_t is a vector, and the contribution function $C(S_t, x_t) = c_t x_t$, where the constraints $\mathcal{X}_t$ are a set of linear equations. These problems are most often modeled where the only source of randomness is in exogenous supplies and demands. In this case, the state S_t consists of just the resource state R_t, and we can also show that the post-decision value function $\overline{V}_t^x(R_t)$ is concave (if maximizing). These problems arise often in the management of resources to meet random demands.

Such problems have been solved for many years by representing the value function as a series of multidimensional cuts based on Benders decomposition, building on ideas first presented in (Van Slyke & Wets, 1969) (which required enumerating all the cuts) and (Higle & Sen, 1991) (which used a sample-based procedure). Building on these ideas, Pereira & Pinto (1991) proposed stochastic dual dynamic programming, or SDDP, as a way of solving sequential problems (motivated by the challenge of optimizing water reservoirs in Brazil).

This strategy has spawned an entire body of research (Infanger & Morton (1996), Shapiro et al. (2013), Sen & Zhou (2014), Girardeau et al. (2014)) which is reviewed in Shapiro et al. (2014). It is now recognized that SDDP is a form of approximate dynamic programming in the context of convex, stochastic linear programming problems (see e.g. Powell (2007)). Related to SDDP is the use of separable, piecewise linear value function approximations that have proven useful in large scale logistics applications (Powell et al. (2004), Topaloglu & Powell (2006), Bouzaiene-Ayari et al. (2014), Salas & Powell (2015)).

9.4. Direct lookahead approximations

Each of the policies described above (PFAs, CFAs, and VFAs) require approximating some function, drawing on the tools of machine learning. These functions may be the policy $X^\pi(S_t)$, an approximation of $\mathbb{E}F(x, W)$, a modified cost function or constraints (for CFAs), or the value of being in a state $V_t(S_t)$. These methods work when these functions can be approximated reasonably well.

Not surprisingly, this is not always possible, typically because we lack recognizable structure. When all else fails (which is quite often), we have to turn to direct lookaheads, where we need to approximate the lookahead policy in equation (41). Since this function is rarely computable, we approach it by replacing the model of the future with an approximation which we refer to as the *lookahead model.* A lookahead model is generated at a time t when we have to make decision x_t. There are five types of approximations that are typically made when we create a lookahead model:

- Limiting the horizon - We may reduce the horizon from (t, T) to $(t, t+H)$, where H is a horizon that is just long enough to produce a good decision at time t.
- Stage aggregation - A stage is a sequence of seeing new information followed by making a decision. A popular strategy is to replace the full multistage formulation with a two-stage formulation, consisting of making a decision x_t now, then seeing all the information over the remainder of the horizon, represented by $W_{t+1}, \ldots, W_{t+H}$, and then making all the decisions over the horizon $x_{t+1}, \ldots, x_{t+H}$. This means that x_{t+1} is allowed to "see" the entire future.
- Approximating the stochastic process - We may replace the full probability model with a sampled set of outcomes, often referred to as scenarios. We may also replace a state-dependent stochastic process with one that is state-independent.
- Discretization - Time, states, and decisions may all be discretized in a way that makes the resulting model more computationally tractable. The resulting stochastic model may even be solvable using backward dynamic programming.
- Dimensionality reduction - It is very common to ignore one or more variables in the lookahead model. For example, it is virtually always the case that a forecast will be held fixed in a lookahead model, while it would be expected to evolve over time in a real application (and hence in the base model). Alternatively, a base model with a belief state, capturing imperfect knowledge about a parameter, might be replaced with an assumption that the parameter is known perfectly.

As a result of all these approximations, we have to create notation for what is basically an entirely new model, although there should be close parallels with the base model. For this reason, we use the same notation as the base model, but all variables are labeled with a tilde, and are indexed by both t (which labels the time at which the lookahead model is created), and t', which is the time within the lookahead model. Thus, a lookahead policy would be written

$$X_t^{LA}(S_t|\theta^{LA}) \; = \; \arg\max_{x_t} \left(C(S_t, x_t) + \tilde{\mathbb{E}} \left\{ \max_{\tilde{\pi} \in \tilde{\Pi}} \tilde{\mathbb{E}}^{\tilde{\pi}} \left\{ \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{X}_{tt'}^{\tilde{\pi}}(\tilde{S}_{tt'})) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \right). \quad (61)$$

Here, the parameter vector θ^{LA} is assumed to capture all the choices made when creating the approximate lookahead model. We note that in lookahead models, the tunable parameters (horizons, number of stages, samples) are all of the form "bigger is better," so tuning is primarily a tradeoff between accuracy and computational complexity.

Below we describe three popular strategies. The first is a deterministic lookahead model, which can be used for problems with discrete actions (such as a shortest path problem) or continuous vectors (such as a multiperiod inventory problem). The second is a stochastic lookahead procedure developed in computer science that can only be used for problems with discrete actions. The third is a strategy developed by the stochastic programming community for stochastic lookahead models with vector-valued decisions.

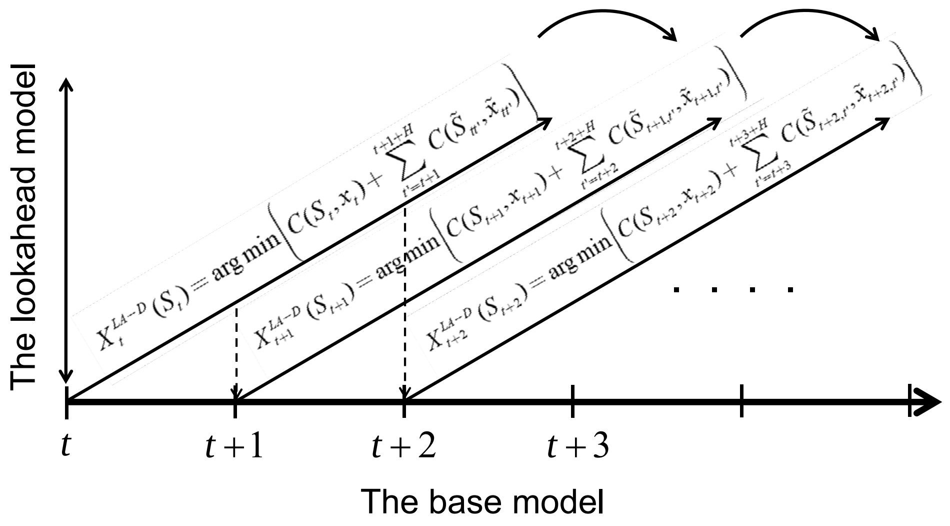
The lookahead model
$X_t^{LA-D}(S_t) = \arg\min\left(C(S_t, x_t) + \sum_{t'=t+1}^{t+1+H} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})\right)$
$X_{t+1}^{LA-D}(S_{t+1}) = \arg\min\left(C(S_{t+1}, x_{t+1}) + \sum_{t'=t+2}^{t+2+H} C(\tilde{S}_{t+1,t'}, \tilde{x}_{t+1,t'})\right)$
$X_{t+2}^{LA-D}(S_{t+2}) = \arg\min\left(C(S_{t+2}, x_{t+2}) + \sum_{t'=t+3}^{t+3+H} C(\tilde{S}_{t+2,t'}, \tilde{x}_{t+2,t'})\right)$
. . . .
t
$t+1$
$t+2$
$t+3$
The base model

Figure 2: Illustration of simulating a direct lookahead policy, using a deterministic model of the future.

Deterministic lookaheads

Easily the most popular lookahead model uses a deterministic approximation of the future, which we might write

$$X_t^{LA-Det}(S_t|\theta^{LA}) = \arg\max_{x_t}\left(C(S_t, x_t) + \max_{\tilde{x}_{t,t+1},\ldots,\tilde{x}_{t,t+H}} \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})\right), \tag{62}$$

where the optimization problem is solved subject to any constraints that would have been built into the policy.

The problem being modeled in (62) could be a shortest path problem, in which case we would likely solve it as a deterministic dynamic program. If x_t is a continuous vector (for example, optimizing cash flows or a supply chain problem), then (62) would be a multiperiod linear program.

Figure 2 illustrates the process of solving a lookahead model which yields a decision x_t which is implemented in the base model. The horizontal axis describes time moving forward in the base model, while the slanted lines represent the lookahead model projecting into the future. At each point in time (we represent t, $t+1$ and $t+2$) we solve the lookahead model, which consists of state variables $\tilde{S}_{tt'}$ and decision variables $\tilde{x}_{tt'}$ (for the lookahead model solved at time t), which returns a decision x_t that is implemented in the base model. We then use the base transition function $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ where W_{t+1} is sampled from the stochastic (base) model, or observed from a physical system. At time $t+1$, we repeat the process.

We note that the strategy of using a deterministic lookahead is often referred to as *model predictive control* (or MPC), which is to say that we use a model of the problem (more precisely an approximate model) to decide what to do now. The association of MPC with a deterministic lookahead reflects the history of MPC coming from the engineering controls community that predominantly focuses on deterministic problems. The term "model predictive control" actually refers to any lookahead model, whether it is deterministic or stochastic. However, stochastic lookahead models that match the base model are rarely solvable, so we are usually using most if not all of the five types of approximations listed above. For good reviews of model predictive control, see Morari et al. (2002), Camacho & Bordons (2003), Bertsekas (2005), and Lee (2011).

Rollout policies

A powerful and popular strategy is to interpret the search over a restricted set of policies in the future, represented as $\tilde{\pi} \in \tilde{\Pi}$ in equation (61). The design of these policies is highly problem-dependent and is best illustrated using examples:

- The time t problem could be the simultaneous assignment of drivers to riders at time t, where an assignment might take a driver at location i to location j. We might then estimate the value of the driver at j by myopically assigning this driver to simulated loads in the future (ignoring all other drivers).
- To solve a time-dependent inventory problem (such as planning inventories before Christmas), imagine testing different ordering decisions now (imagine we have to play orders four weeks in advance). Each decision is evaluated by simulating a simple replenishment rule in the future, to help us evaluate our ordering decision now.

The approximate rollout policy may be a parameterized policy $\tilde{X}^{\tilde{\pi}}(\tilde{S}_{tt'}|\tilde{\theta})$ that is typically fixed in advance (see Bertsekas & Castanon (1999) for a careful early analysis of this idea), but the choice of rollout policy can (and should) be optimized as part of the search over policies in our base model (37). In fact, the best choice of the parameter vector $\tilde{\theta}$ depends on the initial post-decision state S_t^x, which means we could even tune the parameter to find $\tilde{\theta}(S_t^x)$ on the fly (unlikely this would ever be done in practice). Thus, the search over $\tilde{\pi}$ in (61) could be a search for the best $\tilde{\theta}(S_t^x)$.

Monte Carlo tree search for discrete decisions

Imagine that we have discrete actions $a_t \in \mathcal{A}_s$ when we are in state $s = S_t$, after which we observe a realization of W_{t+1}. Such problems can be modeled in theory as classical decision trees, but these explode very quickly with the number of time periods.

Monte Carlo tree search is a strategy that evolved within computer science to explore a tree without enumerating the entire tree. This is done in four steps as illustrated in figure 3. These steps include a) selecting an action out of a decision node (which represents a state $\tilde{S}_{tt'}$), b) expanding the tree, if the resulting observation of $\widetilde{W}_{t,t'+1}$ results in a node that was not already in the tree, c) the rollout policy, which is how we evaluate the value of the node that we just reached out to, and d) backup,

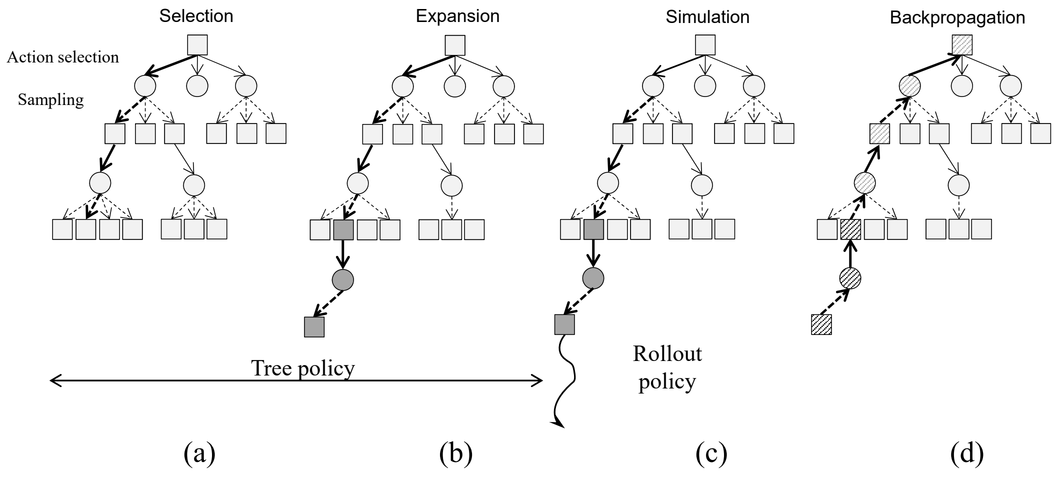

Figure 3: Sketch of Monte Carlo tree search, illustrating (left to right): selection, expansion, simulation and backpropagation.

where we run backward through the tree, updating the value of being at each node (this is what we did in equation (17)).

Central to the success of MCTS is having an effective rollout policy to get an initial approximation of the value of being in a leaf node. Rollout policies were originally introduced and analyzed in Bertsekas & Castanon (1999). A review of Monte Carlo tree search is given in Browne et al. (2012), although this is primarily for deterministic problems. Other recent reviews include Auger et al. (2013) and Munos (2014). Jiang et al. (2017) presents an asymptotic proof of convergence of MCTS if the lookahead policy uses the principle of information relaxation, which is done by taking a sample of the future and then solving the resulting deterministic problem assuming we are able to look into the future.

Monte Carlo tree search represents a relatively young algorithmic technology which has proven successful in a few applications. It is basically a brute force solution to the problem of designing policies, which depends heavily on the ability to design effective, but easy-to-compute, rollout policies.

Two-stage stochastic programming for vector-valued decisions

Monte Carlo tree search requires the ability to enumerate all of the actions out of a decision node. This limits MCTS to problems with at most a few dozen actions per state, and completely eliminates problems with vector-valued decisions.

A popular strategy (at least in the research literature) for solving sequential, stochastic linear programs is to simplify the lookahead model into three steps: 1) making the decision x_t to be implemented at time t, 2) sampling all future information $\widetilde{W}_{t,t+1}(\omega), \ldots, \widetilde{W}_{t,t+H}(\omega)$, where the sample paths ω are drawn from a sampled set $\tilde{\Omega}_t$ of sample paths of possible values of $\widetilde{W}_{t,t+1},\ldots,\widetilde{W}_{t,t+H}$, and 3) making all remaining decisions $\tilde{x}_{t,t+1}(\omega),\ldots,\tilde{x}_{t,t+H}(\omega)$. This produces the lookahead policy

$$X_t^{2stage}(S_t) = \arg\max_{x_t,(\tilde{x}_{tt'}(\omega))_{t'=t+1}^{t+H},\omega\in\tilde{\Omega}_t} c_t x_t + \sum_{\omega_t\in\tilde{\Omega}_t} \tilde{p}_t(\omega) \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\omega)\tilde{x}_{tt'}(\omega), \tag{63}$$

subject to first stage constraints

$$\begin{aligned} A_t x_t &= b_t, & (64)\\ x_t &\geq 0 \quad , & (65)\end{aligned}$$

and the second stage constraints for $\omega \in \tilde{\Omega}_t$,

$$\begin{aligned} \tilde{A}_{t,t+1}(\omega)\tilde{x}_{t,t+1}(\omega) + \tilde{B}_{t,t'-1}(\omega)x_t(\omega) &= \tilde{b}_{t,t+1}(\omega), & (66)\\ \tilde{A}_{tt'}(\omega)\tilde{x}_{tt'}(\omega) + \tilde{B}_{t,t'-1}(\omega)\tilde{x}_{t,t'-1}(\omega) &= \tilde{b}_{tt'}(\omega),\ t' = t+2,\ldots,t+H, & (67)\\ \tilde{x}_{tt'}(\omega) &\geq 0 \quad ,\ t' = t+1,\ldots,t+H. & (68)\end{aligned}$$

We again emphasize that ω determines the entire sequence $\widetilde{W}_{t,t+1},\ldots,\widetilde{W}_{t,t+H}$, which is how each decision $\tilde{x}_{tt'}(\omega)$ in the lookahead model is allowed to see the entire future. However, the here-and-now decision x_t is not allowed to see this information, which is viewed as an acceptable approximation in the research literature, although there has been virtually no analysis of the errors introduced by this assumption.

Since x_t is a vector, even deterministic versions of (63) (that is, where there is only a single ω) may be reasonably large. As a result, the full problem (63) - (68) when the set $\tilde{\Omega}_t$ contains tens to potentially hundreds of outcomes may be quite large. This has motivated the development of decomposition algorithms such as the progressive hedging algorithm of Rockafellar & Wets (1991), which replaces x_t with $x_t(\omega)$, which means that now even x_t is allowed to see the future, and then introduces the constraint

$$x_t(\omega) = \bar{x}_t,\ \forall\omega \in \tilde{\Omega}_t. \tag{69}$$

Equation (69) is widely known as a "non-anticipativity constraint" since it requires that x_t cannot be different for different outcomes ω. However, progressive hedging relaxes this constraint, producing series of much smaller optimization problems, one for each $\omega \in \tilde{\Omega}_t$, which are progressively modified until (69) is satisfied.

The literature on stochastic programming (as this field is known) dates to the 1950's with the original work of Dantzig (1955) and Dantzig & Ferguson (1956). This work has been followed by decades of research which is summarized in a series of books (Birge & Louveaux (2011), King & Wallace (2012), Shapiro et al. (2014)). As with all of our other policies, our two-stage stochastic programming policy $X^{2stage}(S_t)$ should be evaluated using our base model in equation (37), although this is often overlooked, primarily because computing $X^{2stage}(S_t)$, which requires solving the optimization problem

(63) - (68), can be quite difficult. As a result, the problem of carefully choosing the set $\hat{\Omega}_t$ has attracted considerable attention, beginning with the seminal work of Dupacova et al. (2003) and Heitsch & Romisch (2009), with more recent work on uncertainty modeling (see the tutorial in Bayraksan & Love (2015)).

Given the challenges of solving practical two-stage stochastic programming problems, full multistage lookahead models have attracted relatively little attention (Defourny et al. (2013) is a sample). We note that Monte Carlo tree search, by contrast, is a full "multistage" stochastic lookahead model, but it fully exploits the relative simplicity of small action spaces.

Robust optimization

Robust optimization has been extended to multiperiod problems, just as the two-stage stochastic programming model has been extended to multiperiod problems as an approximate way of solving (robustly) sequential decision problems. Assume we are trying to find x_t by optimizing over a horizon $(t, t+H)$. Formulated as a robust optimization problem means solving

$$X_t^{RO}(S_t|\theta) = \arg \min_{x_t,\ldots,x_{t+H} \in \mathcal{X}_t} \max_{(w_t,\ldots,w_{t+H}) \in \mathcal{W}_t(\theta)} \sum_{t'=t}^{t+H} c_t(w_t) x_t, \tag{70}$$

possibly subject to constraints that depend on $(w_t, \ldots, w_{t+H})$. Note that we are using $w_{t'}$ rather than ω or $W_{t'}(\omega)$, since $w_{t'}$ is now a decision variable.

This strategy was proposed in Ben-Tal et al. (2005) to solve a supply chain problem. While not modeled explicitly, the policy was then tested in an expectation-based simulator (what we call our base model).

9.5. Hybrid policies

There are two reasons to articulate the four meta-classes of policies. First, all four classes have problems for which they are well suited. If you only learn one class (as many students of stochastic optimization do), you are going to be limited to working on problems that are suited to that class. In fact, the best policy, even within the context of a single problem domain, can depend on the characteristics of the data. This property is illustrated in Powell & Meisel (2016*b*) for an energy storage problem, where each of the four classes of policies (plus a fifth hybrid) is shown to work best on a particular version of the problem.

The second reason is that it is often the case that the best policy is a hybrid of two, or even three, of the four classes. Below are some examples of hybrid policies we have encountered.

- Lookahead and VFA policies - Tree search can be a powerful strategy, but it explodes exponentially with the number of stages. Value functions avoid this, but requires that we develop accurate approximations of the value of being in a state, which can be hard in many applications. Consider now a partial tree search over a short horizon, terminating with a value function. Now the value function does not have to be quite as accurate, and yet we still get an approximation that extends over a potentially much longer horizon.

- Deterministic lookaheads (DLA) with tunable parameters (CFA) - A common industry practice is to solve a deterministic lookahead model, but to introduce tunable parameters to handle uncertainty. For example, airlines might introduce schedule slack to handle the uncertainty of weather delays, while a grid operator will schedule extra generation capacity to handle unexpected generator failures. These tunable parameters are optimized in the base model in equation (37), where the transition function (38) might be a simulator, or the real world.
- Any optimization-based policy (CFA, VFA or DLA) guided by historical patterns (a form of PFA) - Cost-based optimization models easily handle very high-dimensional data (e.g. optimizing a fleet of trucks or planes), but it can be hard to capture some issues in a cost function (we like to put drivers that work in teams on longer loads, but this is not a hard constraint).

The choice of the best policy, or hybrid, always depends on comparisons using the base model (37)-(38).

Discussion

There is widespread confusion in the research literature regarding the distinction between stochastic lookahead policies (primarily), and stochastic base models. While all policies should be tested in a base model (which can be the real world), tuning in a base model is essential when using PFAs and CFAs, but not with lookahead policies. As a result, many authors will present a stochastic lookahead model without making the distinction of whether this is a lookahead model, or a base model.

In some cases it is clear that a stochastic model is a lookahead model, such as a two-stage stochastic programming approximation of a multiperiod (and multistage) stochastic optimization problem. However, it is possible to solve a stochastic lookahead model as a dynamic program, in which case it may not be clear. We might look for approximations that are typical in lookahead models, but base models use approximations too.

10. A classification of problems

Having organized policies into four classes, we need to address the problem of evaluating policies. For this purpose, we have to recognize that there are different problem classes that introduces different issues for policy evaluation. We first make the distinction between problems where we only care about the final design (as would occur if we are experimenting in a lab) versus problems where we learn by doing in the field, in which case we have to maximize the cumulative rewards. The first objective is *offline* since we are working in a lab or simulated environment, while the second is *online* since we are adapting in a field setting.

It turns out that the machine learning community also uses these terms, but with different meanings. In machine learning, "offline" refers to batch learning, where we have to fit a model using a dataset that has already been generated. By contrast, "online" refers to sequential, since this is what would happen if we were learning in the field. The problem is that there are many uses of sequential algorithms in offline settings. For this reason, we use *terminal reward* to refer to problems where we are only

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{imp}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

Table 1: Comparison of formulations for state-independent (learning) vs. state-dependent problems, and offline (terminal reward) and online (cumulative reward).

interested in the performance of the final design, and *cumulative reward* when we need to maximize performance as we are progressing.

We begin by identifying two key dimensions for characterizing any adaptive optimization problem: First, whether the objective function is offline (terminal reward) or online (cumulative reward), and second, whether the objective function is state-independent (learning problems) or state-dependent (traditional dynamic programs). This produces four problem classes which are depicted in table 1. Moving clockwise around the table, starting from the upper left-hand corner:

Class 1) State-independent, terminal reward - This is our classic stochastic search problem evaluated using a finite budget (as it should be), where the problem is to find the best policy (which could be a stochastic gradient algorithm) for finding the design $x^{\pi,N}$ produced by the policy π within the experimental budget N. This might be called the finite-time version of the newsvendor problem, where the expectation can be written in nested form as

$$\max_\pi \mathbb{E}\{F(X^{\pi,N}, \widehat{W})|S^0\} \quad = \quad \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0} F(X^{\pi,N}, \widehat{W}), \tag{71}$$

where $W^1, \ldots, W^N$ are the observations of W while learning the function, and $\widehat{W}$ is the random variable used for testing the final design $x^{\pi,N}$. The initial state S^0 may be deterministic, but might include a Bayesian prior of an unknown parameter (such as the response of demand to price), which means we have to take an expectation over this distribution.

Class 2) State-independent, cumulative reward - Here we want a policy that learns while it optimizes, where we have to live with the performance of the decisions we make while we are learning the function. This would be our classic multiarmed bandit problem if the decisions x were discrete and we did not have access to derivatives (but we are not insisting on these limitations). Expanding the expectation gives us

$$\max_\pi \mathbb{E}\left\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})|S^0\right\} \quad = \quad \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0} \sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1}). \tag{72}$$

Class 3) State-dependent, cumulative reward - At first glance this looks like a classical dynamic program (when expressed in terms of optimizing over policies), yet we see that it closely parallels the multiarmed bandit problem. This problem may include a belief state, but not necessarily. When we expand the expectation we obtain

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T-1} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\} \quad = \quad \mathbb{E}_{S_0}\mathbb{E}_{W_1,\ldots,W_T|S_0}\left\{\sum_{t=0}^{T-1} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\}. \tag{73}$$

In contrast with problem classes (1) and (2), we model the performance of the policy over time t, rather than iterations n as we did in (72) (which could have been written either way).

Class 4) State-dependent, terminal reward - Here we are looking for the best policy to learn a policy that will then be implemented. Our implementation policy $X^{\pi^{imp}}(S_t|\theta^{imp})$ parallels the implementation decision $x^{\pi,N}$ in (71), where $\theta^{imp} = \Theta^{\pi^{lrn}}(S|\theta^{lrn})$ is a parameter that is learned by the learning policy $\Theta^{\pi^{lrn}}(S|\theta^{lrn})$. The learning policy could be algorithms for learning value functions such as Q-learning, approximate value iteration or SDDP, or it could be a search algorithm for learning a PFA or CFA. The parameters θ^{imp} are parameters that determine the behavior of the implementation policy such as an approximate Q-factor $\bar{Q}(s,a)$, a Benders' cut, or the tunable parameter in a UCB policy.

When we have a state-dependent function, we have to take an additional expectation over the state variable when evaluating the policy. Keeping in mind that the implementation parameters θ^{imp} are a function of the learning policy π^{lrn}, we can write this as

$$\begin{aligned}\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{imp}}(S|\theta^{imp}), \widehat{W})|S^0\} \quad &= \\ &\mathbb{E}_{S^0}\mathbb{E}^{\pi^{lrn}}_{W^1,\ldots,W^N|S^0}\mathbb{E}^{\pi^{imp}}_{S|S^0}\mathbb{E}^{\pi^{imp}}_{\widehat{W}|S^0} C(S, X^{\pi^{imp}}(S|\theta^{imp}), \widehat{W}).\end{aligned} \tag{74}$$

where $W^1,\ldots,W^N$ represents the observations made while using our budget of N experiments to learn a policy, and $\widehat{W}$ is the random variable observed when evaluating the policy at the end. Computing the expectation $\mathbb{E}^{\pi^{imp}}_{S|S^0}$ over the states is typically intractable because it depends on the implementation policy (which of course depends on the learning policy). Instead, we can run a simulation over a horizon $t = 0,\ldots,T-1$ and then divide by T to get an average contribution per unit time. We can think of W^n as the set of realizations over a simulation, which we can write as $W^n = (W^n_1,\ldots,W^n_T)$. We can then write our learning problem as

$$\max_{\pi^{lrn}} \mathbb{E}_{S^0}\mathbb{E}^{\pi^{imp}}_{(W^n_t)^T_{t=1},n=1,\ldots,N|S^0}\left(\mathbb{E}^{\pi^{imp}}_{(\widehat{W}_t)^T_{t=1}|S^0}\frac{1}{T}\sum_{t=0}^{T-1} C(S_t, X^{\pi^{imp}}(S_t|\theta^{imp}), \widehat{W}_{t+1})\right). \tag{75}$$

Here, we are searching over learning policies, where the simulation over time replaces $F(x,W)$ in the state-independent formulation. The sequence $(W^n_t)^T_{t=1}, n = 1,\ldots,N$ replaces the sequence

$W^1, \ldots, W^N$ for the state-independent case, where we start at state $S_0 = S^0$. We then do our final evaluation by taking an expectation over $(\widehat{W}_t)_{t=1}^T$, where we again assume we start our simulations at $S^0 = S_0$.

This organization brings out relationships that have not been highlighted in the past. For example, while ranking and selection/stochastic search has been viewed as a fundamentally different problem class from multiarmed bandits, we see that they are really the same problem with different objectives (final reward versus cumulative reward). We also see that state-independent problems (learning problems) are closely related to state-dependent problems, which is the problem class typically associated with dynamic programming (although all of these problems are dynamic programs).

We have noted that most adaptive learning algorithms for dynamic programming (Q-learning, approximate dynamic programming, SDDP) fall under the category of state-dependent, final-reward in table 1, which suggests that the cumulative-reward, state-dependent case is a relatively overlooked problem class (excluding contextual bandits, which is a special case). Algorithms in this setting have to balance learning while making good decisions (the classic exploration-exploitation tradeoff). Some contributions to this problem class include the work of Duff (Duff et al. (1996) and Duff (2002)) which tried to adapt the theory of Gittins indices to Q-learning algorithms, and Ryzhov (Ryzhov & Powell (2010) and Ryzhov et al. (2017)) who developed both offline (final reward) and online (cumulative reward) adaptations of the knowledge gradient algorithm for state-dependent problems.

There is a substantial literature that makes the distinction between problems in classes (1) and (2), primarily because optimal policies and their behavior (and hence, theoretical properties) are quite different. By contrast, while there are communities doing (state dependent) dynamic programming in both offline and online settings, the algorithms (policies) used for each setting are fundamentally the same. Why is this? We believe it is because classes (1) and (2) are relatively simple, and lend themselves to finding theoretical results characterizing the behavior of policies, where the slight differences between (1) and (2) are important. By contrast, if you are focusing on designing algorithms to find optimal policies, the distinction between the final reward and cumulative reward objective functions is simply not that important. Imagine solving linear programs for deterministic versions of (3) and (4); the simplex algorithm will solve both of these.

11. Research challenges

The framework presented here brings a variety of perspectives from the different communities of stochastic optimization, which creates new opportunities for research. These include:

- Given the complexity of solving a stochastic lookahead model, most authors are happy just to get a solution. As a result, almost no attention has been devoted to analyzing the quality of a stochastic lookahead model. We need more research to understand the impact of the different types of errors that are introduced by the approximations discussed in section 5.2 when creating lookahead models.

- There has been a long tradition of solving problems with belief states as "partially observable Markov decision processes." At the same time, theoreticians have known for decades that dynamic programs with belief states can be modeled simply as part of the state variable (as we have done), which means that POMDPs are really just dynamic programs which can be solved with any of the four classes of policies. In fact, we have described policies designed for problems where the state variable is purely a belief state. We need to explore the four classes of policies for problems with mixed state variables (physical, informational, and belief), rather than assuming that we have to always solve Bellman's equation.

- The quality of a policy depends on the quality of a model; the stochastic optimization literature puts relatively little attention into the model of uncertainty, although some attention has been given to the identification of suitable scenarios in a sampled model, and the design of distributionally robust models. There is, of course, an extensive literature on stochastic modeling and uncertainty quantification; we need considerably more research at the intersection of these fields and stochastic optimization.

- Design of algorithms for online (cumulative reward) settings. The vast majority of adaptive search algorithms (stochastic gradient methods, Benders decomposition, Q-learning, approximate dynamic programming) are implemented in an offline context where the goal is to produce a solution that "works well." There are many settings where learning has to be performed online, which means we have to do well as we are learning, which is the standard framework of multiarmed bandit problems. We can bring this thinking into classical stochastic search problems.

- All of the communities described in section 2 focus on expectation-based objectives, yet risk is almost always an issue in stochastic problems. There is a growing literature on the use of risk measures, but we feel that the current literature is only scratching the surface in terms of addressing computational and modeling issues in the context of specific applications.

- Parametric cost function approximations, particularly in the form of modified deterministic models, are widely used in engineering practice (think of scheduling an airline with schedule slack to handle uncertainty). This strategy represents a powerful alternative to stochastic programming for handling multistage stochastic math programs. We envision that this research will consist of computational research to develop and test search algorithms for optimizing parametric CFAs, along with the theoretical analysis of structural results to guide the design of these policies.

- With rare exceptions, authors will pursue one of the four classes of policies we have described above, but it is not always obvious which is best, and it can depend on the characteristics of the data. We need a robust methodology that searches across classes of policies, and performs self-tuning, in an efficient way. Of course, we will always be searching for the ultimate function that replaces all four classes, but we are not optimistic that this will be possible in practice.

- Multiple objectives - Stochastic dynamic problems tend to be richer and more complex, and one byproduct of this is that these problems are often multi-objective. At a minimum, we have to handle risk and reward, but in real applications, there tend to be several important metrics that are being managed.
- Multiple agents - A rich direction to extend this modeling framework is to include multiple agents. This raises issues of communication, coordination and adversarial behavior.

Each of these topics are deep and rich, and could represent entire fields of research.

Acknowledgements

This work was funded in part by AFOSR grant FA9550-12-1-0200, NSF grant CMMI-1537427 and DARPA grant FA8750-17-2-0027.

Aberdeen, D. (2003), 'A (revised) survey of approximate methods for solving partially observable Markov decision processes', *National ICT Australia, Canberra, Australia* pp. 1–41.

Agrawal, S. & Goyal, N. (2012), Analysis of Thompson Sampling for the multi-armed bandit problem, *in* 'Conference on Learning Theory (COLT)', Association for Computational Learning, Edinburgh, pp. 1–21.

Albers, S. (2003), 'Online Algorithms: A Survey', *Mathematical Programming* **97**(1-2), 3–26.

Andradóttir, S. (1998*a*), 'A review of simulation optimization techniques', *1998 Winter Simulation Conference. Proceedings* **1**(0), 151–158.

Andradóttir, S. (1998*b*), Simulation Optimimzation, *in* J. Banks, ed., 'Handbook of simulation', John Wiley & Sons, Hoboken, NJ, chapter 9, pp. 307–333.

Ankenman, B., Nelson, B. L. & Staum, J. (2010), 'Stochastic Kriging for Simulation Metamodeling', *Operations Research* **58**(2), 371–382.

Asmussen, S. & Glynn, P. W. (2007), *Stochastic simulation: algorithms and analysis*, Springer Science & Business Media.

Astrom, K. J. (1970), *Introduction to Stochastic Control Theory*, Dover Publications, Mineola, NY.

Audibert, J.-y. & Bubeck, S. (2010), 'Best Arm Identification in Multi-Armed Bandits', *CoLT* p. 13 p.

Auger, D., Couëtoux, A. & Teytaud, O. (2013), 'Continuous upper confidence trees with polynomial exploration - Consistency', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **8188**(PART 1), 194–209.

Azadivar, F. (1999), Simulation Optimization Methodologies, *in* P. Farrington, H. Nembhard, D. Sturrock & G. Evans, eds, 'Proceedings of the 1999 Winter Simulation Conference', IEEE, pp. 93–100.

Azevedo, A. & Paxson, D. (2014), 'Developing real option game models', *European Journal of Operational Research* **237**(3), 909–920.

Banks, J., Nelson, B. L. & J. S. Carson, I. I. (1996), *Discrete-Event System Simulation*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Bartlett, P. L., Hazan, E. & Rakhlin, A. (2007), 'Adaptive Online Gradient Descent', *Advances in neural information processing systems* pp. 1–8.

Barut, E. & Powell, W. B. (2014), 'Optimal learning for sequential sampling with non-parametric beliefs', *J. Global Optimization* **58**, 517–543.

Bayraksan, G. & Love, D. K. (2015), 'Data-Driven Stochastic Programming Using Phi- Divergences', *Informs TutORials in Operations Research 2014* pp. 1–19.

Bayraksan, G. & Morton, D. (2009), 'Assessing solution quality via sampling in stochastic programs', *TutORials in Operations Research* **514**, 495–514.

Bayraksan, G. & Morton, D. P. (2011), 'A Sequential Sampling Procedure for Stochastic Programming', *Operations Research* **59**(4), 898–913.

Bellman, R. E. (1954), 'The Theory of Dynamic Programming', *Bull. Amer. Math. Soc.* **60**, 503–516.

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press, Princeton, N.J.

Bellman, R. E. & Dreyfus, S. E. (1959), 'Functional approximations and dynamic programming', *Mathematical Tables and Other Aids to Computation* **13**, 247–251.

Bellman, R. E., Glicksberg, I. & Gross, O. (1955), 'On the Optimal Inventory Equation', *Management Science* **1**, 83–104.

Bellman, R., Kalaba, R. & Kotkin, B. (1963), 'Polynomial approximation— a new computational technique in dynamic programming: Allocation processes', *Mathematics of Computation* **17**, 155–161.

Ben-Tal, A., El Ghaoui, L. & Nemirovski, A. (2009), 'Robust Optimization', *Princeton University Press* **53**(3), 464–501.

Ben-Tal, A., Golany, B., Nemirovski, A. & Vial, J.-p. (2005), 'Retailer-Supplier Flexible Commitments Contracts: A Robust Optimization Approach', *Manufacturing & Service Operations Management* **7**(3), 248–271.

Berbeglia, G., Cordeau, J. F. & Laporte, G. (2010), 'Dynamic pickup and delivery problems', *European Journal of Operational Research* **202**(1), 8–15.

Bertsekas, D. (2005), 'Dynamic programming and suboptimal control: A survey from ADP to MPC', *European Journal of Control* **11**(4-5), 310—-334.

Bertsekas, D. P. (2011), *Dynamic Programming and Optimal Control, Vol. II: Approximate Dynamic Programming*, 4 edn, Athena Scientific, Belmont, MA.

Bertsekas, D. P. & Castanon, D. A. (1999), 'Rollout Algorithms for Stochastic Scheduling Problems', *J. Heuristics* **5**, 89–108.

Bertsekas, D. P. & Shreve, S. E. (1978), *Stochastic optimal control: the discrete time case*, Vol. 0, Academic Press.

Bertsekas, D. P. & Tsitsiklis, J. N. (1996), *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA.

Bertsimas, D., Brown, D. B. & Caramanis, C. (2011), 'Theory and applications of robust optimization', *SIAM Review* **53**(3), 464–501.

Bertsimas, D. J. & Sim, M. (2004), 'The Price of Robustness', *Operations Research* **52**(1), 35–53.

Bertsimas, D. J. & Thiele, A. (2006), 'A Robust Optimization Approach to Inventory Theory', *Operations Research* **54**(1), 150–168.

Birge, J. R. & Louveaux, F. (2011), *Introduction to Stochastic Programming*, 2nd edn, Springer, New York.

Blum, J. (1954), 'Multidimensional stochastic approximation methods'', *Annals of Mathematical Statistics* **25**, 737–74462.

Boomsma, T. K., Meade, N. & Fleten, S. E. (2012), 'Renewable energy investments under different support schemes: A real options approach', *European Journal of Operational Research* **220**(1), 225–237.

Borodin, A. & El-Yanniv, R. (1998), *Online Computation and Competitive Analysis*, Cambridge Univ Press, London.

Bouzaiene-Ayari, B., Cheng, C., Das, S., Fiorillo, R. & Powell, W. B. (2014), 'From Single Commodity to Multiattribute Models for Locomotive Optimization : A Comparison of Optimal Integer Programming and Approximate Dynamic Programming', *Transportation Science* pp. 1–24.

Bouzaiene-Ayari, B., Cheng, C., Das, S., Fiorillo, R. & Powell, W. B. (2016), 'From single commodity to multiattribute models for locomotive optimization: A comparison of optimal integer programming and approximate dynamic programming', *Transportation Science.*

Broadie, M., Cicek, D. & Zeevi, a. (2011), 'General Bounds and Finite-Time Improvement for the Kiefer-Wolfowitz Stochastic Approximation Algorithm', *Operations Research* **59**(5), 1211–1224.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. (2012), 'A Survey of Monte Carlo Tree Search Methods', *IEEE Trans. on Computational Intelligence and AI in Games* **4**(1), 1–43.

Bubeck, S. & Cesa-Bianchi, N. (2012), 'Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems', *Foundations and Trends in Machine Learning* **5**(1), 1–122.

Busoniu, L., Babuska, R., De Schutter, B. & Ernst, D. (2010), *Reinforcement Learning and Dynamic Programming using Function Approximators*, CRC Press, New York.

Camacho, E. & Bordons, C. (2003), *Model Predictive Control*, Springer, London.

Cassandra, A. R., Kaelbling, L. P. & Littman, M. L. (1994), Acting Optimally in Partially Observable Stochastic Domains, *in* 'AAAI', pp. 1–6.

Chau, M., Fu, M. C., Qu, H. & Ryzhov, I. O. (2014), Simulation Optimization: A Tutorial Overview and Recent Developments in Gradient-Based Methods, *in* A. Tolk, S. Diallo, I. Ryzhov, L. Yilmaz, S. Buckley & J. Miller, eds, 'Winter Simulation Conference', Informs, pp. 21–35.

Chen, C. H. (1995), An effective approach to smartly allocate computing budget for discrete event simulation, *in* '34th IEEE Conference on Decision and Control', Vol. 34, New Orleans, LA, pp. 2598–2603.

Chen, C. H. (1996), 'A lower bound for the correct subset-selection probability and its application to discrete event system simulations. IEEE Transactions on', *Automatic Control* **41**(8), 1227–1231.

Chen, C.-H. & Lee, L. H. (2011), *Stochastic Simulation Optimization*, World Scientific Publishing Co., Hackensack, N.J.

Chen, C. H., Donohue, K., Yücesan, E. & Lin, J. (2003), 'Optimal computing budget allocation for Monte Carlo simulation with application to product design', *Simulation Modelling Practice and Theory* **11**, 57–74.

Chen, C. H., He, D., Fu, M. C. & Lee, L. H. (2008), 'Efficient simulation budget allocation for selecting an optimal subset', *INFORMS Journal on Computing* **20**(4), 579–595.

Chen, C. H., Yuan, Y., Chen, H. C., Yücesan, E. & Dai, L. (1998), Computing budget allocation for simulation experiments with different system structure, *in* 'Proceedings of the 30th conference on Winter simulation', pp. 735–742.

Chen, H. C., Chen, C. H., Dai, L. & Yucesan, E. (1997), A gradient approach for smartly allocating computing budget for discrete event simulation, *in* J. Charnes, D. Morrice, D. Brunner & J. Swain, eds, 'Proceedings of the 1996 Winter Simulation Conference', IEEE Press, Piscataway, NJ, USA, pp. 398–405.

Chen, S., Reyes, K.-R. G., Gupta, M., Mcalpine, M. C. & Powell, W. B. (2015), 'Optimal learning in Experimental Design Using the Knowledge Gradient Policy with Application to Characterizing Nanoemulsion Stability', *SIAM/ASA J. Uncertainty Quantification* **3**, 320–345.

Chen, V. C. P., Ruppert, D. & Shoemaker, C. A. (1999), 'Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming', *Operations Research* **47**(1), 38–53.

Cheng, B., Asamov, T. & Powell, W. B. (2017), 'Low-Rank Value Function Approximation for Co-optimization of Battery Storage', *IEEE Transactions on Smart Grid*.

Cheng, B., Jamshidi, A. & Powell, W. B. (2014), 'Optimal Learning with a Local Parametric Belief Model', (2006), 1–37.

Chick, S. E. & Gans, N. (2009), 'Economic Analysis of Simulation Selection Problems', *Management Science* **55**(3), 421–437.

Chick, S. E., Branke, J. & Schmidt, C. (2010), 'Sequential sampling to myopically maximize the expected value of information', *INFORMS Journal on Computing* **22**(1), 71–80.

Cinlar, E. (1975), *Introduction to Stochastic Processes*, Prentice Hall, Upper Saddle River, NJ.

Cinlar, E. (2011), *Probability and Stochastics*, Springer, New York.

Collado, R. A., Papp, D. & Ruszczyński, A. (2011), 'Scenario decomposition of risk-averse multistage stochastic programming problems', *Annals of Operations Research* **200**(1), 147–170.

Cressie, N. (1990), 'The origins of kriging', *Mathematical Geology* **22**(3), 239–252.

Dantzig, G. B. (1955), 'Linear programming with uncertainty', *Management Science* **1**, 197–206.

Dantzig, G. B. & Ferguson, A. (1956), 'The Allocation of Aircrafts to Routes: An Example of Linear Programming Under Uncertain Demand', *Management Science* **3**, 45–73.

Defourny, B., Ernst, D. & Wehenkel, L. (2013), 'Scenario Trees and Policy Selection for Multistage Stochastic Programming using Machine Learning', *Informs J. on Computing* pp. 1–27.

DeGroot, M. H. (1970), *Optimal Statistical Decisions*, John Wiley and Sons.

Denardo, E. V. (1982), *Dynamic Programming*, Prentice-Hall, Englewood Cliffs, NJ.

Duchi, J., Hazan, E. & Singer, Y. (2011), 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization', *Journal of Machine Learning Research* **12**, 2121–2159.

Duff, M. O. (2002), 'Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes'.

Duff, M. O., Barto, A. G. & Du, M. O. (1996), Local bandit approximation for optimal learning problems, *in* M. C. Mozer, M. I. Jordan & T. Petsche, eds, 'Proceedings of the 9th International Conference on Neural Information Processing Systems', Department of Computer Science, University of Massachusetts, MIT Press, Cambridge, MA, pp. 1019–1025.

Dupacova, J., Growe-Kuska, N. & Romisch, W. (2003), 'Scenario reduction in stochastic programming: An approach using probability metrics', *Math. Program.,* **Ser. A 95**, 493511.

Durante, J. L., Nascimento, J. & Powell, W. B. (2017), Backward Approximate Dynamic Programming with Hidden Semi-Markov Stochastic Models in Energy Storage Optimization, Technical report, Princeton University, Princeton NJ.

Dvoretzky, A. (1956), On Stochastic Approximation, *in* J. Neyman, ed., 'Proceedings 3rd Berkeley Symposium on Mathematical Statistics and Probability', University of California Press, pp. 39–55.

Ermoliev, Y. (1968), 'On the stochastic quasi-gradient methods and stochastic quasi-Feyer sequence', *Kibernetika*.

Feng, Y. & Gallego, G. (1995), 'Optimal Starting Times for End-of-Season Sales and Optimal Stopping Times for Promotional Fares', *Management Science* **41**(8), 1371–1391.

Fliege, J. & Werner, R. (2014), 'Robust multiobjective optimization and applications in portfolio optimization', *European Journal of Operational Research* **234**(2), 422–433.

Frazier, P. I. & Powell, W. B. (2010), 'Paradoxes in Learning and the Marginal Value of Information', *Decision Analysis* **7**(4), 378–403.

Frazier, P. I., Powell, W. B. & Dayanik, S. (2008), 'A knowledge gradient policy for sequential information collection', *SIAM Journal on Control and Optimization* **47**(5), 2410–2439.

Frazier, P. I., Powell, W. B. & Dayanik, S. (2009), 'The Knowledge-Gradient Policy for Correlated Normal Beliefs', *INFORMS Journal on Computing* **21**(4), 599–613.

Fu, M. C. (2002), 'Optimization for simulation: Theory vs. practice', *Informs Journal on Computing* **14**(3), 192–215.

Fu, M. C. (2014), *Handbook of Simulation Optimization*, Springer, New York.

Gabillon, V., Ghavamzadeh, M. & Lazaric, A. (2012), 'Best arm identification: A unified approach to fixed budget and fixed confidence', *Nips* pp. 1–9.

Gabrel, V., Murat, C. & Thiele, A. (2014), 'Recent advances in robust optimization: An overview', *European Journal of Operational Research* **235**(3), 471–483.

Ginebra, J. & Clayton, M. K. (1995), 'Response Surface Bandits', *Journal of the Royal Statistical Society. Series B (Methodological)* **57**(4), 771–784.

Girardeau, P., Leclere, V. & Philpott, A. B. (2014), 'On the Convergence of Decomposition Methods for Multistage Stochastic Convex Programs', *Mathematics of Operations Research* **40**(1), 130–145.

Gittins, J. (1979), 'Bandit processes and dynamic allocation indices', *Journal of the Royal Statistical Society. Series B (Methodological)* **41**(2), 148–177.

Gittins, J. (1989), 'Multi-armed Bandit Allocation Indices', *Wiley and Sons: New York.*

Gittins, J. & Jones, D. (1974), A dynamic allocation index for the sequential design of experiments, *in* J. Gani, ed., 'Progress in statistics', North Holland, Amsterdam, pp. 241—-266.

Gittins, J., Glazebrook, K. D. & Weber, R. R. (2011), *Multi-Armed Bandit Allocation Indices*, John Wiley & Sons, New York.

Goh, J. & Sim, M. (2010), 'Distributionally robust optimization and its tractable approximations', *Operations Research* **58**(4, Part 1 of 2), 902–917.

Hagspiel, V., Huisman, K. J. & Nunes, C. (2015), 'Optimal technology adoption when the arrival rate of new technologies changes', *European Journal of Operational Research* **243**(3), 897–911.

Hastie, T. J., Tibshirani, R. J. & Friedman, J. H. (2009), *The elements of statistical learning : data mining, inference, and prediction*, Springer, New York.

Heitsch, H. & Romisch, W. (2009), 'Scenario tree modeling for multistage stochastic programs', *Mathematical Programming* **118**, 371–406.

Heyman, D. P. & Sobel, M. (1984), *Stochastic Models in Operations Research, Volume II: Stochastic Optimization*, McGraw Hill, New York.

Higle, J. L. & Sen, S. (1991), 'Stochastic decomposition: An algorithm for two-stage linear programs with recourse', *Mathematics of Operations Research* **16**(3), 650–669.

Hong, J. & Nelson, B. L. (2006), 'Discrete Optimization via Simulation Using COMPASS', *Operations Research* **54**(1), 115–129.

Howard, R. A. (1960), Dynamic programming and Markov processes, MIT Press, Cambridge, MA.

Infanger, G. & Morton, D. P. (1996), 'Cut Sharing for Multistage Stochastic Linear Programs with Interstage Dependency', *Mathematical Programming* **75**, 241–256.

Ivanov, D. & Sokolov, B. (2013), 'Control and system-theoretic identification of the supply chain dynamics domain for planning, analysis and adaptation of performance under uncertainty', *European Journal of Operational Research* **224**(2), 313–323.

Jaakkola, T., Jordan, M. I. & Singh, S. P. (1994), 'On the Convergence of Stochastic Iterative Dynamic Programming Algorithms', *Neural Computation* **6**, 1185—-1201.

Jaillet, P. & Wagner, M. R. (2006), 'Online Routing Problems: Value of Advanced Information as Improved Competitive Ratios', *Transportation Science* **40**(2), 200–210.

Jiang, D. R. & Powell, W. B. (2016*a*), Optimal Policies for Risk-Averse Electric Vehicle Charging with Spot Purchases.

Jiang, D. R. & Powell, W. B. (2016*b*), Risk-averse approximate dynamic programming with quantile-based risk measures, Technical report.

Jiang, D. R., Al-Kanj, L. & Powell, W. B. (2017), Monte Carlo Tree Search with Sampled Information Relaxation Dual Bounds, Technical report, University of Pittsburgh, Pittsburgh, PA.

Jones, D. R. (2001), 'A Taxonomy of Global Optimization Methods Based on Response Surfaces', *Journal of Global Optimization* pp. 345–383.

Jones, D., Schonlau, M. & Welch, W. (1998), 'Efficient global optimization of expensive black-box functions', *Journal of Global Optimization* **13**(4), 455—-492.

Judd, K. L. (1998), *Numerical Methods in Economics*, MIT Press.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), 'Reinforcement learning: a survey', *J. Artif. Intell. Res.* **4**, 237–285.

Kall, P. & Wallace, S. (2009), *Stochastic Programming*, Vol. 10, John Wiley & Sons, Hoboken, NJ.

Karatzas, I. (1988), 'On the pricing of American options', *Applied Mathematics and Optimization* **17**(1), 37–60.

Kaufmann, E., Cappé, O. & Garivier, A. (2016), 'On the Complexity of Best-Arm Identification in Multi-Armed Bandit Models', *Journal of Machine Learning Research* **17**, 1–42.

Kesten, H. (1958), 'Accelerated Stochastic Approximation', *The Annals of Mathematical Statistics* **29**, 41–59.

Keyvanshokooh, E., Ryan, S. M. & Kabir, E. (2016), 'Hybrid robust and stochastic optimization for closed-loop supply chain network design using accelerated Benders decomposition', *European Journal of Operational Research* **249**(1), 76–92.

Kim, S.-H. & Nelson, B. L. (2007), Recent advances in ranking and selection, IEEE Press, Piscataway, NJ, USA, pp. 162–172.

King, A. J. & Wallace, S. W. (2012), *Modeling with Stochastic Programming*, Springer Verlag, New York.

Kingma, D. P. & Ba, J. L. (2015), Adam: a Method for Stochastic Optimization, *in* 'International Conference on Learning Representations 2015', pp. 1–15.

Kirk, D. E. (2004), *Optimal Control Theory: An introduction*, Dover, New York.

Kleijnen, J. P. (2014), 'Simulation-optimization via Kriging and bootstrapping: a survey', *Journal of Simulation* **8**(4), 241–250.

Kleijnen, J. P. (2017), 'Regression and Kriging metamodels with their experimental designs in simulation: A review', *European Journal of Operational Research* **256**(1), 1–16.

Kleywegt, A. J., Shapiro, A. & Homem-de Mello, T. (2002), 'The Sample Average Approximation Method for Stochastic Discrete Optimization', *SIAM Journal on Optimization* **12**(2), 479–502.

Kozmík, V. & Morton, D. P. (2014), 'Evaluating policies in risk-averse multi-stage stochastic programming', *Mathematical Programming* **152**(1-2), 275–300.

Kupper, M. & Schachermayer, W. (2009), 'Representation results for law invariant time consistent functions', *Mathematics and Financial Economics* **2**(3), 189–210.

Kushner, H. J. & Clark, S. (1978), *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag, New York.

Kushner, H. J. & Kleinman, A. J. (1971), 'Accelerated Procedures for the Solution of Discrete Markov Control Problems', *IEEE Transactions on Automatic Control* **16**, –2147–152.

Kushner, H. J. & Yin, G. G. (2003), *Stochastic Approximation and Recursive Algorithms and Applications*, Springer, New York.

Lai, T. L. & Robbins, H. (1985), 'Asymptotically efficient adaptive allocation rules', *Advances in Applied Mathematics* **6**(1), 4–22.

Lee, J. H. (2011), 'Model predictive control: Review of the three decades of development', *International Journal of Control, Automation and Systems* **9**(3), 415–424.

Lewis, F. L., Vrabie, D. & Syrmos, V. L. (2012), *Optimal Control*, 3rd edn, John Wiley & Sons, Hoboken, NJ.

Löhndorf, N. (2016), 'An empirical analysis of scenario generation methods for stochastic optimization', *European Journal of Operational Research* **255**(1), 121–132.

Longstaff, F. A. & Schwartz, E. S. (2001), 'Valuing American options by simulation: A simple least squares approach', *The Review of Financial Studies* **14**(1), 113–147.

Lovejoy, W. S. (1991), 'A survey of algorithmic methods for partially observed Markov decision processes', *Annals of Operations Research* **28**(1), 47–651.

Luo, J., Hong, L. J., Nelson, B. L. & Wu, Y. (2015), 'Fully Sequential Procedures for Large-Scale Ranking-and-Selection Problems in Parallel Computing Environments', *Operations Research* **63**(5), 1177–1194.

Ma, Y., Chu, C. & Zuo, C. (2010), 'A survey of scheduling with deterministic machine availability constraints', *Computers and Industrial Engineering* **58**(2), 199–211.

Mes, M. R. K., Powell, W. B. & Frazier, P. I. (2011), 'Hierarchical Knowledge Gradient for Sequential Sampling', *Journal of Machine Learning Research* **12**, 2931–2974.

Moazeni, S., Powell, W. B., Defourny, B. & Bouzaiene-ayari, B. (2017), 'Parallel Nonstationary Direct Policy Search for Risk-Averse Stochastic Optimization', *Informs J. on Computing* **29**(2), 332–349.

Montgomery, D. (2000), *Design and Analysis of Experiments*, John Wiley & Sons Inc.

Morari, M., Lee, J. H. & Garc, C. E. (2002), *Model Predictive Control*, Springer-Verlag, New York.

Moustakides, G. V. . (1986), 'Optimal Stopping Times for Detecting Changes in Distributions', *Annals of Statistics* **14**(4), 1379–1387.

Munos, R. (2014), 'From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning', *Foundations and Trends® in Machine Learning* **7**(1), 1–129.

Negoescu, D. M., Frazier, P. I. & Powell, W. B. (2010), 'The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery', *INFORMS Journal on Computing* pp. 1–18.

Nemhauser, G. L. (1966), *Introduction to dynamic programming*, John Wiley & Sons, New York.

Ni, E. C., Henderson, S. G. & Hunter, S. R. (2016), 'Efficient Ranking and Selection in Parallel Computing Environments', *Operations Research* **65**(3), 821–836.

Nisio, M. (2014), *Stochastic Control Theory: Dynamic Programming Principle*, Springer, New York.

Oliehoek, F. A., Spaan, M. T. & Vlassis, N. (2008), 'Optimal and approximate Q-value functions for decentralized POMDPs', *Journal of Artificial Intelligence Research* **32**, 289–353.

Orabona, F. (2014), Simultaneous model selection and optimization through parameter-free stochastic learning, *in* 'Advances in Neural Information Processing Systems', pp. 1–9.

Pereira, M. F. & Pinto, L. M. V. G. (1991), 'Multi-stage stochastic optimization applied to energy planning', *Mathematical Programming* **52**, 359–375.

Perkins, R. T. & Powell, W. B. (2017), Stochastic Optimization with Parametric Cost Function Approximations.

Pflug, G. (1988*a*), *Numerical Methods in Stochastic Programming*, Springer-Verlag.

Pflug, G. (1988*b*), Stepsize rules, stopping times and their implementation in stochastic quasi-gradient algorithms, *in* 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, New York, pp. 353–372.

Philpott, A. B. & de Matos, V. (2012), 'Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion', *European Journal of Operational Research* **218**(2), 470–483.

Philpott, A. B., De Matos, V. & Finardi, E. (2013), 'On Solving Multistage Stochastic Programs with Coherent Risk Measures', *Operations Research* **51**(4), 957–970.

Pillac, V., Gendreau, M., Guéret, C. & Medaglia, A. L. (2013), 'A review of dynamic vehicle routing problems', *European Journal of Operational Research* **225**(1), 1–11.

Pineau, J., Gordon, G. & Thrun, S. (2003), Point-based value iteration: An anytime algorithm for POMDPs, *in* 'IJCAI International Joint Conference on Artificial Intelligence', pp. 1025–1030.

Poor, H. V. & Hadjiliadis, O. (2009), *Quickest Detection*, Cambridge University Press, Cambridge, U.K.

Powell, W. B. (2007), 'Approximate Dynamic Programming: Solving the curses of dimensionality'.

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

Powell, W. B. (2014), 'Clearing the Jungle of Stochastic Optimization', *Bridging Data and Decisions* (January 2015), 109–137.

Powell, W. B. (2016), *A Unified Framework for Optimization under Uncertainty.*

Powell, W. B. & George, A. P. (2006), 'Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming', *Journal of Machine Learning* **65**(1), 167–198.

Powell, W. B. & Meisel, S. (2016*a*), 'Tutorial on Stochastic Optimization in Energy - Part II: An Energy Storage Illustration', *IEEE Transactions on Power Systems.*

Powell, W. B. & Meisel, S. (2016*b*), 'Tutorial on Stochastic Optimization in Energy II: An energy storage illustration', *IEEE Transactions on Power Systems* **31**(2), 1459–1467.

Powell, W. B. & Ryzhov, I. O. (2012), *Optimal Learning*, John Wiley & Sons, Hoboken, NJ.

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

Protopappa-Sieke, M. & Seifert, R. W. (2010), 'Interrelating operational and financial performance measurements in inventory control', *European Journal of Operational Research* **204**(3), 439–448.

Puterman, M. (2005), *Markov Decision Processes*, 2nd edn, John Wiley & Sons Inc, Hoboken, NJ.

Qu, H., Ryzhov, I. O. & Fu, M. C. (2012), Ranking and selection with unknown correlation structures, *in* A. U. C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, ed., 'Proceedings - Winter Simulation Conference', number 1995.

Ramirez-Nafarrate, A., Baykal Hafizoglu, A., Gel, E. S. & Fowler, J. W. (2014), 'Optimal control policies for ambulance diversion', *European Journal of Operational Research* **236**(1), 298–312.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The Annals of Mathematical Statistics* **22**(3), 400–407.

Rockafellar, R. T. & Uryasev, S. (2000), 'Optimization of conditional value-at-risk', *Journal of Risk* **2**, 21–41.

Rockafellar, R. T. & Uryasev, S. (2002), 'Conditional value-at-risk for general loss distributions', *Journal of Banking & Finance* **26**, 1443–1471.

Rockafellar, R. T. & Uryasev, S. (2013), 'The fundamental risk quadrangle in risk management, optimization, and statistical estimation', *Surveys in Operations Research and Management Science* **18**(1), 33–53.

Rockafellar, R. T. & Wets, R. J.-B. (1991), 'Scenarios and policy aggregation in optimization under uncertainty', *Mathematics of Operations Research* **16**(1), 119–147.

Ross, S. M. (2002), *Simulation*, Academic Press, New York.

Ross, S., Pineau, J. & Chaib-Draa, B. (2008*a*), 'Theoretical Analysis of Heuristic Search Methods for Online POMDPs.', *NIPS* **20**, 1216–1225.

Ross, S., Pineau, J., Paquet, S. & Chaib-draa, B. (2008*b*), 'Online planning algorithms for POMDPs', *Journal of Artificial Intelligence Research* **32**, 663–704.

Rubinstein, R. Y. & Kroese, D. P. (2017), *Simulation and the Monte Carlo Method*, 3rd edn, John Wiley & Sons, Hoboken, NJ.

Russo, D. & Van Roy, B. (2014), 'Learning to Optimize via Posterior Sampling', *Mathematics of Operations Research* **39**(4), 1221–1243.

Ruszczyński, A. (2014), Advances in Risk-Averse Optimization, *in* 'INFORMS Tutorials in Operations Research', INFORMS, Baltimore, MD, pp. 168–190.

Ruszczyński, A. & Shapiro, A. (2006), 'Optimization of Convex Risk Functions', *Mathematics of Operations Research* **31**(3), 433–452.

Ryzhov, I. O. (2016), 'On the Convergence Rates of Expected Improvement Methods', *Operations Research* **64**(6), 1515–1528.

Ryzhov, I. O. & Powell, W. B. (2010), Approximate Dynamic Programming With Correlated Bayesian Beliefs, *in* 'Forty-Eighth Annual Allerton Conference on Communication, Control, and Computing', Monticello, IL.

Ryzhov, I. O., Mes, M. R. K., Powell, W. B. & van den Berg, G. A. (2017), Bayesian exploration strategies for approximate dynamic programming, Technical report, University of Maryland, College Park.

Salas, D. & Powell, W. B. (2015), 'Benchmarking a Scalable Approximate Dynamic Programming Algorithm for Stochastic Control of Multidimensional Energy Storage Problems', *Informs J. on Computing* pp. 1–41.

Schildbach, G. & Morari, M. (2016), 'Scenario-based model predictive control for multi-echelon supply chain management', *European Journal of Operational Research* **252**(2), 540–549.

Sen, S. & Zhou, Z. (2014), 'Multistage stochastic decomposition: A bridge between stochastic programming and approximate dynamic programming', *SIAM J. Optimization* **24**(1), 127–153.

Senn, M., Link, N., Pollak, J. & Lee, J. H. (2014), 'Reducing the computational effort of optimal process controllers for continuous state spaces by using incremental learning and post-decision state formulations', *J. of Process Control2* **24**, 133–143.

Sethi, S. P. & Thompson, G. L. (2000), *Optimal Control Theory*, 2 edn, Kluwer Academic Publishers, Boston.

Shani, G., Pineau, J. & Kaplow, R. (2013), 'A survey of point-based POMDP solvers', *Autonomous Agents and Multi-Agent Systems* **27**(1), 1–51.

Shapiro, A. (2011), 'Analysis of stochastic dual dynamic programming method', *European Journal of Operational Research* **209**(1), 63–72.

Shapiro, A. (2012), 'Minimax and risk averse multistage stochastic programming', *European Journal of Operational Research* **219**(3), 719–726.

Shapiro, A. & Wardi, Y. (1996), 'Convergence Analysis of Stochastic Algorithms', *Mathematics of Operations Research* **21**, 615–628.

Shapiro, A., Dentcheva, D. & Ruszczyński, A. (2014), *Lectures on Stochastic Programming: Modeling and theory*, 2 edn, SIAM, Philadelphia.

Shapiro, A., Tekaya, W., Da Costa, J. P. & Soares, M. P. (2013), 'Risk neutral and risk averse Stochastic Dual Dynamic Programming method', *European Journal of Operational Research* **224**(2), 375–391.

Sherif, Y. S. & Smith, M. L. (1981), 'Optimal maintenance models for systems subject to failureA Review', *Naval Reseach Logistics Quarterly* **28**(1), 47–74.

Shiryaev, A. N. (1978), *Optimal Stopping Rules*, Springer, Moscow.

Shor, N. K. (1979), *The Methods of Nondifferentiable Op[timization and their Applications*, Naukova Dumka, Kiev.

Si, J., Barto, A. G., Powell, W. B. & Wunsch, D. (2004), 'Handbook of learning and approximate dynamic programming', *Wiley-IEEE Press.*

Simao, H. P., Day, J., George, A. P., Gifford, T., Powell, W. B. & Nienow, J. (2009), 'An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application', *Transportation Science* **43**(2), 178–197.

Skinner, D. C. (1999), *Introduction to Decision Analysis*, Probabilistic Publishing, Gainesville, Fl.

Slotnick, S. A. (2011), 'Order acceptance and scheduling: A taxonomy and review', *European Journal of Operational Research* **212**(1), 1–11.

Smallwood, R. D., Sondik, E. J. & Oct, N. S. (1973), 'The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon', *Operations Research* **21**(5), 1071–1088.

Smith, R. C. (2014), *Uncertainty Quantification: Theory, Implementation, and Applications*, SIAM, Philadelphia.

Smith, T. & Simmons, R. (2005), Point-Based POMDP Algorithms: Improved Analysis and Implementation, *in* 'Uai', pp. 542–549.

Sondik, E. J. (1971), The optimal control of partially observable Markov decision processes, PhD thesis, Stanford University.

Sondik, E. J. (1978), 'The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs', *Operations Research* **26**(2), 282–304.

Sontag, E. (1998), 'Mathematical Control Theory, 2nd ed.', *Springer* pp. 1–544.

Spall, J. C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, simulation and control*, John Wiley & Sons, Hoboken, NJ.

Stein, M. L. (1999), *Interpolation of spatial data: Some theory for kriging*, Springer Verlag, New York.

Stengel, R. F. (1986), *Stochastic optimal control: theory and application*, John Wiley & Sons, Hoboken, NJ.

Sullivan, T. (2015), *Introduction to Uncertainty Quantification*, Springer, New York.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning*, MIT Press, Cambridge, MA.

Swisher, J. R., Hyden, P. D. & Schruben, L. W. (2000), 'A survey of simulation optimization techniques and procedures - Simulation Conference Proceedings, 2000. Winter', pp. 119–128.

Szepesvári, C. (2010), *Algorithms for Reinforcement Learning*, Morgan and Claypool.

Thompson, W. R. (1933), 'On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples', *Biometrika* **25**(3/4), 285–294.

Topaloglu, H. & Powell, W. B. (2006), 'Dynamic Programming Approximations for Stochastic, Time-Staged Integer Multicommodity Flow Problems', *Informs Journal on Computing* **18**(1), 31–42.

Tsitsiklis, J. & Van Roy, B. (2001), 'Regression methods for pricing complex American-style options', *IEEE Transactions on Neural Networks* **12**(4), 694–703.

Tsitsiklis, J. N. (1994), 'Asynchronous stochastic approximation and Q-learning', *Machine Learning* **16**, 185–202.

Van Slyke, R. M. & Wets, R. J.-B. (1969), 'L-shaped linear programs with applications to optimal control and stochastic programming', *SIAM Journal of Applied Mathematics* **17**, 638–663.

Werbos, P. J. (1974), Beyond regression: new tools for prediction and analysis in the behavioral sciences, PhD thesis, Harvard University.

Werbos, P. J. (1989), Backpropagation and neurocontrol: A review and prospectus, *in* 'IJCNN, International Joint Conference on Neural Networks', pp. 209—-216.

Werbos, P. J. (1990), 'Backpropagation Through Time: What It Does and How to Do It', *Proceedings of the IEEE* **78**(10), 1550–1560.

Werbos, P. J. (1992), Approximate Dynamic Programming for Real-Time Control and Neural Modelling, *in* D. J. White & D. A. Sofge, eds, 'Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches'.

Werbos, P. J. (1994), *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, John Wiley & Sons, New York.

White, D. & Sofge, D. (1992), *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*, Van Nostrand Reinhold Company, New York.

Wiesemann, W., Kuhn, D. & Sim, M. (2014), 'Distributionally Robust Convex Optimization', *Operations Research* **62**(6), 1358–1376.

Wolfowitz, J. (1952), 'On the stochastic approximation method of Robbins and Monro', *Annals Math. Stat.* **23**, 457–461.

Wu, J., Poloczek, M., Wilson, A. G. & Frazier, P. I. (2017), Bayesian Optimization with Gradients, Technical report, Cornell University, Ithaca.

Xu, H., Caramanis, C., Mannor, S. & Caramanis, C. (2012), 'A Distributional Interpretation of Robust Optimization', *Mathematics of Operations Research* **37**(1), 95–110.

Yong, J. & Zhou, X. Y. (1999), *Stochastic Controls: Hamiltonian Systems and HJB Equations*, Springer, New York.

Yu, M., Takahashi, S., Inoue, H. & Wang, S. (2010), 'Dynamic portfolio optimization with risk control for absolute deviation model', *European Journal of Operational Research* **201**(2), 349–364.

Zugno, M. & Conejo, A. J. (2015), 'A robust optimization approach to energy and reserve dispatch in electricity markets', *European Journal of Operational Research* **247**(2), 659–671.